POLITECNICO DI MILANO
Laurea Magistrale in Computer Science and Engineering
Dipartimento di Elettronica, Informazione e Bioingegneria

POLITECNICO
MILANO 1863

Assessment and improvement of state-of-the-art robot binaural hearing

AI & R Lab
Laboratorio di Intelligenza Artificiale
e Robotica del Politecnico di Milano

Relatore: Prof. Matteo Matteucci
Correlatore: Dott. Ing. Giulio Angelo Fontana

Tesi di Laurea di:
Simone Pradolini, matricola 823147

Anno Accademico 2015-2016

Ai miei nonni

Contents

Summary

Sound source localization is an easy and natural task for humans, but it's very hard to create a model for it and make it efficient and feasible for a machine. How human beings can detect where another person is speaking from is based on a very complex process, and, in the last years, many studies on these types of algorithms were developed, based on different algebraic and mathematical models. The audio processing tasks have not been deeply developed in robotics and artificial intelligence. The binaural speech localization task is an ongoing branch of research whose future development in robotics are fundamental to obtain machines that can interact with human beings.

The aim of this thesis is to analyze some of the existing state-of-the-art systems, to find out pros and cons making experiments and simulations with Matlab and in the laboratory and to obtain new knowledge for a new branch of research. The final aim is to propose the assessment and the optimization of a state-of-the-art binaural speech localization algorithm for a robotic application.

We started our work searching for the most advanced sound source localization systems developed in the last years, we selected and analyzed the algorithms that can be applied on a robot and then we characterised the state-of-the-art. We built up an experimental setup to assess an algorithm in a real environment and to test different assumptions we did on the audio signal processing useful to obtain better localization performance than the original algorithms. This research led to the proposal of an improved localization algorithm that works not only in ideal situations, but also in a common reverberating room.

Ringraziamenti

Mi sembra doveroso ringraziare per prime le persone che mi hanno guidato in questa ricerca. Un grazie sincero al Dott. Ing. Giulio Angelo Fontana e al Prof. Matteo Matteucci e a tutto l'AIRLab. Ho trovato un laboratorio accogliente che mi ha permesso di lavorare con serenitá e di poter organizzare il mio lavoro in autonomia, così che ho potuto fare una prima vera esperienza di ricerca sperimentale, ma allo stesso tempo sono stato sempre seguito e consigliato da persone competenti e disponibili per qualsiasi necessitá (anche aprire il laboratorio durante le feste di Natale).
Ringrazio di cuore la mia famiglia perché, nonostante il momento difficile, hanno comunque fatto sacrifici e mi hanno permesso di portare a termine questo percorso. Grazie a loro ho vissuto fin da piccolo nel mondo della musica e delle tecnologie e se ho fatto questi studi é grazie alla passione che mi hanno trasmesso.
Un ringraziamento particolare va ai miei migliori amici e alle persone speciali che mi hanno accompagnato in questi due anni e mezzo. Ho sentito il loro appoggio e la loro forza ed é grazie a loro che ho superato momenti di difficoltá e di sconforto. Grazie alla loro vicinanza sono andato avanti, non mi sono arreso e sono riuscito a raggiungere questo importante obiettivo.
Un ultimo ringraziamento va alle persone che ho incontrato durante i miei studi: colleghi, compagni di studio, professori, personale del Polo di Como del Politecnico di Milano. Se ho potuto terminare i miei studi é anche grazie all'ambiente in cui ho vissuto in questi anni, in cui ho trovato un'ottima qualitá di insegnamento, un ambiente piccolo, ma prezioso, ricco e di respiro internazionale.

Chapter 1

Introduction

"Hello?
Is there anybody in there?
Just nod if you can hear me.
Is there anyone at home?"

Pink Floyd, Comfortably Numb

Localization of sound sources in the space around us and interaction with them are natural processes for human beings, but the way we can achieve the localization of a sound source is based on a very complex process. There are noticeable difficulties when we try to create an efficient and feasible mathematical model that lets a machine execute these tasks. Indeed for a robot, understanding where is a person speaking by only using audio sensors needs a complex geometric interpretation of the data it receives and it requires heavy calculations that must produce results in real time. A useful algorithm must be sufficiently precise, fast and computationally cheap at the same time.

In the last years, many algorithms have been developed, based on different algebraic and mathematical models (e.g., ITD, TDOA, HRTF), using a deep knowledge of different scientific fields (e.g., acoustics, analog/digital signal processing, physiology, machine intelligence), trying to reproduce the performance of the human auditory system.
The aim of this work has been to analyze some of the state-of-the-art in this research field and to find out advantages and disadvantages through experiments. We implemented some of the localization algorithms in the Matlab framework using simulated or already available data and also using new datasets that we created. We built up also a real setup to make

other experiments on rooms that are different from the ones described by the original authors, to really understand the most important features and the drawbacks of these algorithms and their sensitivity to environmental features.
Our results aim at creating a feasible system that can be used in real environment on a real machine and that can localize one person speaking in the vicinity of a robot with a reasonable precision and in real time (or in a few milliseconds so that we can consider the localization instantaneous).

The final step has been the proposal of an improved localization algorithm that works not only in ideal situations, but also in a common reverberating room. We used well-known systems as a starting point, we tested many new hypotheses on the task used by the original authors of the algorithms and finally we presented a new working proposal, a sound source localization algorithm that is feasible for a robot and that can work independently from the environment in which the robot is.

1.1 Thesis outline

Sound localization is a complex topic, that needs a deep knowledge in many scientific fields (e.g., physiology, biology, physics, acoustics, signal processing, machine intelligence). So, the first step of our work has been the research of the basilar features required to understand the topic. We focused our attention on physics, acoustics and sound wave propagation theories [4], [8], psychology and physiology [19], analog and signal processing [35] [20]. The referenced books and articles are very useful as a starting point for understanding this field of research.

Then, we investigated state-of-the-art localization algorithms (e.g., [2], [12], [13], [15], [43]), classifying the different types of systems. Some studies [14] [23] are more focused on simulating the human auditory system, others [11], [45] aim to create an efficient algorithm focusing on the physics of sound waves and acoustics principles, further ones [1] [5] [7] ignore the physiological aspect and search for algebraic solutions that can better fit the data and that can have results comparable with human capabilities. This analysis has underlined where are possible drawbacks, how to achieve the same goal in different ways, which features can be employed and why, which algebraic tools are more suitable for use for audio input.

The next step has been an in-depth analysis of some of these systems, the ones that seemed to us more efficient, with the best results and the

more impressive structure. We focused our attention on [12], [15], [43] and [45]. We chose these papers because their authors achieve good result, each one in a completely different way with respect to the other ones. Authors develop different algorithms according to the different goals that they want to achieve. We dug into the approaches of these papers, because we wanted to find out how they work, which preliminary hypothesis they used and how these are linked to the result they obtained. This analysis told us important information about the processing we could do on input data, the features we could develop and use in our framework, how parameters could be tuned and which type of result we could expect.

After studying the state-of-the-art, we have reproduced the experiments of the INRIA group described in [14] and [12]. We started from [12], taking its original Matlab code, that is freely available on the website of the INRIA group department and checking its results. This algorithm basically works in two steps: a training stage using white noise sources all around the sensors and a test stage in which the authors recorded some people talking from different positions in a dataset used to test the algorithm. Also these datasets are freely available and in the first step we used them simulating the entire experiment on Matlab. The core of the system is a Piecewise Affine Mapping, an algebraic tool that allows to link the position of an incoming sound to a feature vector, composed by concatenating ILD (*Interaural Level Difference*) and IPD (*Interaural Phase Difference*), i.e., the two feature vectors used in this case. Starting from the original code, we understood how the code works and on which parameter Deleforge et al. acted. We noticed a scarce preprocessing on the incoming data. No analysis of sampling frequency, spectral components, windowing, filtering is present in the paper they published. So, we made some changes in the code, we added a new preprocessing stage and we tested all the parameters trying to achieve better results than the original ones. We executed the Matlab code several times, changing each time some parameter, finding out a new configuration of values that allows the system to reach better results than the original code.

We found that Deleforge et al. overlooked another important cue in this research: the influence of the acoustic features of the room on the algorithm. An analysis of the effect of reverberation and of the dimension of the room is essential to understand how the sound waves propagate from the source to the microphones and how much they can affect the efficiency of the algorithm. From this we built up an experimental setup to reproduce the entire experiment in the AIRLab (*Artificial Intelligence and Robotics Laboratory*

at the Department of Electronics, Information and Bioengineering of Politecnico di Milano). The main setup was composed by an array of eight MEMS microphones, an external sound card and an experimental board, all of them by STMicroelectronics. With this system we can recorded eight audio tracks all together in PCM at 16KHz, 16 bit. As audio framework, we used Audacity and also Matlab to handle input data. As audio sources, we used a computer speaker already present in the laboratory and a Genelec 8030, a professional active loudspeaker. Before doing any experiment and implementing the Deleforge algorithm in a real environment, we tested this setup and we figured out some problems on the lab speaker and, most important, on three MEMS that did not work properly.

With this setup we acquired two new audio datasets, one for the training and the other one for the test stage of the Deleforge algorithm, composed by audio test signals and speech signals emitted by the Genelec speaker or the lab one at different positions in the semiplane in front of the microphone array. Each dataset contains all the possible combination of stereo audio signals available with 5 microphones in addition to the sensors positions and the sources positions. Positions were accurately captured using OptiTrack, a commercial 3D localization system composed by position markers, infrared cameras and processing software that is able to localize the markers. Then, we used our new dataset on the already modified Deleforge code to test its effectiveness on new data, in a new environment and with a real setup. We stressed the algorithm to understand how it changes the results changing its parameters and if the hypothesis done in the previous step on the original code can be demonstrated or not.

On the next step we focused on the Willert algorithm [45]. This algorithm is more related to the physics of sound, introducing a deeper digital processing and analysis stage than the Deleforge one. Willert uses as audio features ITD and ILD as Deleforge, but with a simpler mathematical tool, a Gaussian Mixture model obtained thanks to an Expectation-Maximization process, with which he obtains worse results than Deleforge. This algorithm seemed interesting because we could modify it easily using the knowledge we got in previous experiments and implementing a Matlab application that could achieve comparable results with respect to the original systems described by Willert and by Deleforge. We created an algorithm taking into consideration all the observation we made during the research, comparing our results with the original ones, we described our results and which will be the future approach on this field.

1.2 Thesis structure

This thesis is organized in the following way:

- in Chapter 2 we describe the main mathematical and psychoacoustic features described in the whole research and the most important localization systems developed in the last years;
- in Chapter 3 we focus our attention on two algorithms, e.g., those developed by Deleforge and Willert proposals and we describe their pros and cons in a detailed way;
- in Chapter 4 we describe the first experiments we did on the Deleforge algorithm, how it works originally, which test we did on that code, the results we obtained and the modifications we proposed on the original algorithm;
- in Chapter 5 we describe the experimental setup we used to validate our assumptions and how we built up new original datasets. We describe also the experiment done with our framework and the final conclusion on the Deleforge algorithm, with the proposed modifications and new implementation;
- in Chapter 6 we described our conclusions and some possible approaches, i.e., the one based on the Willert algorithm, the modification we proposed on that code based on the result of all the previous experiments, the Matlab implementation and the results;
- Appendix A includes the Matlab code used in Deleforge algorithms, both the original and the improved one, as described in Chapters 4 and 5;
- Appendix B includes the implementation of the Willert algorithm, as described in Chapter 6.

Chapter 2

State-of-the-art on binaural localization algorithms

During this chapter, we cite the most important mathematical tools and acoustics features that are used by algorithms for sound localization. We describe more in depth the basic knowledge you need to have a full understanding of this research, the features used and the mathematical tools cited in the whole thesis. We present also the most important state-of-the-art binaural speech localization systems.

2.1 Psychoacoustics measurements

Sound localization is one of the primary abilities of human beings. It's a fundamental feature, used for self-orientation and for attention control, thanks to that we can interact each other. So, if we want to develop a computational model of this task, it's mandatory to understand physiologically how the human auditory system works.

The first task we need to analyze is how *ear* works. We can divide it in three parts:

- The *outer ear*, composed by the *pinna* (the irregular external space of the ear) and the *ear duct*. The strange structure and dimension of the pinna is not casual. It works as a directional and frequency filter and conveys the sound waves to the ear duct, boosting the frequencies in the interval 1-4KHz, where humans emits at most. Also the length and the width of the ear duct are designed for this scope. Humans has a ear duct length of about 2.3-3.3 cm, that corresponds to a sound

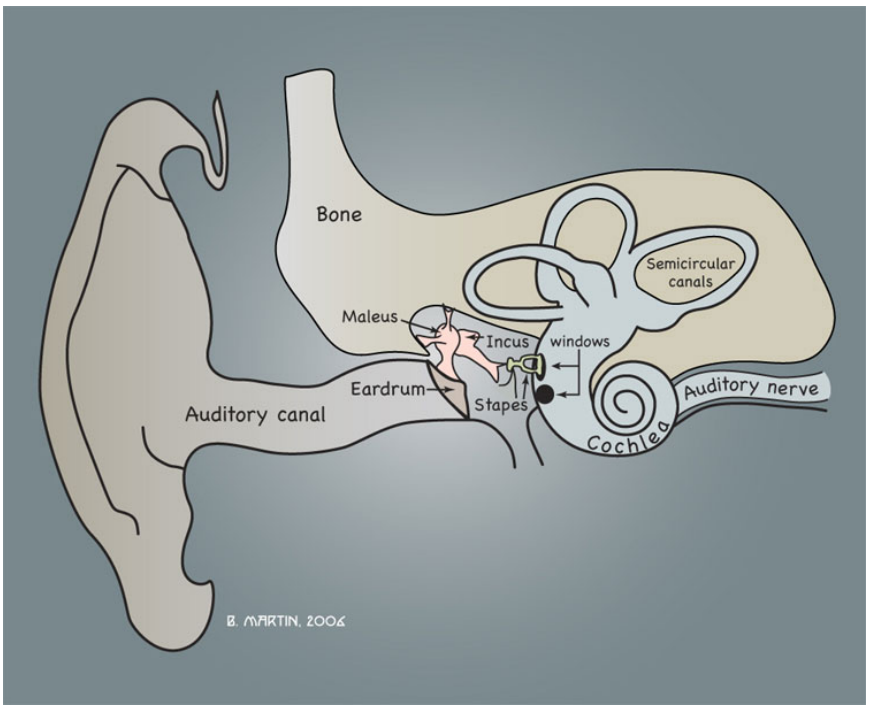

Figure 2.1: The ear

wave frequency of 3.8-2.6KHz. The whole system (pinna + ear duct) can achieve a gain of 10-20dB in the interval 1-7.5KHz.

- The *middle ear*, composed by the *ear drum*, the 3 *ear bones (malleus, incus and stapes)* and the *oval window*. This part acts as a transducer and a boost. The sound wave hit the ear drum, transmitting this pressure among the 3 bones that put in vibration the oval window. The information overcast as a pressure wave is transformed in a motion wave, with a gain up to 30dB.

- The *inner ear* is the last stage and its principal component is the *cochlea*, a snail shell shaped cavity whose length is about 3.5cm containing a fluid. Unrolling the cochlea, we can observe its structure: the cavity is divided by the *basilar membrane* in two parts, the *scala vestibuli* and the *scala timpani*. The width of the membrane (so also its rigidity) changes from the oval window to the top of the structure: it's narrow near the base and becomes wider at the end. Also the cavity changes its width: it's wider near the the oval window with respect to the apex. All the membrane is covered by thousands of hair cells. When the oval window receives the vibration, the liquid starts

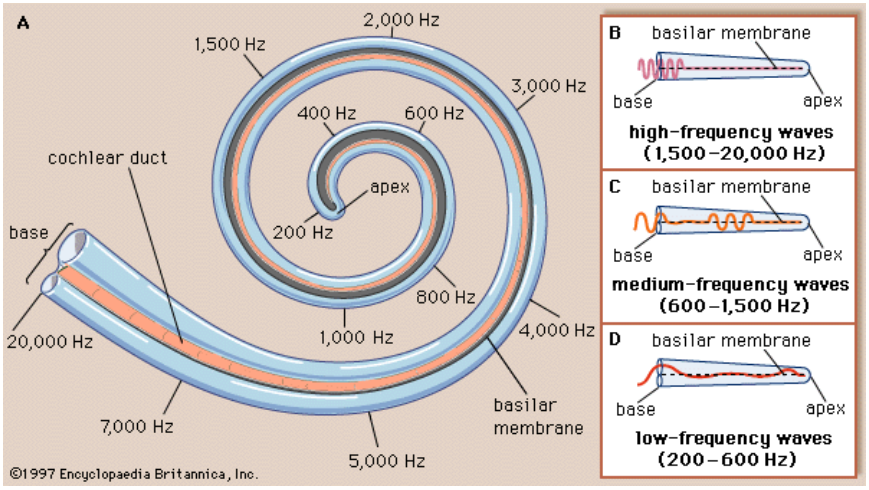

Figure 2.2: Cochlea

moving, moving the membrane that moves the ceils that broadcast this movement to the brain as an electric signal. Due to the different rigidity and width in every point, the cochlea acts as a filterbank: every point has a particular resonating frequency, so every cell is linked to a particular frequency content and can move only when receives a sufficient energy around its central frequency. The distribution of this frequencies is not linear nor normal: about two thirds of the length of the cochlea is devoted to low frequencies (from about 200 to 4000 Hz), having a quasi logarithmic distribution.

This complex structure explains why we must use logarithmic scale in music, we need to use many logarithmic unities of measure, we perceive some frequency component better than other ones and so on.

Robotics audition and active perception of sound source are a hard task. The common starting point of different existing algorithm is the knowledge of some psychoacoustics measurements. These main features are *ITD (Interaural Time Difference)* and *ILD (Interaural Level Difference)*, and they describe mathematically how humans can localize a sound source.
ITD measures the difference in time-of-arrival of a sound wave between the two ears (its unity of measure is ms). ILD measures the difference in sound pressure level (dBSPL) between the two ears.
ITD depends on the distance between the ears, the frequency content of the sound wave and the angle of incidence of the incoming sound. For geometric reasons, ITD gives useful measurements below the frequency threshold

of 1500-2000Hz. In some systems, it could be useful to make this analysis in the frequency domain instead of the time domain; the equivalent measure in this case is the *IPD (Interaural Phase Difference)*. As for ITD, ILD depends on the frequency content of the sound wave, on the incident angle and on physical structure of the human body. For these reasons, the most useful ILD measurements are at high frequency (many systems use the same threshold as ITD, 1500Hz).

2.1.1 Interaural Time/Level Difference (ITD/ILD)

We would like to build a model that can perceive and elaborate sounds and that can understand the origin of sound in a 3D world with only two sensors representing our ears.
The first approach that we can have is a simple model of our perception system: a spherical head with two identical ears put at the same height at opposite side each other. The presence of the head in the model is fundamental. For an incoming sound, the head is an obstacle; it can generate reflections, refractions, it's a real physical object put in the middle of the two ears. The sound wave can't propagate freely from the source to our sensors if there is an obstacle during its path. The model we built can describe the source position using two binaural cues called the *Interaural Time Difference (ITD)* and the *Interaural Level Difference (ILD)*.

ITD measures the difference in time-of-arrival of a sound wave between the two ears (its unity of measure is ms). It depends on the distance between the ears, the frequency content of the sound wave and the angle of incidence of the incoming sound. This *Time Difference Of Arrival (TDOA)* is due to the different path that the sound wave must do to reach the two different sensors. In our simple model, we can introduce some other approximations to obtain a closed formula, a rule that can be applied easily. We suppose that the sound source is placed far from the sensors and that the measurement is frequency independent. Far means that the distance is so that we can consider the incoming wave as a planar wave and frequency independent means that the measurements is still valid for each frequency tone we want to detect. With this assumptions, we can define ITD as

$$ITD = \Delta T = \frac{a}{c}(\theta + \sin\theta) \tag{2.1}$$

where c stands for sound velocity ($\sim$ 343m/s), a for the head radium, Θ for the incidence angle as you can see in Figure 2.3.

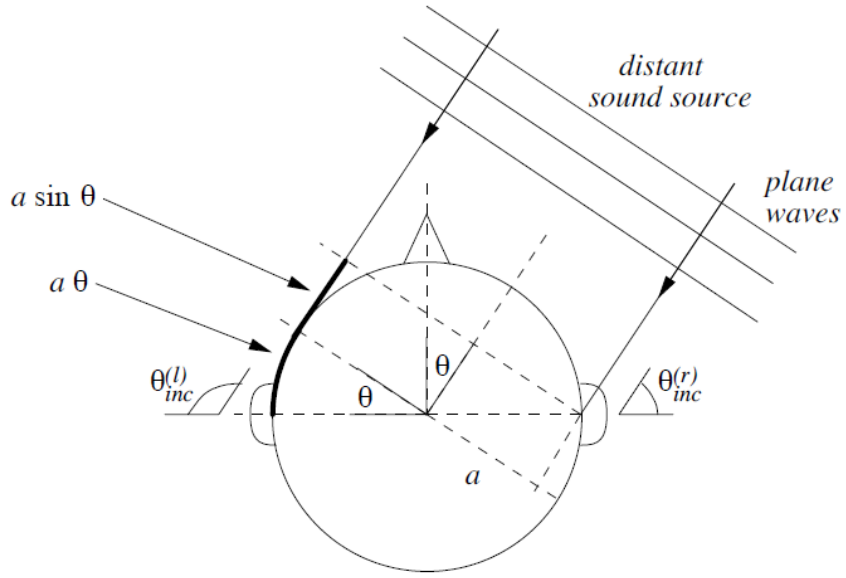

Figure 2.3: ITD

The result we can obtain with this assumption are not always valid. Given a, to observe some differences between the incoming signal at the left and at the right ear the wavelength of the sound must be bigger than the radium ($wavelength > a$). We can find out a frequency threshold from the condition above, that is $f = 1500Hz$. In other words:

- For $f > 1500Hz$, the ITD is not univocally defined
- For $f < 1500Hz$, the interaural difference is visible and identified with no possible confusion

There is also a lower bound for having valid result into this approximation, that depends on the minimum visible phase difference. This threshold depends on a and on the sampling frequency (the highest frequency we can use, the smallest is the timestep we have, the lowest phase difference is visible). See also [33] for a complete description of the problem.

ILD measures the difference in pressure level (dBSPL) between the two ears. As for ITD, ILD depends on the frequency content of the sound wave, on the incident angle and on physical structure of the human perceiving. Using the same model described above, we can observe this energy damping between the left and right signal because of the reflection and diffraction caused by the head. Due to the geometry of the head, this phenomena is

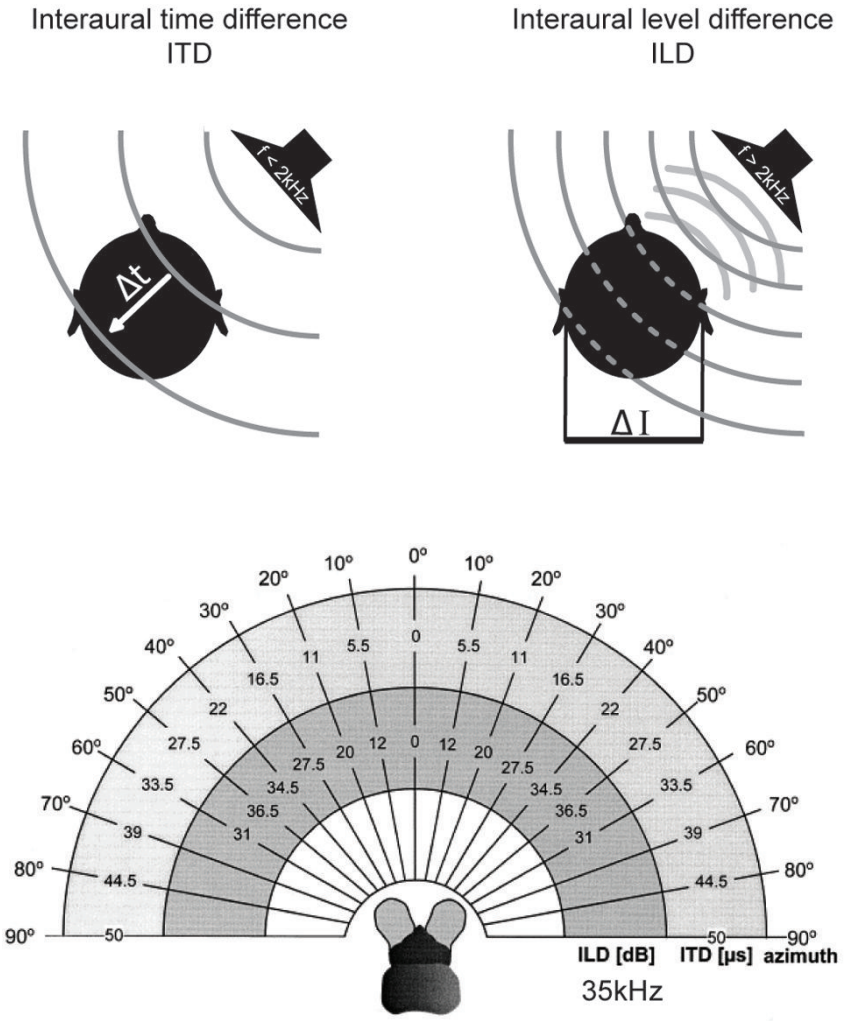

Figure 2.4: Some examples of ITD and ILD measurements

very sensible only to high frequencies, because at low frequencies correspond a long wavelength, so the head dimension couldn't be an obstacle for this type of wave. The frequency threshold for having ILD valid values is about $f = 1500 - 2000Hz$. Figure 2.4 represent ITD and ILD and also some useful numeric values obtainable with respect to the angle of the source position.

Maybe you noticed that you can choose identical thresholds for both phenomena. This task could be very useful: we can create a mixture feature vector composed by both the original cues after filtering out the not useful results. This process is validated by the *Duplex theory* that states that ITD and ILD are totally complementary parameters for azimuth perception. So

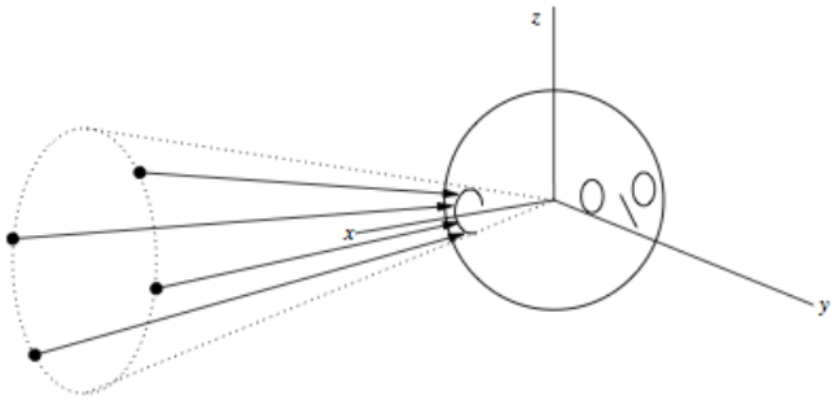

Figure 2.5: Representation of a cone of confusion

we can use:

- ILD for $f > 1500Hz$.
- ITD for $f < 1500Hz$.

Duplex theory works only in particular situations, when there is no reverberation in the room (so, in an anechoic chamber) and only in the frontal azimuthal semiplane and in a particular frequency interval. These limitations are due to the poverty of our model.

- Two sound coming from two opposite directions (from θ and from $\pi - \theta$) results in the same ITD and ILD. This phenomenon is called *Front-back confusion*.
- No anechoic condition are available in real environment. The precision of ITD can degrade very rapidly if the reverberant energy component is comparable with the direct energy of the sound source.
- As described above, there is also a threshold at low frequency, that can't allow to localize that frequencies.
- There is another type of ambiguity, called *Cones of confusion*. As in Figure 2.5, all the sound sources located at the same distance from the ear at different elevation angle could generate the same ITD and ILD, becoming not more distinguishable.

ITD and ILD are poor models, but this does not mean that they are totally useless. They are easy to be understood and calculated, are based

on some physical and acoustic tools and the preliminary hypothesis we need to state to use them correctly could be often relaxed and approximated as we needs.

When the aim of the system is to analyze the environment around or to go more in depth in the analysis of the sound source, ITD/ILD can not be enough. A more complex measurement used in this case is the *HRIR/HRTF (Head Related Impulse Response/Head Related Transfer Function)*. In few words, HRTF measurements give better and stable results, but can not be used for a general purpose system. HRTF strictly depends on the acquiring object and on human physiology. It's also computationally expensive to apply after acquisition.
This feature represents the acquiring object (for example the human who's listening) as a filter that depends on body geometry on the frequency content and on the source position on the space. The result achieved thanks to HRTF are more precise than using only ITD/ILD, but it requires a much more complex acquiring and processing stage, more expensive instruments and the result becomes directly linked to the actual situation in which the experiment is done and to the performer.

2.1.2 Head Related Transfer Function (HRTF)

The human perception model described above has many drawbacks. Studying more in depth the human physiology could be useful if we would like to obtain a richer model. Let's start considering how much every component of human body can interact with the incoming sound waves and how we can model it.

- *External and middle ear*: its form can be approximated as a tube of constant width and high acoustic impedance, so it's an acoustic resonator, whose pinna acts as a sound filter and reflector. In particular, its frequency response is directional dependent, so it can't be modeled as a simple filter. You can find out which frequencies are cancelled out from which source direction and create a *Pinna notch function* or you can take the system and make some measurements to retrieve experimentally the filter.

- *Torso and shoulder*: they introduce additional reflections and possible shadowing effect for sound incoming from low elevations and high frequencies. In this case, reflections are heavily elevation dependent: the same sound coming from the same azimuth angle but from different

elevation changes dramatically its perception. The torso introduces another distortion on the incoming signal, acting as a comb filter (in the frequency domain) with a strong attenuation for the high frequencies.

- *Head*: its real form is not completely spherical and symmetric and also the ears are not exactly at the same height and sited in opposite position.

All these physical considerations change dramatically the first simplest model we created to describe ITD and ILD. We need a new mathematical model, that must describe the interaction between the sound sources and the whole human perception system: it's called *Head Related Transfer Function (HRTF)*. As suggested by its name, we can project the perception system as a filter; more precisely, as a bank of filters, everyone describing one stage of the signal processing that we described before. The peculiarity of this model is that the HRTF is frequency dependent and also space dependent. To have a complete HRTF model of one humanoid acquiring system, you should describe it with a series of experiments, emitting each sound from each available position. This calculation can be made respecting the classical signal processing theory: taking a sound source and the signal acquired at a sensor, we can calculate the *Head Related Impulse Response (HRIR)* and then, through a Fourier Transform, obtain the HRTF of that specifical incoming signal emitted from that particular position with respect of that sensor.

There is another important difference between HRTF and ITD/ILD cues: given a couple of microphones, you have 1 interaural measurement but you need to calculate 2 different HRTF for each of the sensor. If in some papers you find an HRTF as a unique cue, maybe they are not properly using that name for describing a relation between the 2 real HRTF, or maybe they are approximating the HRTF with only a dummy head or a simpler model than the one we are describing, so be careful about the terminology used.

The calculation of an HRTF could be done in several ways, we focus our attention on two method.

- You can build an accurate physical model, that describe the physiology as above (i.e., you reproduce entirely a human body with a manikin) in an anechoic chamber. Then you put one microphone for each ear into the ear canal and a series of speaker (or 1 moving speaker) all around the acquiring object. Start the acquisition of a signal test (i.e.,

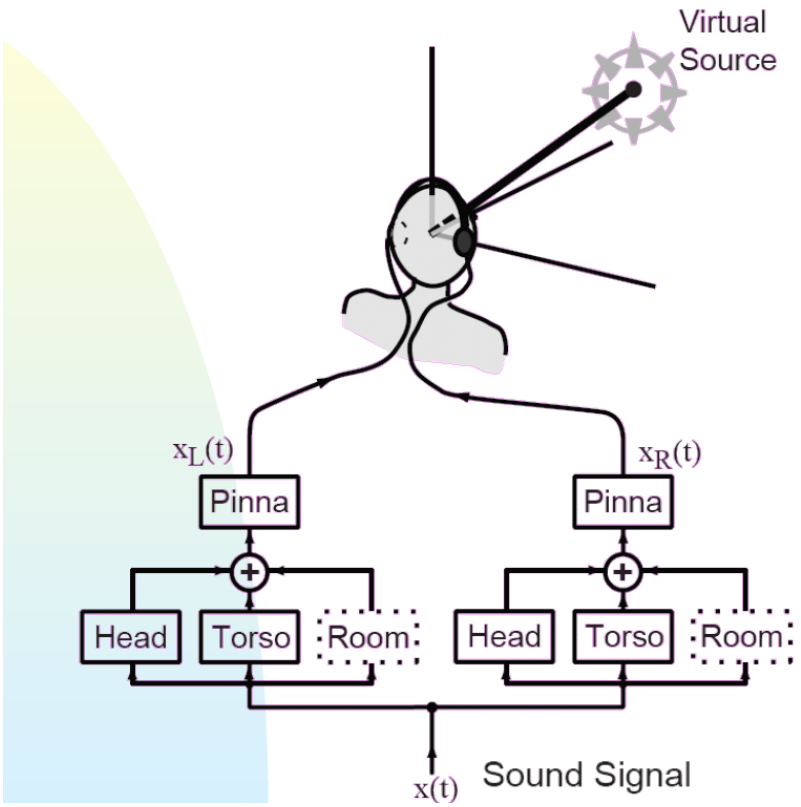

Figure 2.6: An example of synthetic HRTF

a white noise) for each microphone for each source position and save each HRIR.

- You can make a synthetic HRTF, creating a series of Linear Time-Invariant filter for each component of the system. The HRTF become a cascade of filters. For example, you can write a filter describing the head as an elliptic object with not-centered ears, another one describing the pinna notch filter, you can design the torso as a series of FIR comb filters or a delay line...

In Figure 2.6 you can visualize an example of synthetic HRTF structure.

For our primary task, that is a simpler and stable source localization algorithm for robotics with no willing of emulating human perception system, working with ILD/ITD could be sufficient. If we would like to go in depth with the analysis of human physiology, human perception or environment analysis, HRTF could be a better choice in future.

2.2 Algebraic tools

2.2.1 Gaussian Mixture Model (GMM)

A *Gaussian Mixture Model (GMM)* is a parametric probability density function represented as a weighted sum of Gaussian component densities. The GMM has the following form:

$$f(x) = \sum_{m=1}^{k} \alpha_m \phi(x; \mu_m, \Sigma_m) \tag{2.2}$$

α_m denotes the mixing proportion, the weight to assign to the $m-th$ Gaussian and $\sum_m \alpha_m = 1$. μ stands for the mean of the Gaussian density and Σ stands for the covariance matrix. We can view this tool as a kernel method in clustering problem: if we cannot describe how the parameters are jointly distributed, we can assume that each one has a Gaussian density with its own mean and covariance matrix. We can try to fit our model summing up all the single pieces that we have, often through an *Expectation-Maximization algorithm*. In other words, if I can do some preliminary hypothesis on the single parameters but I can not know how the joint distribution is, I mix up the original Gaussian distribution using a weight function, often unknown. The problem is reduced in complexity (studying the single relationship is easier than studying a joint one), shifting the problem on how to find out the correct α.

2.2.2 Expectation Maximization (EM)

An *Expectation Maximization (EM) algorithm* is an iterative method for finding maximum likelihood estimates of parameters in statistical models, The EM iteration alternates between performing an *expectation (E) step*, which creates a function for the expectation of the log-likelihood evaluated using the current estimate for the parameters, and a *maximization (M) step*, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter-estimates are then used to determine the distribution of the latent variables in the next E step. We define The idea is simple:

- Start with an initial guess of the parameters
- Calculate the expected value using the current value of the parameters
- Find out the new parameters that maximize the likelihood function

- Repeat the previous 2 steps until convergence (old parameters = new parameters)

2.2.3 Local Tangent Space Alignment (LTSA)

[12] and [15] used a strong algebraic framework to understand how to do a mapping between the feature vector they used and the sound source position. In their works, they demonstrate that the space of source positions has a cylindrical topology. To make reliable and stable the system, they adopted a low-dimensional space to achieve a linear topology for the front of the head and the rear independently, then they do a mixture of the two model to obtain the original cylindrical space. Instead, the incoming signal is composed by a high-dimensional vector (the interaural cues).

The *Local Tangent Space Alignment (LTSA)* is a *Manifold Learning technique*, a *Diffusion Kernel Clustering algorithm*, based on the idea that the input data are locally linear. It works in the following way:

- Input data lies on a high-dimensional space. LTSA builds a local neighborhood for each of this point using a *k-Nearest Neighbors algorithm (kNN).*

- Apply *Principal Component Analysis (PCA)* to each neighborhood to find out a low-dimensional representation of each data

- Build a local map computing by optimal alignment of these local representation. This optimization can be done in different ways (e.g., computing the largest eigenvalues, via EM, via Least Squares).

2.2.4 Probabilistic Piecewise Affine Mapping (PPAM)

The INRIA group states that there exist a locally linear bijection $g : X \to Y$, where X is the low-dimensional space of the sound source position and Y is the subset of the high-dimensional space of the binaural cues of the signal contained in X. The *Probabilistic Piecewise Affine Mapping (PPAM)* is based on the assumption that is possible to compute a *piecewise-affine probabilistic approximation* of g using the training dataset and them to estimate the inverse of g via a Bayesian framework. If we assume that g is locally linear, we can state that each point $y_n \in Y$ is the image of a point $x_n \in X$ trough an affine transformation plus an error term. So, we can try to find out the all possible affine transformation, assume that their number is finite (we call it K) and define the associated regions R_K. We can define:

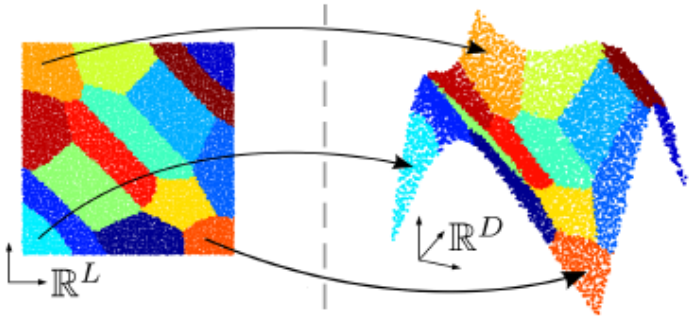

Figure 2.7: An example of PPAM

$$y_n = \sum_{k=1}^{K} I_{z_n=k} * (A_k x_n + b_k) + en \tag{2.3}$$

where $z_n \in (1 \dots K)$ and $z_n = k$ if the k-th affine transformation can map x_n in y_n; $I = 1$ if $z_n = k$ otherwise $I = 0$; A_k and b_k describe analytically the k-th transform; e_n is an error term distributed as a Gaussian variable with zero mean and and diagonal covariance matrix σ. So, we can state:

$$p(y_n \mid x_n, Z_n = k; \Theta) = N(y_n; A_k, x_n + b_k, \sigma) \tag{2.4}$$

where Θ represents the model parameters. After having defined all the prior probabilities, you can solve the equation above through a EM algorithm. The whole computation and the closed formula used are all described in the article. Solved the problem and found out the parameters Ω, you can do the mapping from x.

2.3 Binaural localization algorithms

In this section, we describe state-of-the-art speech localization algorithms (e.g., [2], [13], [26], [43]), classifying their different types of systems. Some studies [14] [23] are more focused on simulating the human auditory system, others [11], [45] aim to create an efficient algorithm focusing on the physics of sound waves and acoustics principles, others [1] [5] [7] ignore the physiological aspect and search for algebraic solutions that can better fit the data and can have results comparable with human capabilities. In this way we can understand where are the drawbacks, how to achieve the same goal in different ways, which features can be employed and why, which algebraic tools are more comfortable to use for audio input.

2.3.1 The baseline algorithm: GCC-PHAT

The *GCC-PHAT algorithm* is one of the most used localization model. Despite its age (the first article we found describing it is dated 1976 [27]), the simplicity and the robustness of this system makes it the starting point for every research in the field of sound source localization. As in [9], the main idea is easy: find out the direction of arrival of an incoming sound on two separated sensors by measuring the time delay between the two signals. This technique is called *Time-Delay Estimation (TDE)* and one of the most important mathematical tool used is *Generalized Cross-Correlation (GCC)*.

Given x_1 and x_2 as the signals at the two microphones, the GCC-TDE algorithm is based on the following equation:

$$R_{x_1x_2}(\tau) = \int_{-\infty}^{+\infty} W(\omega)X_1(\omega)X_2^{'}(\omega)exp(j\omega\tau)d\omega, \tag{2.5}$$

where τ represents the time-delay variable, W is a weighting function, X_1 and X_2 are the Fourier transform of the input signals. You could see $R_{x_1x_2}$ also as the Fourier transforms of the signal cross-spectrum. The time-delay we are searching for is the one which maximizes the GCC function. The localization algorithm is solved as a maximization problem. The choice of the weighting function is fundamental. There are several approaches described in [27] (e.g., the *Roth processor*, the *Smoothed Coherence Transform (SCOT)*, the *Eckhart filter*), but the one that gives the best results for sound source localization is the *Phase transform (PHAT)*. PHAT was developed ad hoc for this purpose:

$$W_{PHAT}(\omega) = |X_1(\omega)X_2^{'}(\omega)|^{-1}. \tag{2.6}$$

The GCC-PHAT algorithm works very well in ideal conditions (small rooms, no reverberation), but its results degrades if the Signal-to-Noise Ratio (SNR) arise. Its computational weight is light, so GCC-PHAT is still used nowadays in systems were the power consumption is limited and you need a rough estimation of the sound source. That is why in many state-of-the-art algorithm GCC-PHAT is used as a reference to demonstrate their precision, their reliability and their capacity to adapt to different environments.

2.3.2 The Deleforge et al. algorithms

The main idea of the 2D separation and localization algorithm presented in [14] is a probabilistic model, a *Gaussian Mixture model*, solved via *Expectation Maximization* using as cues *ITD* and *IPD*, plus a binary mask, used to

separate useful data from noisy components. In their work, Deleforge et al. use implicitly the HRTF jointly with ITD and IPD. The sound incoming is acquired using an active binaural robot, a system with a real human head shape that can move in horizontal and vertical planes (pan and tilt movements). The sound source is in a fixed point, while the robot can change its orientation, moving in pan and tilt. The utilization of a humanoid head makes implicit that they are filtering the input signal via the HRTF of this robot. This is the main process:

- The acquired signal is used to calculate 2 spectrogram, for left and right sound separately, through a FFT.
- These spectrograms are used to create ITD and IPD feature vectors. Due to some geometric constraints, called *W-disjoint orthogonality* [42], the authors can state that these features could be representative of the position of the source.
- To attenuate the noisy components (e.g., noise produced by the robot, reverberation), a binary mask is created. They decide a threshold, a minimum energy level for both ITD and IPD, to separate useful data from silence.
- ILD and IPD are assumed to be Gaussian distributed, building up a probabilistic framework. Estimating the source position becomes finding out the parameters of the Gaussian distributions and the angle that maximize a log-likelihood function. The problem is solved via a *Stochastic Expectation Maximization algorithm (SEM)*.

There are no strong assumption on the convergence of the algorithm, nor on the starting guess that one must choose to start the SEM, neither on the choice of the parameters used for the signal processing. Authors only demonstrate experimentally that their system works and gives better results with respect to other ones, but without any strong conclusion or physical and acoustic connections about why they obtain these results.

[12] describes another algorithm developed by Deleforge and his research group. they selected as audio cues *IPD* and *ILD*, but using a much more detail algebraic method called *Binaural manifold*, based on *Piecewise Affine Projection*. As in [14], they used a head shaped acquiring system that can move freely on horizontal and vertical planes. A non-linear dimensionality reduction technique is used to show that these data lie on a two-dimensional smooth manifold parameterized by the motor states of the listener, or equivalently, the sound source directions. Deleforge et al. propose a *Probabilistic Piecewise Affine Mapping model (PPAM)* specifically designed to deal

with high-dimensional data exhibiting an intrinsic piecewise linear structure. They derive a closed-form *Expectation-Maximization (EM)* procedure for estimating the model parameters, followed by Bayes inversion for obtaining the full posterior density function of a sound source direction.
Deleforge et al. extend this solution to deal with missing data and redundancy in real world spectrograms and hence for 2D localization of natural sound sources such as speech. They further generalize the model to the challenging case of multiple sound sources and propose a variational EM framework. The associated algorithm, referred to as variational EM for source separation and localization (VESSL) yields a Bayesian estimation of the 2D locations and time-frequency masks of all the sources. Deleforge et al. make freely available the entire Matlab source code and the training and the test set used in their experiment.

2.3.3 The Willert et al. algorithm

The goal of [45] is to estimate a position of a sound source in the frontal azimuthal half-plane, using binaural cues. *ITD* and *ILD* are measured separately through a *cochlear model*, then mapped together in a frequency vs time-delay world, called *Activity map*. The measurements are estimated via a probabilistic system. The learning stage is done acquiring white noise source in an anechoic chamber emitted at different azimuthal positions to get the conditional probabilities. These ones, with the binaural cues, are used to estimate the position of an unknown sound source.

Differently from [15], Willert et al. describe more in detail the choice they did, why they used some parameters instead of other, which physical limitation they found out. They aim at a sound localization algorithm that emulates the human auditory system as much as possible; so they make a deeper study on biology and psychoacoustics before choosing the localization features. This study describes much more in depth which are the links between mathematical parameters and physical cues, and this is very useful for our scope.

As in Deleforge research, Willert et al. acquire ITD and ILD from the two separated spectrograms, but, differently from [11] and [12], with a big attention on the biological model they analysed before. The binaural feature vectors are built as realizations of Gaussian distribution: for every frequency bin and time frame, they build a patch from the original left and right signals, they fix one of them as a reference and move the other one timestep by timestep. So, they obtain ITD and ILD feature vectors at each frequency and for each time difference between the two incoming signals called

Cochleogram. In the same way, they define a *Reference map*, collecting some training sounds in an anechoic chamber.

Willert et al. state the localization problem as a probabilistic one: the correct angle is the one that minimize the square difference between the cochleogram and the reference map at every frequency, time and timestep. Thanks to Bayes theorem, they simplify the probabilistic framework and obtain a closed form solution. They analyse also the case of a moving source, stating that their particular framework works well also in this case.

2.3.4 Other approaches

The systems we are presenting in this section represent a brief introduction to other state-of-the-art binaural localization algorithms. They are less interesting than the ones described above because they use complex psychoacoustic cues (i.e., HRTF in [23]) or algebraic tools (e.g., Particle Filtering in [32] or Neural Systems in [39]). We thought that for a robotic application this systems could be too much complicated. Nevertheless, they could contain some significant hypothesis and ideas that could be useful to study for our scope or for other future application in our field of research.

The algorithm presented in [23] by Keyrouz is focused on how to detect and understand sound events under severe acoustic conditions, using both monaural and binaural sensors. These measurements are obtained separately and then fused together through a *Bayesian network* to get the final feature vector that is checked with respect to a well-known sound source position lookup table to get the 3D direction of arrival. The training dataset and the cue used are composed by *HRIRs/HRTFs*, using also a semi-humanoid robot with pinnaes to better emulate its HRIR. The main mathematical tool used for getting the position is the *Cross Correlation*. The article contains also a detailed analysis on their results with respect to other existing approaches, like *PHAT* or *SCA (Source Cancellation Algorithm)*, and with respect to different conditions (i.e: high reverberant rooms, low SNR).
The Keyrouz algorithm has the advantage to work well also in high reverberant rooms, but at the cost of a specific humanoid physical model.

The aim of the system described in [32] by Lu et al. is to use binaural cues not only for detecting the direction of arrival, but also its distance without using HRTFs. The authors developed a system based on *ITD*, *probabilistic inference framework* and *particle filtering* to handle energetic measurements. They use a ratio between the direct energy and the reverberating one to understand the source distance. So the researchers can find the 3D position of a sound source with a particular mathematical tool, but

without having to use HRTF or other heavy human auditory models. Simply using ITD they achieve a great result.
Firstly, using *cross-correlation*, ITD and a lookup table of known audio signals, the authors do a first hypothesis on the azimuth; this knowledge is used in a *Equalization-Cancellation (EC) algorithm* to separate the direct sound from the reverberation and understand the source distance.
The EC algorithm is composed by a *delay-line model* for each frequency bin of the incoming signals. One signal is taken as reference, the other one passes through a delay-line. The power spectrum of the two signals are subtracted each other for each element of the delay-line, finding out all the directional energy components. Knowing the azimuth of the incoming sound, they can sum up all the other directional components defining the reverberant component of the sound and with them they define a Direct to Reverberant Ratio. The researchers demonstrate that this system works well both for simulated signals and real speech ones and it can achieve a good performance in excluding reverberation from the further processing.
This simple approach seems to work well, but only when the SNR is sufficiently high. If the direct sound has a lower energy than a given threshold with respect to the noisy component energy due to the reverberation, it is impossible to detect the angle of the useful signal. So the authors introduce a *Gaussian Mixture Model* to describe the relationship between azimuth and distance with respect to the Direct to Reverberant Ratio and run firstly an Expectation Maximization algorithm to find out the parameters of the Gaussian Mixture, then a Particle Filtering algorithm (PF) to obtain also suitable solutions for moving sources.

In [39], Nakashima proposes a robot with two acquiring sensors (two microphones) and with 2 humanoid pinnae to achieve both horizontal and vertical detection and localization. He uses as features *ITD* for the azimuth and *spectral features* for the elevation. In this way, Nakashima creates a physical model, a sort of simplified version of an HRTF, by creating a transfer function between the incoming signal and the "humanoid" robot sensors composed only by ears and a quasi-spherical main corpus. The training set is done mapping the minimum of this mixed feature vector (composed by ITD and spectral content) feeding the system with white noise emitted from every azimuth and elevation position through a *neural system*, a machine learning algorithm based on a biological model of a brain. The system is composed by a network in which different nodes are connected each other, exchanging messages between each other. The connections have numeric weights that can be tuned based on experience, making neural nets adaptive to inputs and capable of learning.

2.4 A note on binaural speech localization algorithms

All the systems we described above are based on different algorithms and different binaural cues. We decided to focus on the papers written by Deleforge et al. and Willert et al. because they seems for us the most effective despite they used different approaches.

The Deleforge algorithm is based on strong assumptions and on a deep algebraic tool, thanks to that Deleforgeet al. obtained very well results. This task represents also the algorithm drawback: it seems that the speech localization system works by mapping the impulse response of the environment in which the sound source is emitted. The core of the Deleforge algorithm can not separate between the original speech signal, the room response and the acquiring system response; the PPAM tool used is acting also on the impulse response of the room. The aim of the next chapters is also to understand how much the Deleforge algorithm can be improved, how much its result are related to the processing stage and how much they are linked to the room frequency response.

The Willert algorithm is near to our scope. It is focusing on the direction of the speech source signal without any knowledge about the environment. The cues used are directly linked to the original sound source; no processing on the room is described in the algorithm because it is not in its scope. The system acts only on the audio data, it calculates the relationship between the sound source and the direction of arrival. To do this, Willert et al. thought to a biological model, that is over of our scope. We are not interested in localizing a speech source direction by emulating the human auditory system; we simply would like to make a robot able to recognize and understand the direction of arrival of a sound. This is the reason why we investigated this algorithm, understanding how this algorithm works, but remembering our main target.

Chapter 3

Algorithm baseline analysis

One of the goals of this thesis is to understand how the most effective sound localization algorithms work, which tools are used and which types of parameters are needed. In this chapter, we do a deeper analysis of some of the systems described in Chapter 2, the ones that seem to us more efficient, with the best result and the most impressive structure. We focus our attention on [12], [23] and [32], [43], [45]. We choose these papers because they achieve good results, each one in its own completely different way. This analysis can tell us important information about the processing we can do on input data, the features we can develop and use in our framework, how many parameters can be tuned and which type of result we can find out changing them.

3.1 A 2D localization algorithm based on manifold learning

The PERCEPTION group of the INRIA (the French National Institute for computer science and applied mathematics) has worked for years on speech recognition, segregation and localization algorithms, becoming one of the most important research institute in this filed. Their works (e.g., [2], [12], [13], [14]) contributed to the European project HUMAVIPS, whose aim was to create a robot that could interact with humans. We chose to study their works and, in particular, we focus our attention on [12] and [15] written by Deleforge et al. and on the algorithm they developed in the articles.

3.1.1 Physical and acoustic parameters of the algorithm

The aim of the research is to obtain a 2D localization and separation of audio sources using *ILD* and *IPD* taken by the audio signals recorded from 2 microphones put into a dummy head that can be rotated on the horizontal and on the vertical axes. They choose these two cues because they are independent from the sound source content and they contains spatial information. To ensure a complete knowledge on the spatial cues, Deleforge et al. decide to work in the time-frequency domain, using a *Short Time Fourier Transform (STFT) analysis.*
The spectrograms of the incoming signals are passed through a 64ms-length window, with 8ms hop-size, then are transformed using the FFT algorithm with a sampling frequency of 16000Hz in [12] and [15] and 24000Hz in [14]. With these parameters, each time window contains 1024 or 1536 samples, with 512 or 1024 frequency bins into the frequency interval of 0-8KHz or 0-12KHz.

Taking one sound source in a fixed position at a certain distance from the two microphones, for each time-step and frequency bin we can define an *Emitted Acoustic Level* $a_{f,t}$ and a *Perceived Acoustic Level* $a_{f,t}^{L,R}$ as:

$$a_{f,t} = 10\log([S_{f,t}]^2), \tag{3.1}$$

$$a_{f,t}^{L,R} = 10\log([S_{f,t}^{L}]^2 + [S_{f,t}^{R}]^2), \tag{3.2}$$

where $S_{f,t}$ stands for the emitted spectrogram and $S_{f,t}^{L,R}$ for the perceived spectrogram at left/right microphone.

The linkage between emitted and perceived sounds can be described with the *HRTF*:

$$S_{f,t}^{L,R} = h^{L,R} * S_{f,t}, \tag{3.3}$$

where $h^{L,R}$ denotes the left/right HRTFs.

Let's define the *Interaural Transfer Function (ITF)* in the following way:

$$ITF = \frac{h^R}{h^L}. \tag{3.4}$$

You can notice that ITF depends from the source position because HRTFs, by definition, depend on it. The *Interaural spectrogram* is then defined as:

$$\hat{I_{f,t}} = \frac{S_{f,t}^{R}}{S_{f,t}^{L}}, \tag{3.5}$$

where, from Eq. 3.3, $\hat{I_f} = ITF$.

The distance between the sound source and the microphone can be considered as far away from the dummy head. This hypothesis makes valid the *W-disjoint orthogonality* [42], which states that in this condition the $HRTFs$ mainly depends on the sound source position instead of the distance; so we can say that $\hat{I_f}$ depends only from the source position.

Finally, we can define the *ILD* and *IPD* as:

$$\alpha_{f,t} = 20 * \log [\hat{I_f}], \tag{3.6}$$

$$\phi_{f,t} = \exp(j \arg(\hat{I_f})). \tag{3.7}$$

That is, we get two binaural measurements that are representative function of the position of the incoming signal.

These cues can be used to calculate temporal means. The final inputs of the localization algorithm are 2 vectors: the *interaural mean vectors* associated to the source position. They are called $\bar{\alpha}$ and $\bar{\phi}$ and are defined as:

$$\bar{\alpha_f} = \frac{1}{T} \sum_{t=1}^{T} \alpha_{f,t}, \tag{3.8}$$

$$\bar{\phi_f} = \arg[\frac{1}{T} \sum_{t=1}^{T} \exp(j\phi_{f,t})]. \tag{3.9}$$

Finally, Deleforge et al. do sound source localization using the temporal mean of the binaural cues for each frequency bin. So, $\bar{\alpha_f} \in \Re^F$ and $\bar{\phi_f} \in \Re^{2F}$, where F is the number of frequency bins available.

3.1.2 Data validation

The Deleforge group operates a further processing before feeding the mean interaural vectors to the localization algorithm. Indeed, the mapping between the emitted spectrograms and the binaural cues are valid only if there is a valid signal in each (f, t) frame, i.e., if there is emission ($S_{f,t} \neq 0$). If the input signal is a white noise, by definition, we can observe a power spectrum spread over all the frequency bins, but real sounds (e.g., speech sound, music) are sparse signals. They could have missing values and, in those frames, all the previous definitions are not more valid. So, only useful data are further processed and the useless signals are filtered out. To do so, the authors create a *binary mask* called ξ, defining $\xi = 1$ if the input

power spectrum is over a given threshold, unless $\xi = 0$. This is a simple, but efficient way to filter out the noisy component of the incoming signal (e.g., room silence, late reverberations). This mask is applied over the binaural cues α and ϕ, so to put to 0 the noisy component.

Another preprocessing Deleforge et al. do is based on the *Duplex theory* [33]. The cues used are not valid for each frequency bin. This theory states that ILD values are useful only at high frequencies $f > 2KHz$, while IPD is valid only in a range of mid-low frequencies $300Hz < f < 2KHz$. So, before calculating the mean interaural vectors, α and ϕ are filtered as the rules described above, getting a *mean high/low-ILD vector* and a *mean high/low-IPD vector*.

The mean binaural vectors live in a high-dimensional space (remember that $\bar{\alpha} \in \Re^F$ and $\bar{\phi} \in \Re^{2F}$, where F is the number of frequency bins available), but the goal of the algorithm is to find out a relationship between these cues and the 2D world space. The subspace selected is a cylinder, because a cylindric coordinate system can describe the position of a sound source around the receiver completely. To create the mapping between the two spaces, the authors analyze a *manifold structure*, a non-linear 2D subspace, in which the binaural vectors can lie. The manifold learning technique they choose is a *Local Tangent Space Alignment (LTSA)*. You can find a description of this clustering algorithm in Chapter 2.

Applying LTSA on the four available binaural subsets, they find out that mean low-ILD, mean low-IPD and mean high-ILD are homeomorphic to a cylinder, instead mean high-IPD shows several distortions. This observation, in part, confirms the duplex theory, but introduces an important difference: it seems that also low-ILD cues can be mapped in a 2D world, supposing that also this cue can be representative for the source localization of the sound that it represents. This experiment leads to the definitive decision on how to use binaural cues: the mean high-IPD vector is discarded, while all the remaining cues are grouped together to create a unique big feature vector called *ILPD spectrogram*. According to the initial hypothesis done in [12], the ILPD vector is a vector of dimension 730. Surprisingly, in [15] all the above preprocessing stage is ignored, and the ILPD vector created is composed by all the binaural cues component, so its dimension is $3F = 1536$.

3.1.3 The core of the algorithm

The goal of the algorithm is to find out a mapping between the incoming sound, represented via the ILPD vector, and the 2D coordinates of the sound source. To do this, Deleforge et al. used a *Probabilistic Piecewise Affine*

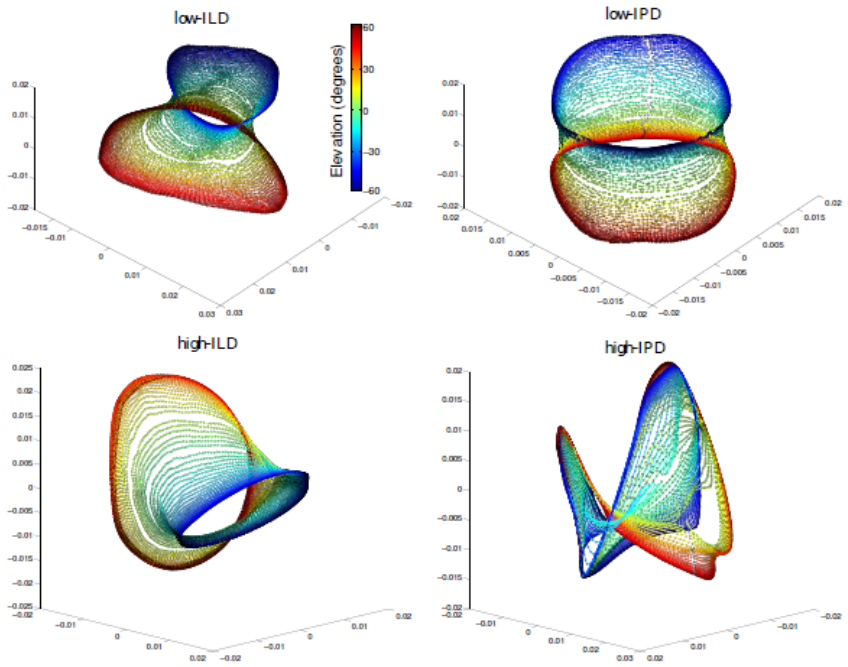

Figure 3.1: Representation of binaural manifolds using LTSA

Mapping (PPAM) described in details in Chapter 2. The system works in the following way:

- The system needs a training stage. A dataset of well-known training sounds and source direction is built from white noise signals emitted by a speaker. The distance between source and listener is bigger than the distance between the two microphones, such that we can assume that sound wave propagates as a free-field and the W-disjoint orthogonality still holds. In [15] it is called *loudspeaker-WN dataset*, and it is composed by 432 sounds at different azimuth and elevation angles. To create the position ground-truth, they used a camera placed on the top of the dummy head used in the experiment. The dummy head is put in the center of a reverberating room in a fixed position and the speaker is moved manually, covering all the positions available in the field of view of the camera. The position is then retrieved via a visual detection and localization algorithm as the pixel position of the camera.

- Every signal in the training dataset is passed as input to the PPAM algorithm. More precisely, the authors do an inverse regression from the low-dimensional space to the high-dimensional space, then they

Method/cues	Azimuth err. (°)	Elevation err. (°)
PHAT	2.80 ± 2.25	not available
ILD	1.20 ± 0.99	1.09 ± 1.45
IPD	1.05 ± 0.90	1.46 ± 1.70
ILPD	$\mathbf{0.96 \pm 0.73}$	$\mathbf{1.07 \pm 0.92}$

Table 3.1: Mean localization error in azimuth and elevation angles of the Deleforge et al. algorithm using different binaural cues (ILD, IPD and ILPD) compared with respect to the PHAT algorithm.

invert the mapping, finding out the most suitable approximation of the requested affine transformation. All the retrieved parameters are then saved in a structured variable called Θ.

- Given a new incoming sound, the system downloads Θ, builds up the affine transformation, calculates the binaural vector and the PPAM. The result is the supposed position of the sound source.

The original Matlab code and all the dataset are freely available on the web site of the INRIA group and are reported on Appendix A.

3.1.4 Results and validity of the algorithm

To test the validity of the method, the INRIA group tested the algorithm in different situations. They build up other two dataset in a similar way than the loudspeaker-WN dataset, called *loudspeaker-TIMIT dataset* and the *Person-live dataset*. The loudspeaker-TIMIT dataset is composed by 108 emissions from a speaker whose position ground-truth is known. The sounds emitted are human speech signals in English, of different length, taken by the TIMIT dataset, a bigger dataset publicly available for different kind of speech processing. The Person-live dataset is composed by real people talking in front of the dummy head whose ground-truth directions are saved using a face recognition algorithm with the camera put on the head.

Deleforge et al. tested different scenarios, comparing different binaural cues, but using always the same training test and rooms with similar dimensions. The result obtained are then compared with respect to other state-of-the-art (e.g., Linear regression (MPLR), Sliced-Inverse Regression (SIR) and GCC-PHAT) to demonstrate the effectiveness of their algorithm. Tables 3.1 and 3.2 show the results:

Table 3.1 demonstrates how much the choice of the binaural cue can affect the final source localization resolution. It seems that the more complete

Method	ILD		IPD		ILPD	
	Az. err. (°)	El. err. (°)	Az. err. (°)	El. err. (°)	Az. err. (°)	El. err. (°)
PPAM	$\mathbf{2.2 \pm 1.9}$	$\mathbf{1.6 \pm 1.6}$	$\mathbf{1.5 \pm 1.3}$	$\mathbf{1.5 \pm 1.4}$	$\mathbf{1.8 \pm 1.6}$	$\mathbf{1.6 \pm 1.5}$
MPLR	2.4 ± 2.2	2.2 ± 2.1	1.8 ± 1.7	1.9 ± 1.7	2.2 ± 2.9	2.1 ± 2.0
SIR	41 ± 34	17 ± 13	36 ± 25	24 ± 17	41 ± 34	16 ± 13

Table 3.2: Comparison between mean localization error in azimuth and elevation angles for different binaural localization algorithm (Piecewise Affine Mapping used by Deleforge et al., Multiple Linear Regression and Sliced Inverse Regression) with respect to different binaural cues (ILD, IPD and ILPD).

and long is the feature vector, the more precise could be the algorithm. As we described in 2, this result shows that the affine transformation used in the INRIA algorithm allows the system to find out also the elevation of the speech source; the PHAT algorithm can not achieve this task.
Table 3.2 shows how much the localization method proposed by Deleforge et al. improves its performance with respect to other state-of-the-art algorithms. We could state that a speech source localization algorithm must use an algebraic tool to cluster the data and to obtain useful results. A simple linear regressor fails, the system needs at least a multiple linear regressor or a more complex mapping, as the PPAM Deleforge et al. proposed.

3.2 A biologically-inspired sound localization system

In [45], the Control Theory and Robotics Lab of the Darmstadt University of Technology, in collaboration with the Honda Research Institute, proposes an algorithm that can estimate the position of a sound source in the frontal azimuthal half-plane. The main cues used are binaural cues like *ITD* and *ILD* measured thanks to a *cochlear model* that emulates the human auditory system. The two feature vectors are measured separately each other, then they are mapped together in an *activity map*, a frequency vs time-delay representation of the binaural cues. To estimate the position, they use a *probabilistic framework* based a *learning stage* in which different training sound are emitted by different known azimuthal position to get *time-dependent conditional probabilities*. These probabilities combined with ITD and ILD allow to estimate the source position.

3.2.1 Physical and acoustical parameters of the algorithm

One of the aim of this article is to find a way to model the human auditory system, so they choose as cues *ITD* and *ILD*, measured via a *cochlear model*.

As described in Chapter 2 , ITD measures the time difference Δt between the two incoming signal due to the distance between the two ears Δs. As first approximation you can think that the time shift between the two signal is only due to this distance. This leads to a poor model, because it does not account for reflections and refractions of the whole human body and for other phenomena like head-shadow effect. Furthermore, the time-delay is not linear with respect to the incidence angle α. The measurement must depends also on the frequency content of the incoming sound, the source direction and the human physiology. To take care about all this, the authors describe a non linear dependence between Δs and Δt:

$$\Delta t = \frac{d \sin(\alpha)}{v}, \tag{3.10}$$

where v stands for acoustic velocity. This formulation is valid only below the frequency threshold of $f = 1500Hz$.

ILD measures the energy level difference between the two ears (see Chapter 2). It's strongly dependent on the shadow effect, the loss of energy of high frequency components due to the geometry of the human body. Head and shoulders create reflection and diffraction phenomena that hit mainly the high frequencies. Willert et al. observe that this measurement is frequency dependent (it is strongly non linear) and depends also on α. All this observations confirm the *Duplex theory*: ITD and ILD are complementary features that can describe the human auditory system at different frequencies: for $f > 1500Hz$ the authors used ILD, otherwise ITD.

The next step of the localization algorithm is the creation of a biological model. Willert et al. focus their attention on the *Cochlear model*, based on the Patterson formulation. They describe the cochlea as a filterbank, based on *ERB-filters*, equivalent rectangular bandpass filters whose bandwidths changes for each characteristic frequency respecting the law $f_c = \omega_c/2pi$. Using some predetermined parameters, the authors can compute f_c and the bandwidth, leading to the final filterbank as in Figure 3.2. Notice that the filters are overlapping each other and the characteristic frequencies are logarithmic distributed.

Willert et al. notice also that this type of filtering produces phase distortion, so the frequency information at a timestep does not correspond to a single timestep of the input signal. They need to introduce a phase-compensated technique to obtain a useful cochleagram. The authors decide to do a forward-backward filtering, simply filtering first forward than backward the signal to obtain zero-phase distortion and also doubling the filter order. The final signal obtained after the cochlear filter is called *cochlea-*

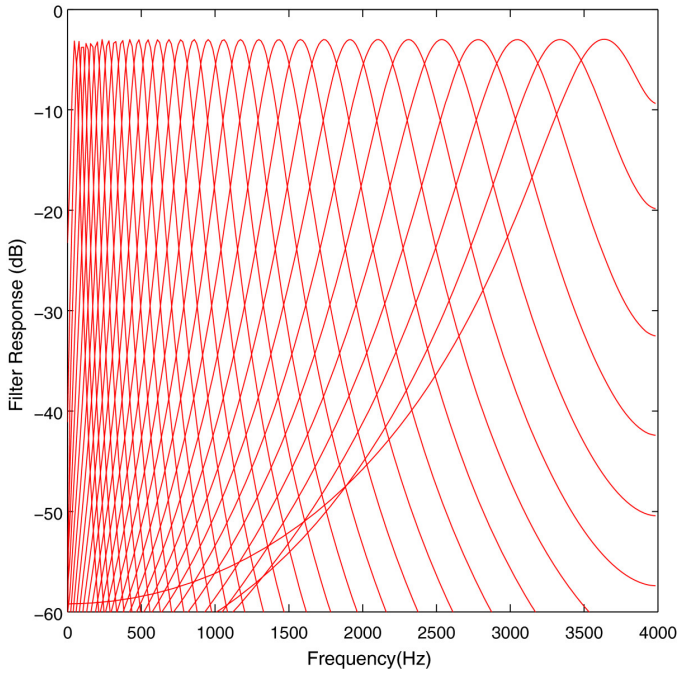

Figure 3.2: The ERB-filter used in the Willert algorithm

gram. It is a frequency vs time measurement and its length depends on the length of the input signal and on the sampling frequency used.

Also Willert algorithm assumes that W-disjoint orthogonality holds, so for each frequency and time there is only one ITD and ILD valid measurement. In other words, each frequency and time is uniquely determined by ITD and ILD.

The main idea to measure ITD/ILD is to compare left and right patches taken from the cochleagrams at each time step. A patch is defined as a restriction of the complete signal whose dimension is smaller both for frequency and time than the original signal. The searched time difference/energy level is the one thanks to that the two patches corresponds. Giving L and R the input cochleagram, defining $L^{f,t}$ the left patch at frequency f and timestep t, $R^{f,t+\Delta t}$ the right patch at frequency f and timestep $t+\Delta t$, W a gaussian window of the same dimension of the patches used to weight the values in every patches, the problem is stated as:

$$\Delta t = \arg\max_{f,t}[W. * R^{f,t+\Delta t} = W. * L^{f,t}]. \tag{3.11}$$

To solve this equation, they create a *generative probabilistic model*, introducing also a jittering 0-mean Gaussian-distributed noise μ and a model accounting for possible mean shift and level difference of the amplitude of

signal frequencies, described by parameters λ and k.

$$W. * R^{f,t+\Delta t} = \lambda W. * L^{f,t} + kW + \mu. \tag{3.12}$$

Solving Eq 3.12 with respect to μ, they get:

$$\mu = W. * R^{f,t+\Delta t} + \Delta t - \lambda W. * L^{f,t} - kW = A. \tag{3.13}$$

The authors can state that also A is Gaussian distributed and they can define the *ITD Activity map* $C^T_{ITD}(f, \Delta t)$ as:

$$C^t_{ITD}(f, \Delta t) = \frac{1}{\sigma_\mu \sqrt{2\pi}} \exp(-\frac{1}{2\sigma^2_\nu}||A||^2). \tag{3.14}$$

Willert would like a ITD measurement independent from the noise components put in the model. So, they define an estimate of C as:

$$\widetilde{C^t_{ITD}}(f, \Delta t) = \frac{1}{\sigma_\mu \sqrt{2\pi}} \exp(-\frac{\sigma^2_R}{2\sigma^2_\mu}(1 - \rho^2_{L,R})) \tag{3.15}$$

where σ_R is the variance of the patch R and $\rho_{L,R}$ is the cross-correlation between R and L.

In a similar way, the authors define the *ILD Activity map* as:

$$\widetilde{C^t_{ILD}}(f, \Delta t) = 20 \log(\frac{\sigma_L}{\sigma_R}). \tag{3.16}$$

3.2.2 The core of the algorithm

Also this algorithm requires a learning stage, done in a supervised way. Some training sounds are acquired in an anechoic chamber using 2 cardioid microphones at a distance of 14cm from each other and separated by an isolating sphere. The Analog-to-Digital converter works at 44.1KHz and 16 bit. Many kind of sound are used (e.g., white noise, speech signals) emitted by a speaker placed in 13 different positions in the half-plane in front of the experimental setup at 2m. They define the *Reference maps* as:

$$I_c(\alpha, f, \Delta t) = \frac{1}{t_\alpha} \sum_{t_\alpha} C^{t_\alpha}_c(f, \Delta t), \tag{3.17}$$

where $c \in (ITD, ILD)$, α is the incident angle, t_α is the number of timesteps used for the measurement in that position. In other words, the Reference map taken in a position is the average of the related Activity map with respect to the number of measurement done.

The above definition is fundamental to build up the *likelihood formulation* for the sound source estimation:

- The incoming sound is processed as described above to obtain its Activity maps.
- The Activity maps are compared with the Reference maps to obtain the following probabilistic distribution called *Likelihood maps*

$$P(z^t|\alpha, f, c) = \frac{1}{\sigma_p\sqrt{2\pi}} \exp(-\frac{1}{2\sigma_p^2} \sum_{\Delta t}(C_c(f, \Delta t) - I_c(\alpha, f, \Delta t))^2) \tag{3.18}$$

  where σ_p is the standard deviation of the Gaussian distribution that they chosen as $1/2$ the sum of the square differences between all the C and I from the experiment. It describes the probability that the measurement z^t has happened at timestep t given α, f, and c.

- Willert would like to achieve a measurement system that is frequency and cue independent, so they demonstrate how to get $P(z^t|\alpha)$. Applying marginalization from the Eq3.18, they can write:

$$P(z^t|\alpha) = \sum_{f,c} P(z^t|\alpha, f, c)P(f|\alpha, c)P(c|\alpha). \tag{3.19}$$

  Applying the *Duplex theory* and due to the preliminary hypothesis done in the section before, they can state that:

  - $P(ITD|\alpha)$ is a valid measurement only for small values of α
  - $P(ILD|\alpha)$ is a valid measurement only for large values of α
  - $P(f|\alpha, ITD)$ is a valid measurement only for low frequencies
  - $P(f|\alpha, ILD)$ is a valid measurement only for high frequencies

  This measurements are called *Knowledge maps (K_c)* and they contain the values of conditional probability distributions $P(f|\alpha, c)$ and $P(c|\alpha)$ that could be pre-determined, estimated or put as equally distributed.

- Estimate the angle means search for the distribution $P(\alpha|z^t) = P(z^t|\alpha)P(\alpha)$, as stated by the Bayes theorem. So, the whole localization problem can be solved as a *Maximum a Posteriori estimation algorithm*:

$$\bar{\alpha_t} = \arg\max_{\alpha} P(\alpha|z^t). \tag{3.20}$$

  Notice that this equality holds if you know $P(\alpha)$. But $P(\alpha)$ describes the relationship between the localization system and the environment around it. So the authors need in some way to do some prior assumption on the environment you are working in.

- To avoid this problem, Willert et al. can approximate $P(\alpha)$ with the previous knowledge they get from the previous estimate. They state that:

$$P(\alpha) = P(\alpha|z^{t-\Delta t}). \tag{3.21}$$

The momentary prior is simply the previous posterior probability distribution. Equation 3.21 can be extremely useful also to extend the localization algorithm to a moving source. In fact, we can define

$$P(\alpha) = \sum_{\alpha'} P(\alpha|\alpha')P(\alpha'|z^{t-\Delta t}) \tag{3.22}$$

where $P(\alpha|\alpha')$ describes as a Gaussian distribution how much the position change with respect to the previous position

Resuming the algorithm:

1. Calculation of the left and the right cochleagram L, R
2. L, R are used for the calculation of the ITD and ILD activity maps C_{ITD}, C_{ILD}
3. C_{ITD}, C_{ILD} are used for the calculation of the likelihood maps P_{ITD}, P_{ILD} using learned reference maps I_{ITD}, I_{ILD} and knowledge maps K_{ITD}, K_{ILD}
4. P_{ITD}, P_{ILD} are used for the calculation of one posterior probability distribution that is propagated over the time to estimate the most probable sound source direction $\bar{\alpha}_t$

3.2.3 Results and validity of the algorithm

In the paper, the authors describe also some results obtained both in an anechoic chamber and in a reverberant room. The test set is composed by 13 classes of sounds, each one containing 38 speech signal whose length is 47 time frame. Each test sound is put at 3 different distances from the acquiring system.

In the anechoic chamber the accuracy reaches the 87% using only ITD as cue, the 89% using only ILD, the 99% using both cues. The comparison is done with the resolution of the training set that is 15°. In the reverberant room the algorithm is less precise: it reaches the 62% using ITD, the 33% using only ILD, the 69% using both, always with a resolution of 15°. The accuracy value is calculated by the ratio between the number of speech sound source correctly located with respect to the training resolution and the total amount of test sound used for the experiment.

3.3 Analysis and assessment of the baseline

3.3.1 Deleforge algorithm

One of the goals of our research is to understand the main features of the existing localization algorithm, focusing on a robot able to localize one speech signal in a realistic environment.

The result obtained by the INRIA group (see Tables 3.1 and 3.2 were very interesting: they achieved a resolution in azimuth and elevation angles that in some cases outperformed the human auditory system, with some strong algebraic analysis but using only binaural cues based on simple psychoacoustic models. They stressed the mathematical analysis, obtaining an efficient, but really hard to understand algorithm, meanwhile we think that the preprocessing analysis on the audio signal was poor. We found out some critical points investigated in the next sections of the thesis:

- Firstly, they do not make any restrictive assumption on the incoming data. Their aim is to localize a human talking, but they do not use any kind of filter to focus the analysis only on the frequency intervals linked to the speech signal. In other words, in their articles they do some preliminary hypothesis (e.g., sampling frequency, length and type of windows, length of FFT, length of interaural cues), but without any demonstration nor justification. It seems that the algorithm works well only because of the maths they introduce; it is more an algebraic exercise than a psychoacoustic system.

- Also the acquiring system chosen is fundamental. No analysis on how much the system is linked to this choice is done. We do not know if the same algorithm can be applied on another machine with other sensors put at different positions.

- Another critical point for us is that they do not say anything about the interaction between the sound source and the room. There is no deep analysis on reverberation, they only do the experiment in few small rooms with similar characteristics. Nobody can tells us that this algorithm could be feasible also in noisy and reverberant rooms.

- The last problem we would like to focus on is the training stage of the algorithm. To find out the source position we need to check the feature vectors with a training set, so it seems impossible to export this algorithm on a real machine, because we can not have hundreds of sounds available for each rooms we will enter. How much the system is

linked to this training? Is it possible to obtain useful result also with a lighter training?

We would like to investigate how much powerful is this system, testing it in MATLAB with other training and test sound, changing all the parameters available, introducing a preprocessing stage, checking different filters. We want to really understand how much the algorithm can be improved for our purpose and how much its strength is due to the PPAM algorithm and how much is due to the acoustic parameters.

3.3.2 Willert algorithm

With respect to [12], this algorithm has poor results. The accuracy and the resolution achievable by the Willert algorithm are much lower than the Deleforge systems. Another weak point of the Willert algorithm is that the model used is too much linked to the human auditory system for our scope. The great part of the Willert paper is dedicated to the creation of a physiologically accurate model of the auditory system. It seems that the aim of authors is to achieve the localization task by emulating the human system. The aim of the algorithm we would like to create is to localize a speech signal in a real situation, without being focused on the model used. Our primary goal is the localization task instead of the creation of a system that must emulate human physiology.

But we found out some strong points in this system, that could be the starting point for our future proposals:

- Despite the complex biological model, in which we are not too much interested in, the math used is very efficient and simple. It is easy to understand and not computationally heavy. The choice of the likelihood framework is a sort of Gaussian Mixture Model, that is a standard and well-known probabilistic tool.

- The training set is done in an anechoic chamber, but is very fast. The algorithm does not require hundreds of sounds emitted by a great number of positions with a high resolution, so the preprocessing required for this part of the system is lighter than in [15].

- Willert demonstrates also the validity of his framework both in anechoic and in real reverberant rooms without changing the training set. The train is done once, then it seems valid in every environment. This could be useful if we want to create a feasible localization system that must obtain some result in any situation.

- As stated in 2, the strength of the Willert et al. algorithm is in the filtering stage. The authors are not interested in modelling the environment, so, they do not explore this task and focus their attention on the sound source and on the response obtained by the acquiring system.
- It seems that the low accuracy and resolution is directly dependent on the number of training position used. We can think about a new system based on the measurement proposed by Willert but with a deeper training set and we can observe if we can get better results.

Our proposal is to implement the original algorithm using other signals for the training and the testing to understand the real powerful of the system. Then we would like to improve it, using also the knowledge acquired during the experiment done with the Deleforge algorithm so that we can obtain a simple, but useful algorithm than could be applied in any kind of rooms and environmental conditions.

Chapter 4

Assessment and improvement of the Deleforge algorithm

In the next sections we describe the experiments we performed on the two analysed algorithms. We started from [12] and [14], taking the original Matlab code and we checked the results. The Deleforge algorithm basically works in two steps: a training stage using white noise sources emitted all around the sensors and a test stage in which the authors recorded some people talking from different positions in a dataset that they used to test the algorithm. In the first step we simulated the original experiment. The core of the Deleforge algorithm is a Piecewise Affine Mapping, an algebraic tool that allows to link the position of an incoming sound to a feature vector composed by concatenating ILD and IPD. Starting from the original code, we understood how the code works and on which parameter Deleforge acted. No analysis of sampling frequency, spectral components, windowing, filtering is present in the paper they published. So, we made some changes in the code and added a new preprocessing stage to test all the parameters trying to achieve better results than the original. The modified Matlab code is tested 78 times, finding out a new version of the system that allows to reach improved results.

4.1 The original code

As described in Chapter 3, the algorithm described in [14] and [15] aims to localize the azimuth and the elevation source direction using as binaural cues IPD and ILD. The measurement are taken in a time-frequency framework using an STFT, then are grouped together in a unique longer feature vector called ILPD. This cue is passed through a Probabilistic Piecewise Affine Mapping algorithm that designs the relationship between the ILPD and the source direction. This procedure is repeated 2 times; First to train the system using a set oh 432 white noise samples emitted all around the listener. In this way, the algorithm can get all the parameter needed for the affine mapping of new data. Then each new incoming data is passed through the PPAM algorithm using the parameters trained before. You can find in Appendix A all the original MATLAB codes used.

The INRIA group put online all the MATLAB scripts, the functions and the toolbox used and you can download them freely. Each stage of the algorithm is developed in a particular function. We focused our attention on the function called *EXAMPLE_training1source* and *EXAMPLE_test1source*. Some of the other functions are used for solving other kind of localization problem that we are not interested in this thesis (e.g., separate and localize 2 sound sources, use audio-video cues). On the INRIA department website, you can find also 2 complete dataset called *AVASM* and *CAMIL*, created for the training and testing of this code, but that can be freely used for each type of sound signal processing experiment.

In *EXAMPLE_training1source* Deleforge et al. use the following parameters:

- Sampling frequency $f_s = 16KHz$
- Time window $time\_window = 64ms$
- Spectrogram overlap $overlap = 87.5\%$

They load each of the 432 sounds in the AVASM dataset, for each of them the algorithm calculates the IPD and ILD as 513-length separated vector, creates the ILPD training vector, loads the ground-truth position and trains the system using PPAM whose number of piecewise affine components changes from $k = 2$ to $K = 32$. The parameters found out by the PPAM and the binaural cues are saved as *.mat* files.

In *EXAMPLE_test1source* the system uses the same parameters as before, it selects the desired K and uploads a sound from the test set of the AVASM dataset. Then, it calculates IPD, ILD, creates the ILPD vector,

loads the ground-truth and the PPAM parameter, does the inverse PPAM to obtain an estimate of the sound source position expressed in pixel position (the screen coordinate of the camera). The function calculates also the localization error between the estimate and the ground-truth.

4.2 Preliminary hypothesis

Our goal was to find out how much the performance of this algorithm is related to the algebraic tool and how a digital signal preprocessing to the data could affect its stability. We would also like to understand its computational cost to discover if a system like this one could be exported on an embedded device.

From the analysis of the algorithm we did in Chapter 3 we found out some critical point. The absence of a deep signal processing analysis can not allow us to know in which situation the system can work. It seemed that they could get some results only in this very restrictive set of parameters, so we would like to demonstrate that changing them the results change dramatically and we would like to find out which parameter combination let the algorithm having useful results. Furthermore, it seemed for us that they did not consider at all the fact that they are working with speech and audio signal in reverberant rooms. We could define a wave propagating in the space as an audio signal if it is audible by human, so if its spectral content is in the interval $20Hz - 20KHz$ as a first rough approximation. Psychoacoustic studies can show that the spectre of a human speech signal is in a narrower interval between $1KHz - 4KHz$. We also know that the integration time of the human auditory system is about $20 - 32ms$ (check for Chapter 2 for more details). For these reasons, we thought that a richer filtering stage composed by different kind of filters and windows could improve their result and the experiment we explain could demonstrate it.

4.3 Experiments

All the experiments described here use $K = 16$ as parameter for the PPAM. The standard sampling frequency used is $Fs = 16KHz$ and the standard window length is $64ms$. Notice also that, as described in Chapter 3, the error (average + standard deviation) are measured in pixel position. Using the parameters of the camera they used, the conversion between pixels and angular distance is: 23 pixels = 1°. If anyone of the parameters above is changed, we highlight it, unless they remain the same ones.

fs	azimuth err. (pixel)	elevation err. (pixel)
4KHz	23.08 ± 17.48	$\mathbf{22.86 \pm 20.71}$
8KHz	22.28 ± 18.24	26.14 ± 23.50
16KHz	$\mathbf{21.62 \pm 14.95}$	32.29 ± 38.82
32KHz	NaN	NaN

Table 4.1: Inference of the sampling frequency on the mean azimuth and elevation localization error.

First, we executed the original code as it is, without any further modification, to verify the correctness of the result presented by the authors in [12] and [15]. The test is executed over the first 50 test sounds available in the AVASM dataset then averaging the localization error obtained. We got an error in the azimuth of 24.24 ± 16.17 pixels and in the elevation of 38.87 ± 49.46. This result was consistent with the ones reported in the article.

Then we checked how much the sampling frequency acted on the algorithm. The test was done using all the 108 test sounds. The results are showed in Table 4.1.

Choosing $Fs = 32KHz$, the algorithm became very slow (the training stage is completed after minutes of execution) and sometimes the PPAM algorithm failed due to an *Out of memory* error of Matlab. The algorithm required so much memory that sometimes could not be executed on a standard 4 GB RAM desktop PC. If we would like to create an algorithm that only do a localization task, we could think to use less memory using a lower sampling frequency because the average error was still reasonable at low frequency. If its aim was also to separate data or you are interested in preserving the incoming sounds for further processing, a sampling frequency less than 8KHz could make not more understandable speech and music signals.

Then we would like to filter the incoming audio signal, remembering that we are analyzing speech signals. Using the *FDAtool*, the Filter Design Analysis tool belonging to the Matlab Signal Processing Toolbox, we designed 11 Hann bandpass filters with different orders (from 10th to 100th) and cut-off frequencies. Their name is in the format *X_Y_Z*, where X is the order, Y is the high-pass frequency in KHz and Z is the low-pass frequency in KHz. We found the result as the average between 50 sound test in the Table 4.2.

As you can see, it seems that the filters acted mainly on the elevation estimation, changing dramatically the results. The lower error is achieved using the filter called *30_1_2*: both azimuth and elevation error went down. It seems that filtering could be a useful task to reach better results.

Filter	azimuth err. (pixel)	elevation err. (pixel)
no filter	24.24 ± 16.17	38.87 ± 49.46
30_0.1_4	23.64 ± 16.18	32.72 ± 36.28
30_0.1_2	28.95 ± 25.14	26.49 ± 21.82
30_1_2	$\mathbf{22.30 \pm 16.83}$	$\mathbf{26.49 \pm 21.38}$
30_1_4	23.36 ± 18.15	41.84 ± 56.83
30_2.6_3.8	23.79 ± 17.38	80.24 ± 90.75
100_0.1_4	23.75 ± 16.29	32.16 ± 36.14
100_0.1_2	24.30 ± 16.43	27.79 ± 22.86
100_1_2	23.17 ± 18.76	32.10 ± 39.81
100_1_4	22.41 ± 16.96	55.00 ± 75.41
10_1_2	27.79 ± 24.23	38.88 ± 49.14
10_1_4	23.41 ± 17.17	32.08 ± 35.74

Table 4.2: Inference of different Hann bandpass filters on the mean azimuth and elevation localization error. The first row shows the baseline, the first column describes the filter used, the following columns describes the mean and the standard deviation of the azimuth and the elevation error described in pixel. The best results obtained are in bold.

Taking the idea from [45] we would like to test this algorithm also with a filterbank whose features could describe the human perception system. It was one of the existing implementation of a Bark scale, based on the ERB bilinear transformation found in [10]. We did some little changes on this Matlab function to obtain the one we need for our scope. We called it *PRADO_oct3bank*. You can find the script in Appendix A.

Table 4.3 resumed the average error we got using the 108 test sounds. Looking at the result, we could state that a preprocessing stage using an ERB filter could be useful and it improved the algorithm, but only using it carefully. As we described in Chapter 2 and 3, the Deleforge algorithm acts directly on the frequency response of the environment in which the experiment was executed. Filtering the training sounds modifies the PPAM algorithm, so modifies the affine projection parameters. The experimental data tell that this modification is dangerous. We could state that a deep filtering stage in the training stage could modify the room impulse response, acting on the frequency component that contains the most important localization information. In other words, a filterbank in the training stage could distort the original frequency response of the environment, obtaining a distorted training parameter set. So, ERB filter could help the system in estimating the sound source position, but the training stage must be done

Filter	azimuth err. (pixel)	elevation err. (pixel)
no filter	24.24 ± 16.17	38.87 ± 49.46
train + test filterbank	22.16 ± 18.43	32.35 ± 26.84
test filterbank	$\mathbf{20.30 \pm 14.66}$	$\mathbf{27.21 \pm 29.84}$

Table 4.3: Inference of a ERB filterbank on the mean azimuth and elevation localization error. The first row shows the baseline, the second row describes the result obtained using the filterbank both on the training and the test set, the last row describes the experiment in which we used the filterbank only on the test signal, using the PPAM parameter got by the standard training stage with no preprocessing at all. The first column describes the filter used, the following columns describes the mean and the standard deviation of the azimuth and the elevation error described in pixel. The best results obtained are in bold.

Filter	azimuth err. (pixel)	elevation err. (pixel)
no filter	24.24 ± 16.17	38.87 ± 49.46
dereverb	23.93 ± 15.95	37.73 ± 49.23
30_1_2	$\mathbf{22.30 \pm 16.83}$	$\mathbf{26.49 \pm 21.38}$
dereverb + 30_1_2	$\mathbf{22.32 \pm 16.87}$	$\mathbf{26.38 \pm 21.34}$

Table 4.4: Inference of the dereverberation filter and Hann bandpass filter on the mean azimuth and elevation localization error. The first row shows the baseline, the second row describes the result obtained using the dereverberation filter, the third row describes the experiment using the Hann bandpass filter called 30_1_2*, the last row shows the results obtained using both the filters. The first column describes the filter used, the following columns describes the mean and the standard deviation of the azimuth and the elevation error described in pixel. The best results obtained are in bold.*

without adding any kind of distortion. This was the case in which the azimuth error could be reduced up to the 6% and the elevation error could decrease up to the 16% with respect to the original system. Notice that the result for the azimuth was the best we achieved until now.

The next step was to design a simple system that could interact with the environment reverberation. We found a useful algorithm in [43] that use a dereverberation system to separate direct sound to room reflections for a further processing in their localization algorithm. We decided to use the dereverberation algorithm only to filtering out the noise, but the entire algorithm could be useful for other future experiments in this research field. Using always the same subset of 50 test sound we can obtain the result in the Table 4.4 .

It seems that the results we got are improved thanks to the preprocessing stage that we were proposing, in particular the filtering stage was the one

time_window	fs	azimuth err. (pixel)	elevation err. (pixel)
64ms	4KHz	$\mathbf{23.08 \pm 17.48}$	22.86 ± 20.71
	8KHz	22.28 ± 18.24	26.14 ± 23.50
	16KHz	21.62 ± 14.95	32.29 ± 38.82
16ms	4KHz	79.09 ± 105.78	101.15 ± 109.98
	8KHz	41.93 ± 49.26	74.26 ± 79.53
20ms	4KHz	54.68 ± 84.09	61.72 ± 75.53
	8KHz	26.13 ± 23.12	59.55 ± 80.10
32ms	4KHz	36.56 ± 36.11	35.07 ± 35.97
	8KHz	24.78 ± 21.70	31.33 ± 27.77
48ms	4KHz	23.67 ± 20.11	25.84 ± 21.21
	8KHz	23.03 ± 18.21	26.93 ± 26.08
80ms	4KHz	23.75 ± 18.70	$\mathbf{20.49 \pm 17.66}$
	8KHz	23.63 ± 19.45	24.00 ± 21.89
	16KHz	23.40 ± 20.92	27.61 ± 26.31
96ms	4KHZ	23.20 ± 18.54	24.89 ± 21.21
	8KHz	$\mathbf{22.00 \pm 17.67}$	25.33 ± 24.33
	16KHz	$\mathbf{20.65 \pm 16.48}$	$\mathbf{24.66 \pm 21.21}$
128ms	4KHz	23.40 ± 18.29	21.32 ± 20.79
	8KHz	22.52 ± 19.49	$\mathbf{23.32 \pm 20.87}$
	16KHz	23.76 ± 20.70	25.06 ± 21.61

Table 4.5: Inference of the sampling frequency and of the window length on the mean azimuth and elevation localization error. The first rows shows the baseline. The first column describes the window length used, the second column describes the sampling frequency used, the last columns show the mean and the standard deviation of the azimuth and the elevation error described in pixel. The best results obtained are in bold.

thanks to which we are getting better result. The azimuth error decreased about the 8% and the elevation error decreased about the 32%.

Another critical point we found out on the original code was the choice of the window length used for the FFT and the creation of the interaural cues. The INRIA group chose 64ms, but without any justification. We did different experiments changing each time the window length from 16ms up to 128ms and the sampling frequency from 4KHz to 16KHz and using all the 108 test sound available. The result are shown in the Table 4.5.

As expected, the window size influenced the algorithm. A length lower than 48ms creates not more useful estimates, so demonstrating the psychoacoustic theories about integration time of an auditory system. The most use-

ful sizes seems 64ms, 96ms and 128ms, using $fs = 8KHz$ or $fs = 16KHz$. The choice of one of them did not affect too much the resolution in the azimuth, but the change in the elevation error were visible. We could state that if the aim of the algorithm was the estimate of the incident angle of the sound source (so we are interested in the azimuth only), the best result found out is using $time\_window = 96ms$ and $fs = 16KHz$. If we must take care about the execution time of the algorithm, we noticed that long windows slow down the convergence of the training stage, maybe because longer vectors make the affine mapping more difficult.

The aim of the last experiment we built up is to summarize all the previous ones. We combined some of the previous hypothesis we did on the processing stage and on the parameters and we compared the result, using the 108 test sounds available. The Table 4.6 describes this comparison. The number in the first column describes the experimental parameters:

1. Original experimental setup
2. 16KHz, 64ms, no filtering for the training + 16KHz, 64ms, ERB filterbank, dereverberation filter for the test set
3. 16KHz, 64ms, ERB filterbank for the training + 16KHz, 64ms, ERB filterbank, dereverberation filter for the test set
4. 16KHz, 96ms, no filtering for the training + 16KHz, 96ms, no filtering for the test set
5. 16KHz, 96ms, ERB filterbank for the training + 16KHz, 96ms, ERB filterbank, dereverberation filter for the test set
6. 16KHz, 96ms, no filtering for the training + 16KHz, 96ms, ERB filterbank, dereverberation filter for the test set

The parameters we chose depends on the results we got in the previous experiments: from the analysis on the sampling frequency, it seems that $fs = 16KHz$ could be an optimal choice; the window length optimal parameters we found out are in the range between $64ms$ and $96ms$; the ERB filterbank and the dereverberation function could be useful and resulted in lower localization error in the previous simulations.

The best result are the setup 2 and the setup 6, the ones with no modifications on the training set and both dereverberation and ERB filters used for the sound source estimate. The setup 6 improve the original system for the 5% in azimuth precision and for the 27% in elevation precision with respect to the original algorithm.

Setup	azimuth err. (pixel)	elevation err. (pixel)
1	21.62 ± 14.95	32.29 ± 38.82
2	$\mathbf{20.23 \pm 14.92}$	27.58 ± 30.04
3	21.04 ± 16.24	26.70 ± 29.13
4	20.65 ± 16.48	24.66 ± 21.21
5	24.17 ± 23.83	24.75 ± 20.18
6	$\mathbf{20.55 \pm 16.62}$	$\mathbf{23.51 \pm 20.56}$

Table 4.6: Final experiment using 5 different preprocessing stages. The first column describes the filter used, the following columns describes the mean and the standard deviation of the azimuth and the elevation error described in pixel. The best results obtained are in bold.

4.4 Analysis of the results

After the execution of all this experiment based on the MATLAB framework and on the data available in the original research, we could state that some of our original idea could be useful and could improve the system. We could summarize the most important consequences of our simulations in the following points:

- The choice of the input parameters is essential to get the best result and to manage the memory used by the algorithm. A lower sampling frequency resulted in worse localization performance, but speeded up the convergence of the algorithm, in particular in the PPAM stage of the training algorithm that is the heaviest part of the system. If some energy or memory management is needed in a localization system, $fs = 8KHz$ could be a reasonable trade off. Notice that the best result is achieved with $fs = 16KHz$ and for $fs < 8KHz$ the error started increasing sensibly. The same statement is valid for the window length: the lowest error is obtained using $time\_window = 96ms$, a good trade off between efficiency and precision could be $time\_window = 64ms$. $time\_window < 32ms$ is not useful at all.

- Reverberation affects the sound source estimate. The usage of a dereverberation filter could be a good choice if you want to improve the localization task.

- Filtering the audio signal seems a fundamental task, which influence on the data allow us to improve the original system. The ERB filter wa the example we decided to use, but also some simpler bandpass filter, as the one we called 30_1_2 could help. The choice must be done

always keeping in mind the main goal of the algorithm: if we could have more computational power, a filterbank could be a solution.

- We could state that the effectiveness of the algorithm is strictly linked to its training set. A heavy filtering stage in the training could modify the parameters found by the PPAM, damaging all the system. It seems that this algorithm can not be used for a general purpose system because of the relationship between the environment and the training stage.

To validate our analysis, we need to repeat some of the experiments. We would like to build an experimental setup with the aim of reproducing the entire system in different rooms with respect to the standard one, create new dataset by our own, change parameters freely (e.g., the distance between the microphones, the frequency sampling, the number and the kind of training sounds). The final goal is to confirm our first conclusions, to have a deeper understanding about the effectiveness and the reliability of the algorithm and to understand if its results are good only in the particular conditions they describe in the paper or if a general purpose system can be developed.

Chapter 5

Improvements of Deleforge algorithm for real environments

We thought that an analysis of the effect of reverberation and of the dimension of the room is essential to understand how the sound waves propagate from the source to the microphones, so how much they can affect the effectiveness of the algorithm. So we built up an experimental setup to reproduce the original Deleforge experiment at the AIRLab. The main setup was composed by an array of eight MEMS microphones, an external sound card and an experimental board, all of them by STMicroelectronics. With this system we could record eight audio tracks all together in PCM at 16KHz, 16 bit. As audio framework, we used Audacity and also Matlab to handle input data. As audio sources, we used a computer speaker already present in the laboratory and a Genelec 8030. Before doing any experiment and implementing the Deleforge algorithm in a real environment, we tested this setup to be sure about the features described in the data-sheets and we figured out some problems on the lab speaker and, most importantly, on three MEMS that did not work properly. So we changed the planned setup, using only the five working microphones mounted on a bar at different distances.
The setup was used also to acquire two new audio datasets, one for the training and the other one for the test stage of the Deleforge algorithm. Each dataset contains all the possible combination of stereo audio signals available with 5 microphones, the sensors positions and the sources positions, taken thanks to OptiTrack, a commercial motion capture system used to precisely localize the sound source with respect to the microphone array. Then, we

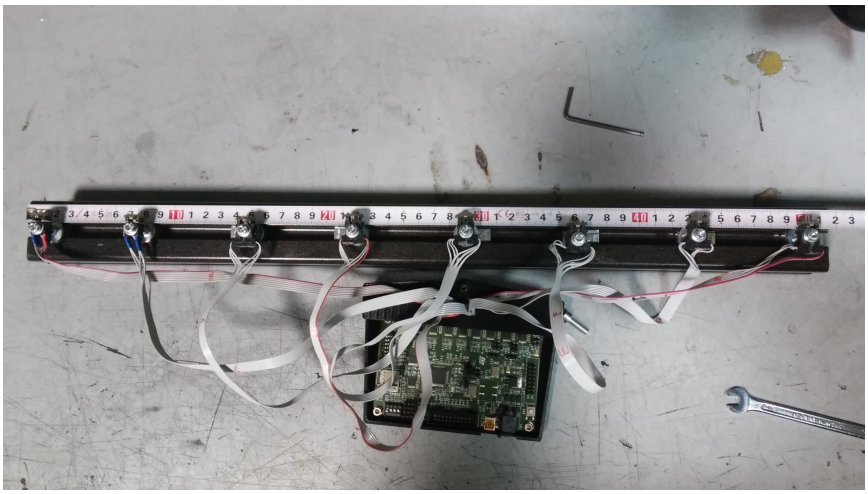

Figure 5.1: The planned microphone setup. The reference microphone is the right one. As explained in the text, the final setup used only 5 of the 8 microphones sown by this picture

used our new dataset on the already modified Deleforge code to test its efficiency on new data, in a new environment and with a real setup. The goal was to stress the algorithm to understand how results change according to the algorithm parameters and if the validity of the preliminary proposals on the Deleforge algorithm we did in Chapter 4 could be demonstrated or not.

5.1 The experimental setup

In Fig 5.1 you can see the 8 omnidirectional MEMS microphones and the 16KHz, 16 bit sound card we planned to use in our experiments. The system is by STMicroelectronics and it was already available in the laboratory. With this material, we could study also the Deleforge algorithm strength with respect to the distance between the two sensors and its dependency from other parameters of the system (e.g., sampling frequency, window time-step). We received the ST setup without any datasheet, so we did some tests to understand how it works, if there are some limitations or some particular tools to know to use it.

We built the first setup putting the 8 microphones at different distances on a bar so to create an array, as you can see in Fig 5.1. Taking the first microphone as a reference, all the other ones are at position: 7cm - 15cm - 21cm - 28cm - 35cm - 42cm - 49cm. The sensors are connected to the sound board and the sound board is linked to a computer via USB. In a Windows

Figure 5.2: The Phonic PAA2

framework we found some problem, but Linux recognized the external sound card without any problem so we used a Linux machine for this stage. We decided to use as a registration audio tool Audacity, an open source and easy audio editor that can allow us to register the 8 audio track simultaneously.

We also made some measurement with a Phonic PAA2 (see Fig 5.2), a handheld professional audio analyzer that includes in a unique object an omnidirectional condenser microphone, a 31-band 1/3 octave real-time spectrum analyzer, SPL meter, A and C weighting and a noise and test signal generator. PAA2 can be linked to a loudspeaker, so we decided to use it also as a sound source. Furthermore, we downloaded other test signals (e.g., different white and pink noises, sweep sine, sine tone).

With the first experiment, we would like to understand if the sound card really works at 16KHz registering 8 track simultaneously. We emitted from a loudspeaker a white noise, a signal which has a flat spectre between the audible frequency range, and a sweep sinusoid signal. So, depending on the sampling frequency, we could notice if there is aliasing and how much the signal were affected by this phenomenon. Selecting the "MAX" available sampling frequency, we did the registrations and we heard the results. The aliasing phenomenon was evident, in particular hearing from the acquired sweep sinusoid, and also the spectrogram plot by Audacity showed this limitation. The maximum sampling frequency available is confirmed

and is 16KHz.

Then we made a test emitting a white noise from the loudspeaker in front of each microphone at the same distance and at the same SPL volume. Our goal was to understand how the frequency response and the polar diagram of each MEMS is. Then we used an FFT tool available freely in the Audacity framework to make a comparison between each signal spectrum. The resulting graphics were automatically scaled by the tool, so the comparison could be a little more problematic, but this was sufficient for our scope. We were interested in a first rough understanding of the setup and we wee not willing to characterize completely and deeply it. In the Fig 5.3 and 5.4 you can compare the results between the 8 microphone and we find out an unexpected problem: 3 MEMS were not working properly. If you notice the dB_{FS} values on the vertical axes, you can find lower values for microphones we defined as 2, 6 and 8. The difference is about $4dB_{FS}$, that is audible, so can introduce an important distortion in the whole acquiring process.

To be sure that the problem was due to the MEMS and not the sound device or to the USB port, we did from scratch the experiment exchanging the microphones, with another USB port and with another computer. The results were the same, so the problem was linked to the sensors cited above. The easiest way to solve this task was to throw away these microphones. We did some changes in the setup so to use only 5 sensors instead of the original 8 (see Fig. 5.5 and 5.7).

From this test we could make a simplified but useful description of the frequency response and polar pattern of the microphones. To go more in depth, we did another experiment emitting continuously a white noise and moving the speaker all around the MEMS array. The spectre was not linear, nor constant with respect to the frequency and they were omnidirectional as we were supposing. To be sure that the non-linear response was due to the microphone and not to the speaker, we needed to do another experiment.

Using the PAA2 (see Fig. 5.2) we started to do a basilar characterization of the computer speaker we used in the previous experiments. We emitted white and pink noise and measured through the PAA2 the emissions. We focused our attention in the frequency range covered by the acquiring system (maximum frequency = 8KHz). The frequency response of the pink noise observed by the 1/3 octave spectrum analyzer is constant in the interval between 160Hz and 8KHz (there were some oscillation due to the signal that for its stochastic nature is not constant over time) and we observed an important decreasing for the lower frequencies. A further analysis was done using the white noise: we observed an increasing logarithmic response in the

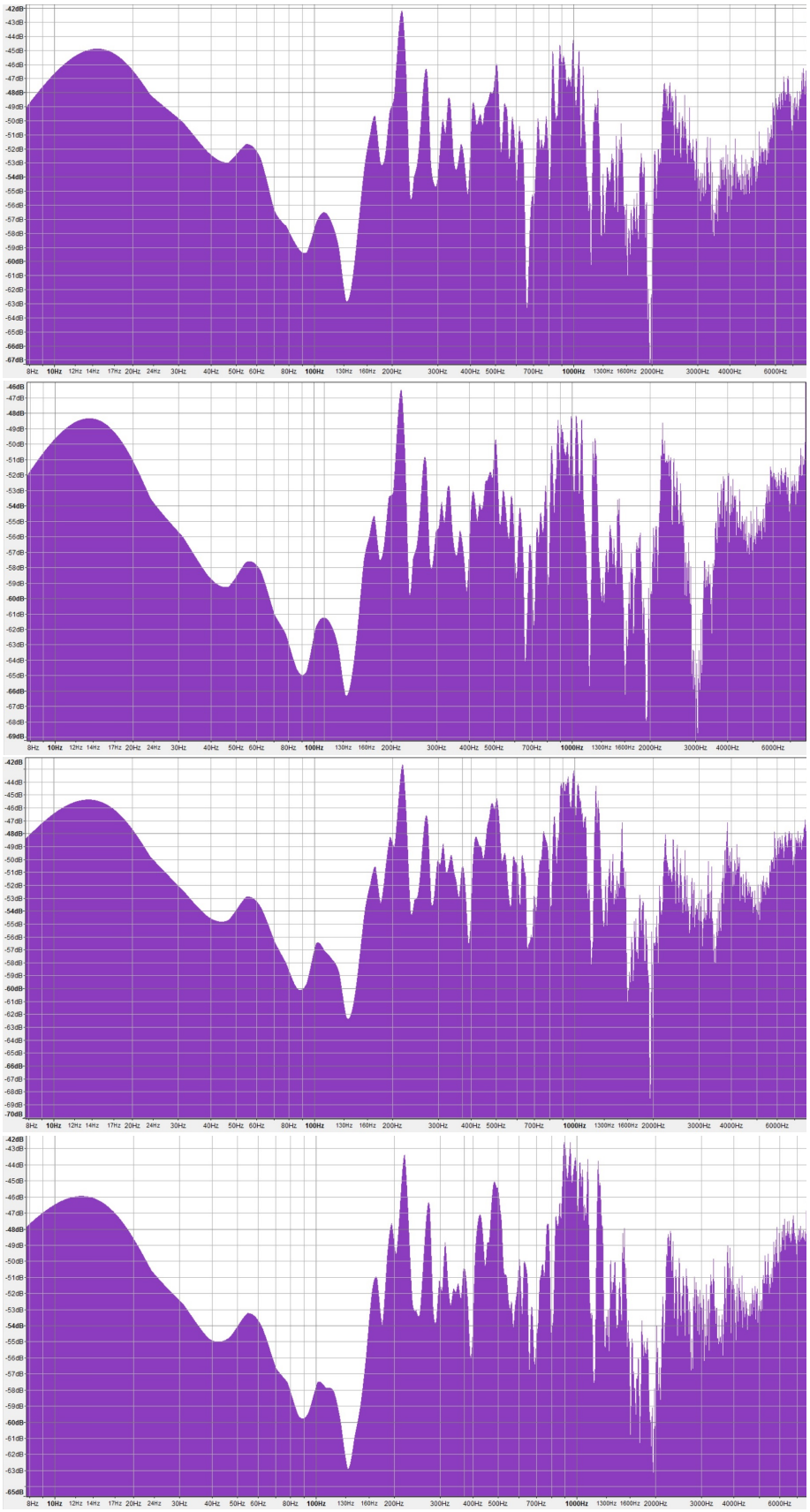

Figure 5.3: Comparison between the frequency response of each microphone. From top to the bottom you can see from mic 1 to mic 4. The unity of measures are dBFS for the vertical axes and Hz for the horizontal axes. Notice that this Audacity tool autoscales the results.

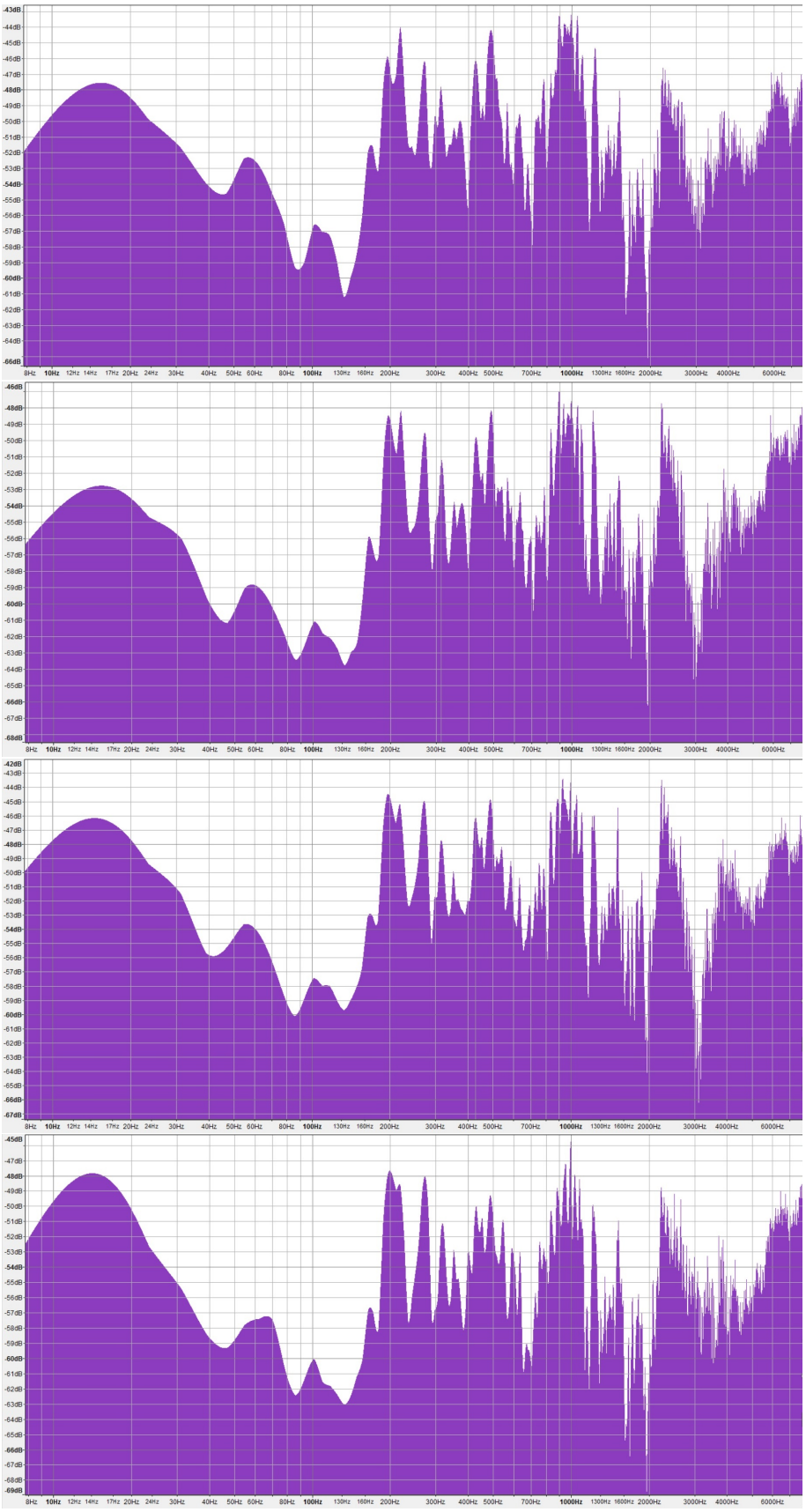

Figure 5.4: Comparison between the frequency response of each microphone. From top to the bottom you can see from mic 5 to mic 8. The unity of measures are dBFS for the vertical axes and Hz for the horizontal axes. Notice that this Audacity tool autoscales the results.

Figure 5.5: The final setup

Figure 5.6: The Genelec 8030

same main interval (between 160Hz and 8KHz) with a slope of 3dB/octave and a abrupt performance decreasing for low frequencies. Looking at these results we can state that the loudspeaker has a severe limitation in the low frequency range, but for middle and high frequencies had good performances. To avoid some possible future problems, we decided to use also another loudspeaker, a professional one by Genelec: the 8030 model (see Fig 5.6). This is a studio speaker, designed to have a flat response between 58Hz and 20KHz, so it works well for a bigger frequency interval than the computer speaker we used before.

One of the main goal of this research was to build up the original experiment and reproduce it changing its parameters and testing it with new sound test. So, we would like to save the loudspeaker position to create the new sound source ground truth. To make it easily, we decided to reproduce our experiment in the AIRLab. The laboratory is equipped with a motion capture system called Optitrack, based on 12 IR cameras and a series of markers, and a tracking software called Motive Tracker. The lenses of the cameras are surrounded by IR light emitting diodes (LEDs). This light is reflected in the opposite direction by reflective markers and captured by the cameras. Hence, the cameras only see a set of small dots in their 2D images. Knowing the relative position between the cameras, the system can compute the 3D position of the dots from their 2D positions in the camera images via a calibration routine.

With the setting visualized in Fig 5.5 and Fig 5.7, we got 10 possible couples whose distance between the sensors are: 6cm, 12cm, 14cm, 15cm, 20cm, 21cm, 26cm, 27cm, 41cm. The MEMS array was put in the center of the room, mounted on a vertical bar whose height is 150cm. The loudspeaker was on a 80cm-height structure that can be moved manually all around the experimental setup. We decided to fix the distance between microphones and speaker to 2m and we moved the structure in the frontal half-plane with respect to the array, as you can see in Figure 5.9.

5.2 Experiments on the Deleforge algorithm in a real environment

In Chapter 4 we proposed some modification and improvement on the algorithm described in [12] and [15]. Our new proposals were tested on the original Matlab code using only the original data available from the INRIA group experiment. The main idea of the experiments that we are going to explain in this chapter is to demonstrate the hypothesis we did on the sound

Figure 5.7: The final version of the array we will use in the experiment

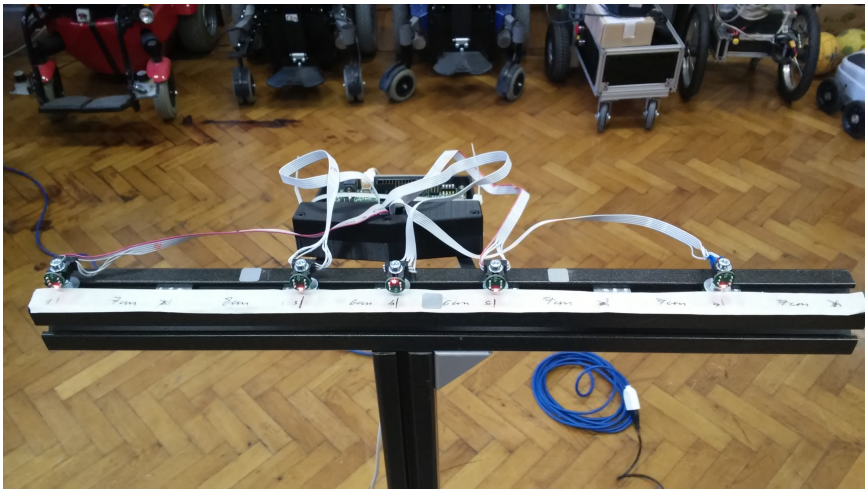

Figure 5.8: The definitive version of the MEMS array

Figure 5.9: The experimental setup using the Genelec 8030

source algorithm reproducing entirely the experiment with an original setup, in a real reverberating chamber and with original dataset. Fig 5.8 shows the setup.

We would like to understan how the environment could affect the affine mapping and how much the algorithm is stable with respect to the quality of the loudspeaker, the distance between the microphones and the angular position of the sound source, provided we are interested in a robot deployment.

The experiment is organized in the following way:

- We needed to create our original training and testing dataset, using both the laboratory loudspeaker and the Genelec 8030, acquiring the incoming sound with all the 5 available microphones and saving the ground truth positions.
- We analyzed the new dataset and compared it with respect to the one used by the INRIA group
- We executed the algorithm with the improvement we introduced before comparing the result using different subsets of our dataset
- We acted on the parameters, so to understand the pros and the cons of the PPAM-based algorithm

5.3 The new database

Fig 5.7 and 5.9 showed the experimental setup used for the creation of our own dataset. For the training dataset, we made 289 multi-track registrations which generated a database containing 2890 stereo tracks whose length is about $2.5s$. Fig 5.10 shows the position of all the object involved in this experiment.

We are interested in saving the ground truth position using the Optitrack. So, firstly, we completed the initial setup of the motion capture system and the related markers we put on the array and on the loudspeakers defining the coordinate system represented also in Fig 5.10. We put the origin of the cartesian system on the microphone 3 and we moved manually the loudspeaker on the frontal half-plane at a mean distance of $2m$. The calibration of the Optitrack allowed us to make the measurement with an error of $0.5cm$.

Then we started acquiring the training sounds. We used a slightly modified white noise to eliminate the aliasing phenomenon. Remember that the

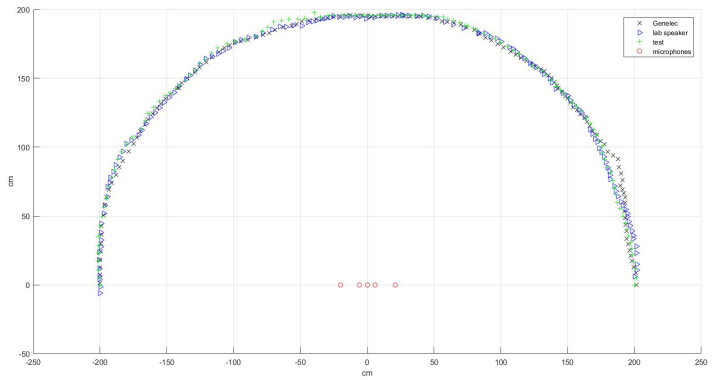

Figure 5.10: The position of the 5 MEMS, the training and the test sounds

sound card can register at a maximum sampling frequency of $16KHz$, so we decided to filter a standard white noise using a *Butterworth low-pass filter* with a cutoff frequency of $8KHz$. Due to our experimental limitations, we focused our attention only to the frequency range between $20Hz$ and $8KHz$, so we could deal with our training signal as if it was a white noise. We know that a white noise is by definition a stochastic signal whose spectrum is spread among all the audible frequency, but, for brevity, we defined our input signal as white noise and we used this term for the rest of the thesis.

The recording stage was done as a series of multi-track session acquired in the Audacity framework. Then each single audio track was cut by this session using some Audacity tools and saved as a *.wav* file. The ground truth was saved firstly manually as a *.txt* file, then using a Matlab script, we designed a Nx13 *.mat* table (see Fig 5.11), where N is the number of recording done, column 1 indicates the name of the sound file, columns 2 and 3 show the sound source coordinates and the columns from 4 to 13 show the sound source angle with respect to the center of each couple of microphones available. All the geometric consideration (g.e., microphones and speakers angles and positions), the ground truth and the audio tracks were then organized in folders in a dataset that we called *PRADO dataset*. All the dataset, the Matlab script used, the mono and stereo audio signal and the related *README* files, containing the instruction to correctly use and understand the content of the related files, are present in a *.zip* folder.

We created also a test dataset in the same way as described above for the training set. We emitted different kind of test signal from the Genelec 8030 put in many positions around the MEMS array, saving all their positions.

PLOTS VARIABLE VIEW

pos_gen

142x13 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	201.5000	0	0	0	0	0	0	0	0	0	0	0
2	2	201	8.5000	2.5956	2.5548	2.4582	2.5153	2.4215	2.3859	2.4275	2.3401	2.3069	2.2746
3	3	201	8.5000	2.5956	2.5548	2.4582	2.5153	2.4215	2.3859	2.4275	2.3401	2.3069	2.2746
4	4	201	8.5000	2.5956	2.5548	2.4582	2.5153	2.4215	2.3859	2.4275	2.3401	2.3069	2.2746
5	5	199.5000	13	3.9980	3.9348	3.7850	3.8735	3.7283	3.6732	3.7376	3.6022	3.5508	3.5008
6	6	198	17.5000	5.4183	5.3322	5.1282	5.2487	5.0509	4.9759	5.0636	4.8793	4.8092	4.7412
7	7	197	21	6.5286	6.4245	6.1781	6.3236	6.0847	5.9941	6.1001	5.8774	5.7928	5.7106
8	8	196.5000	25	7.7791	7.6552	7.3618	7.5351	7.2506	7.1427	7.2689	7.0037	6.9029	6.8050
9	9	195.5000	29.5000	9.2069	9.0601	8.7126	8.9178	8.5809	8.4531	8.6026	8.2885	8.1691	8.0531
10	10	194	33.5000	10.5142	10.3461	9.9481	10.1831	9.7972	9.6509	9.8221	9.4623	9.3256	9.1928
11	11	194	39	12.1923	11.9988	11.5405	11.8112	11.3667	11.1980	11.3953	10.9807	10.8230	10.6698
12	12	193.3000	43.5000	13.6005	13.3854	12.8758	13.1769	12.6825	12.4948	12.7143	12.2529	12.0774	11.9069
13	13	193	48.5000	15.1200	14.8826	14.3197	14.6523	14.1061	13.8986	14.1412	13.6311	13.4370	13.2483
14	14	192.5000	53.5000	16.6405	16.3810	15.7655	16.1293	15.5318	15.3047	15.5703	15.0118	14.7993	14.5925
15	15	193	59.5000	18.3391	18.0574	17.3884	17.7839	17.1340	16.8868	17.1759	16.5677	16.3360	16.1106
16	16	193	63.5000	19.4818	19.1851	18.4802	18.8970	18.2120	17.9513	18.2562	17.6146	17.3701	17.1320
17	17	192	66.5000	20.4328	20.1224	19.3845	19.8209	19.1037	18.8307	19.1500	18.4781	18.2220	17.9726
18	18	191	69	21.2427	20.9204	20.1542	20.6074	19.8626	19.5790	19.9106	19.2127	18.9465	18.6874
19	19	189.5000	72	22.2490	21.9117	21.1095	21.5840	20.8041	20.5071	20.8545	20.1235	19.8446	19.5731
20	20	191	76	23.1791	22.8337	22.0113	22.4978	21.6979	21.3929	21.7495	20.9987	20.7121	20.4328
21	21	190	81	24.6515	24.2874	23.4201	23.9334	23.0893	22.7673	23.1438	22.3509	22.0479	21.7527
22	22	188.5000	85.5000	26.0388	25.6567	24.7458	25.2849	24.3981	24.0596	24.4554	23.6217	23.3030	22.9924
23	23	187.5000	91.5000	27.7382	27.3367	26.3784	26.9457	26.0124	25.6557	26.0727	25.1940	24.8579	24.5300
24	24	184	94	28.8688	28.4482	27.4445	28.0388	27.0611	26.6875	27.1243	26.2039	25.8519	25.5085
25	25	180	96.5000	30.0957	29.6538	28.5991	29.2235	28.1963	27.8037	28.2627	27.2957	26.9258	26.5651
26	26	177.5000	102	31.8796	31.4157	30.3074	30.9638	29.8837	29.4707	29.9536	28.9358	28.5462	28.1661
27	27	177.5000	102	31.8796	31.4157	30.3074	30.9638	29.8837	29.4707	29.9536	28.9358	28.5462	28.1661
28	28	174.5000	104.5000	32.9863	32.5051	31.3552	32.0362	30.9154	30.4867	30.9880	29.9315	29.5270	29.1324
29	29	172	109	34.5162	34.0163	32.8208	33.5290	32.3633	31.9170	32.4388	31.3388	30.9174	30.5061
30	30	172	109	34.5162	34.0163	32.8208	33.5290	32.3633	31.9170	32.4388	31.3388	30.9174	30.5061
31	31	168.5000	111.5000	35.7295	35.2105	33.9688	34.7044	33.4934	33.0297	33.5719	32.4289	31.9910	31.5634
32	32	166.5000	115.5000	37.0492	36.5157	35.2382	35.9952	34.7488	34.2711	34.8295	33.6519	33.2004	32.7594
33	33	164.5000	118.5000	38.1237	37.5776	36.2692	37.0447	35.7676	35.2780	35.8504	34.6430	34.1798	33.7273

Figure 5.11: An extract from the PRADO training dataset

The signals we used are part of a freely available online dataset based on the *TIMIT dataset*. It's composed by about 400 audio files in which there is a person speaking a phrase in English. The list of words and phrases used is the one created by the TIMIT research: they were selected because they represent all the English existing phonemes. In this way, the database could be valid and useful for each kind of speech processing experiment like ours. This particular implementation of the TIMIT algorithm has been executed by different people, both male and female, and at different volume levels. This task was particularly useful for us, because we could stress the reliability of the system. The organization of the testing database was the same as for the training one.

Resuming, we created a training set composed by 2890 white noise signals emitted by 289 points, 142 of them where emitted by the Genelec 8030, the remaining 147 ones by the laboratory loudspeaker. The test set was composed by 3780 different speech signals emitted by 126 points by the Genelec 8030. All the audio signal were emitted at a distance of about $2m$ moving the speaker manually on the fontal half-plane as in Fig 5.11. All the ground truth were saved with a quantization error of $0.5cm$ as in Fig 5.10.

5.4 Experiments

5.4.1 Experiments on the database

The first experiment we executed is on the validity of the *PRADO dataset*. We would like to compare it with the original *AVASM dataset*, in particular we were focusing on the two training sets. We know that these two dataset were the result of the recording of some training sound in different environment, so comparing them means also comparing how the environment affects the data.

The AVASM dataset is composed by $44KHz$ audio streams whose white power spectrum reaches the maximum level of $-55dB/Hz$. We observe the STFT of the incoming signal in the INRIA system without any kind of filter, neither the ones we introduced and the ones already introduced by them (i.e., the resampling stage). The original emitted sound was a white noise. The figure shows clearly the distortion introduced by a non-anechoic chamber: the perceived sound is no more strictly white, has some periodic oscillations whose principal lobes are at 5KHz and 8KHz. This task is also known as the *Comb effect*. Then we compared this result with the same signal after the resampling stage. We remind that the original algorithm uses a downsample function reaching the sampling frequency of $16KHz$. As showed by Fig 5.12,

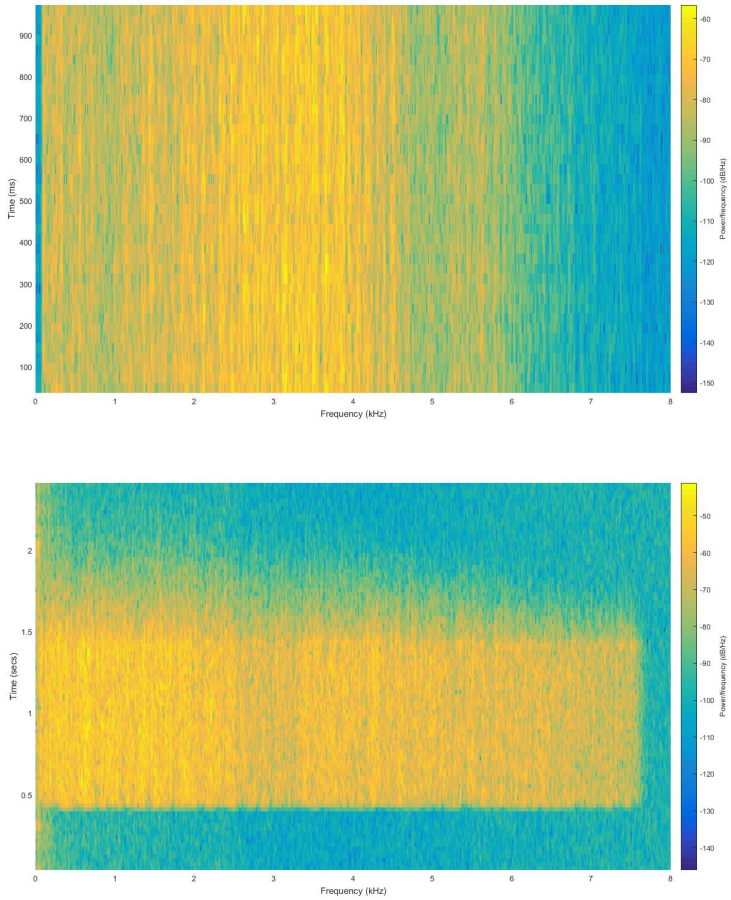

Figure 5.12: STFT of a training sound taken by the AVASM dataset (the top image) and from the PRADO dataset (the bottom one)

the maximum power spectrum level is always $-55dB/Hz$, but the distortion introduced are more evident, in particular in the frequency interval between $6.5KHz$ and $8KHz$. The *resample* function used attenuates significantly the signal in this range, changing its spectrum. Here, the power reaches the value of $-120dB/Hz$.

The PRADO training dataset is composed by $8KHz$ white noise signal, they are longer than the AVASM training ones and they contain also the attack and the release of the signal. As you can see in Fig5.12, we recorded more powerful signal (the maximum level if $-40dB/Hz$) and the STFT has a similar oscillating figure, but with larger lobes (due to the wider dimension of the room). We can think that the filter we built to cut the emitted white noise under $8KHz$ and the reverberation effect act together arising the total

amount of power received by the microphones, also influencing the higher frequency range. The spectral power at $f > 6.5KHz$ reaches the minimum level of $-100dB/Hz$.

It seems that the reverberating environment affects the incoming data, changing in particular the Signal-to-Noise Ratio and the different Comb effect generated by the dimension of the room, but the STFTs also show that these phenomena do not affect dramatically the signal. The changes are visible, but not so heavily as we expected.

To analyze more in depth this first assumption, we analyzed how the different signals affects the binaural cues. We calculated IPD and ILD for both PRADO dataset and AVASM dataset, changing also the input parameter (e.g., sampling frequency, window length, filters). We supposed that IPD and ILD could have very different results and that changing the parameters we will observe severe changes. In Fig 5.13 we can see the feature vectors calculated for 1 training signal; each row corresponds to a frequency bin, each column describes the cue for a time window. For the PRADO dataset we use the couple of MEMS composed by microphones 1 and 2 at a distance of $15cm$.

As we thought, if we change the input signal, the binaural cues change, as you can notice from the Fig 5.13. In general, IPD has a bigger interval values with respect to ILD. That's what we are expecting: as we describe in Chapter 2, the ILD cue is strictly connected to a task called *masking effect* that is due to the presence of an absorptive obstacle in the middle of the two sensors. Both our setup and the Deleforge one have not any kind of material that separate the two microphones, so the level difference between the 2 acquired signal is very low. Also for this reason, the changes in IPD are more visible than in ILD. We will focus our attention on the first cue.

Referring on Fig 5.14, we could easily state that the sampling frequency affects dramatically IPD. Basically, a higher sampling frequency allows the system to deal with more data (from 8KHz to 16KHz the frequency bin available for the FFT are doubling), so it can build a more precise feature vector. We compared also the results using the different filters we introduced before. The results are showed in Fig 5.15. Despite the result we had in previous experiments, it seems that the dereverberation stage filters out all the high frequencies, affecting dramatically the binaural measurement. This is an unexpected result, also because the result we got using also this filter were slightly better then avoiding to use it. Having this result, we decided for now to not use this filter.

We tested also how the distance between the microphones can act on the IPD calculation. Fig 5.16 shows the main result: we take the 4 couple com-

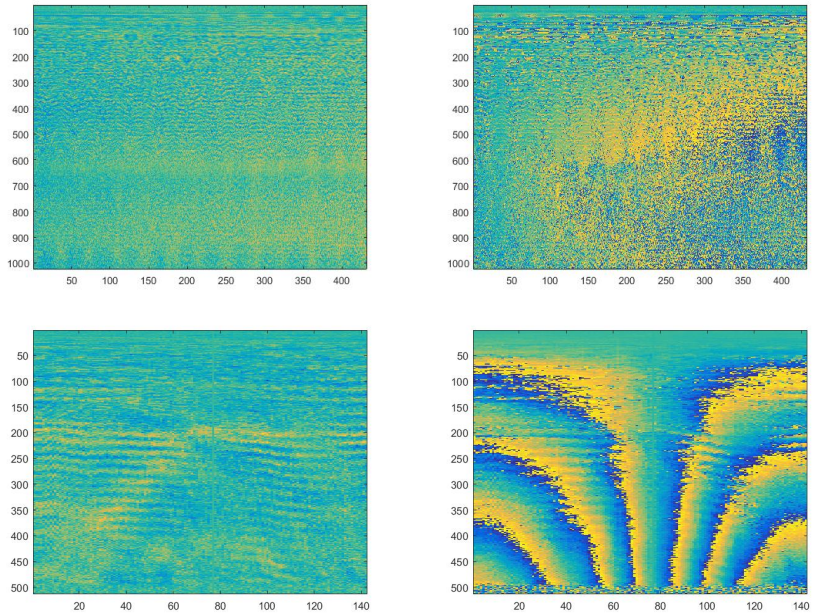

Figure 5.13: Comparison between the binaural cues obtained by the two different setups. The top 2 graphics show ILD and IPD cues calculated using the AVASM database framework, the other 2 have as input a training sound from the PRADO dataset. The horizontal axes represent the training sound processed, the vertical axes the frequency bin

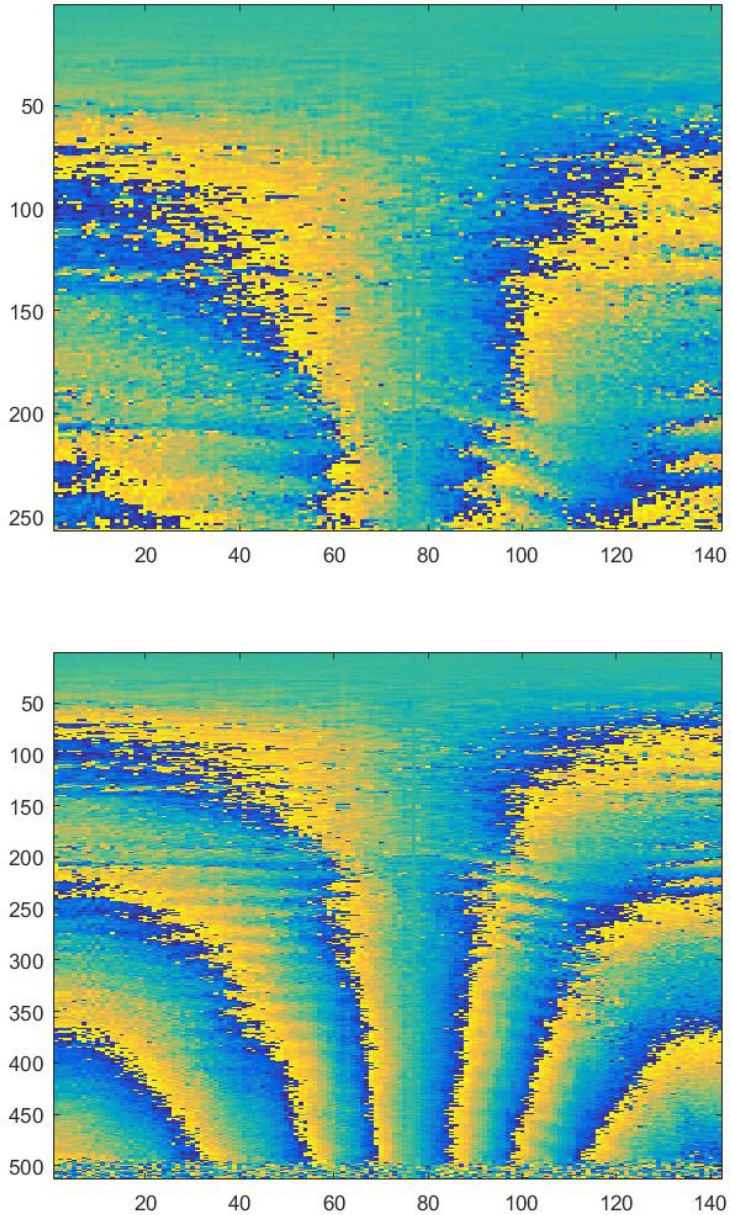

Figure 5.14: Comparison between the same binaural feature calculated for the same training set using $fs = 8KHz$ and $fs = 16KHz$. The horizontal axes represent the training sound processed, the vertical axes the frequency bin

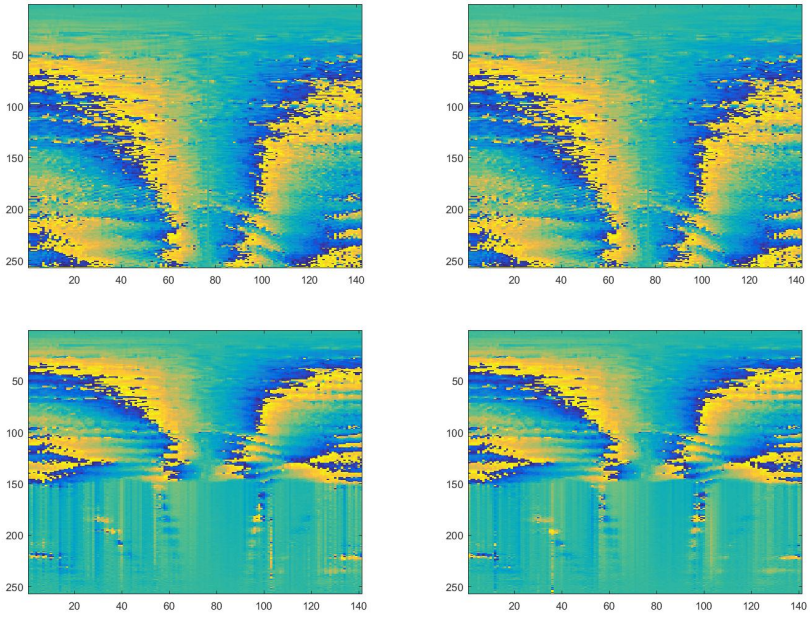

Figure 5.15: Calculation of 4 IPD using different filters. From the first to the last graphics: no filters, ERB-filter, dereverb filter, erb + dereverb filter. The horizontal axes represent the training sound processed, the vertical axes the frequency bin.

posed by microphone 1 and each other one and we do the measurement. As we thought and we explain also in Chapter 2, IPD depends on the distance; the binaural measurement changes deeply. We did not know yet if this can affect the whole sound source direction algorithm, but we can suppose that something could change in the convergence of the PPAM task.

5.4.2 Test on the improved Deleforge algorithm

We executed the localization algorithm with our new proposal using the MEMS couple composed by microphones 1 and 2, whose distance between them is $15cm$. We used as training set the complete PRADO training dataset and as test set a random selection of 100 speech signal taken from the PRADO test dataset. The algorithm parameters are the ones we found out as the best in the previous experiment: $Fs = 16KHz$, $time\_window = 96ms$, $overlap = 87.5\%$, dereverb and ERB-filter. Despite our assumption on the influence of reverberation on the result, we can get great result also in our laboratory.

We got this result after a series of test in which we focus our attention on how to filter the input data. As we described in this chapter, we recorded

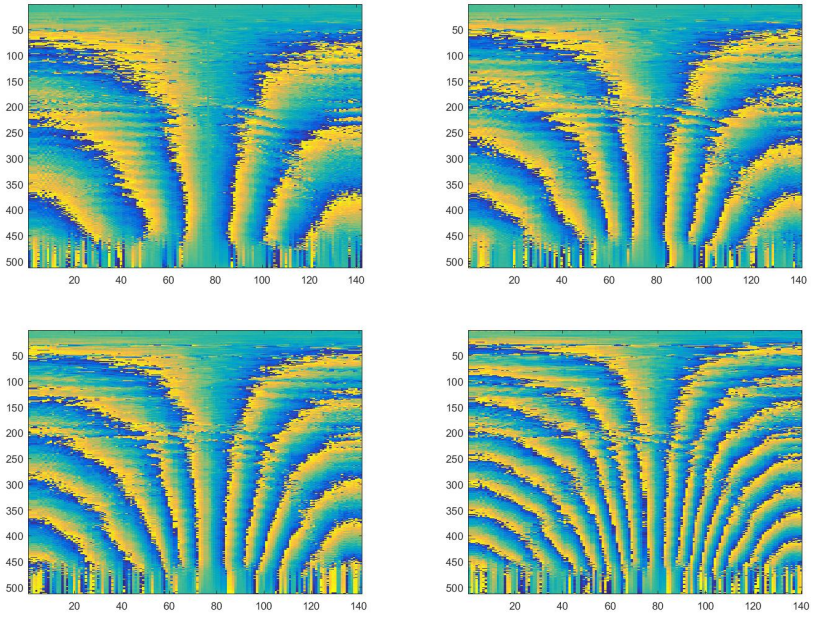

Figure 5.16: Comparison of the IPD calculated through 4 different setup: mic 1-2, mic 1-3, mic 1-4, mic 1-5. The horizontal axes represent the training sound processed, the vertical axes the frequency bin.

the PRADO training set saving also the attack and the release of the noise emitted. The interaction between the direct sound and the reflection due to the environment is mainly visible in this part of the signal. We create a filter to deal with the dereverberation task, based on a simple energetic threshold mask to separate the direct signal from the reflection, whose output is: the complete original input or only the central part of the signal (so when Signal-to-Noise Ratio or the Direct-to-Reverberant Ratio is over a high threshold). We use the term *1S* and *3S* to distinguish between the input signals described above. Table 5.1 shows the main result.

PPAM algorithm can not deal with reverberating signals at all. We can achieve very interesting result only pre-filtering the signal to cut out the reverberation. In other words, it seems that the Deleforge algorithm could work only at high SNR/DRR. That's why we will use only 3S-type input signals for all future experiments.

The next stage is to understand how the parameters can influence the output results. We executed 16 times the algorithm, changing the sampling frequency fs, the number of affine transformation k and choosing if we filter the signal with the ERB-filter or not. The setup used is called *xK_64(_bank)*, where $x = fs$ used and *_bank* contains the ERB-filter; 64 stands for the

fs	input	azimuth err.
8KHz	1S	29°
16KHz	1S	76°
8KHz	3S	3.51°
16KHz	3S	**3.18°**

Table 5.1: Result of the Direct-to-Reverberant filter used at different sampling frequencies.

window_length we used and this is kept fixed. Table 5.2 resumes all the measurement done.

One of the evident results is that the algorithm works well for $k > 16$ and it seems that the ERB-filter improves the system at $fs = 8KHz$, but it is not useful at $fs = 16KHz$. To really understand the behaviour of the algorithm, we need to go more in depth. We execute the algorithm 90 times, changing the same parameters, but focusing on the interval 16-32 for k and selecting randomly 3 different test set composed by 100 signal taken by the PRADO test set. Table 5.3 shows the main results.

Summarizing, we could reach the lowest azimuth error of $2.1° \pm 2°$ using $fs = 16KHz$, $k = 32$ without any filter. In the case we use the ERB-filter, we observed a non significant increase or the error ($2.3° \pm 2.6°$). With the experiment done until now, it seems that, despite our first assumptions, the introduction of the ERB-filter does not improve the system and that the best sampling frequency to use is $fs = 16KHz$.

Another important result is that we achieved this measurement using directly the azimuth and the elevation angles instead of the pixel position of azimuth and elevation. The PPAM algorithm can work also changing the 2D coordinate system used for the estimate of the sound source direction. We supposed that the linkage between the binaural world and the azimuth and elevation 2D world used by Deleforge can be extended also to other 2D world like our x and y coordinate or a polar coordinate system. The main consequence of this statement is that we can use this tool to map a sound source also in a 3D space, executing the same algorithm with 2 different orthogonal 2D coordinate systems. This task could be fundamental for developing a new localization algorithm, so we decided to focus our attention also to this feature. We executed the algorithm other 45 times, using the x and y coordinate in cm instead of the angle and taking the same 3 randomly chosen test signals dataset, and we compared the new results with the above ones. Table 5.4 shows the results obtained.

	k	azimuth err.
8K_64	4	7.5°
	8	6.9°
	16	$3.51° \pm 3.69°$
	32	$4.78° \pm 6.15°$
8K_64_bank	4	6.4°
	8	$4.33° \pm 4.15°$
	16	$3.41° \pm 2.68°$
	32	$3.93° \pm 4.41°$
16K_64	4	8.8°
	8	$3.43° \pm 2.80°$
	16	$3.18° \pm 3.08°$
	32	$\mathbf{2.06° \pm 2.13°}$
16K_64_bank	4	11°
	8	$5.07° \pm 4.54°$
	16	$3.56° \pm 3.11°$
	32	$4.06° \pm 4.69°$

Table 5.2: Inference of the dimension of the affine transformation used by PPAM on the mean angular direction error. The experiment was executed using different sampling frequencies (8KHz or 16KHz), different k values in the interval 4-32 and a window length of 64ms. The term bank *means that in that execution of the algorithm we used the ERB-filter. The best results obtained are in bold.*

Using the cartesian coordinate system, using the ERB-filter becomes fundamental to have the best possible precision. We had a new confirm that using $k = 32$ we can get the best localization performance, but, in this particular case, also $k = 24$ could be interesting: the results are a bit worse, but the training stage is faster at a lower k. Depending on the application, both options can be considered.

We tested also the response of the PPAM algorithm to another coordinate system: the polar system. We always used the same parameters and the same input signals. From the previous results, we decided to execute the algorithm using the configuration *16K_64_bank* ($fs = 16KHz$, $time\_window = 64$ and using the ERB-filter). After seen the results obtained in the previous experiments, we restricted the interval of k testing values used to 16, 24 and 32. Table 5.5 shows the results:

Also using this coordinate system, we could observe that the system works better using a bigger k value but, as in the other cases, the algorithm

revealed good localization performances also for the lower k testes. As we stated before, we can do the choice of this parameter in base of the goal of the final application and of the computational power available.

Using the same 3 test subsets of the test data, we tested also how much the length of the window can affect the mean localization error. To demonstrate so, we did other 18 execution of the algorithm, changing the parameters as describe in Table 5.6.

The localization performance behaved differently depending on the coordinate system. Using the cartesian one, with the longer window the results were a bit worse than before; instead using the polar coordinate system we could observe a lower error. A good parameter choice could be $time\_window = 96ms$, $fs = 16KHz$, $K = 32$ using the polar system.

The last test we would like to execute is about the relationship between the microphone distance and the localization error. Looking at the IPD feature vectors in Fig 5.16, we can suppose that a longer distance can help the PPAM algorithm in finding the best affine projection, so we were expecting better results using, for example, the MEMS couple 1-5 instead of the couple 1-2 that we used in all the previous experiments. We decided to use the polar coordinate system, we set $k = 32$ and we executed the algorithm both using a window length of 64ms and 96ms. We generated other 3 test subsets and then we averaged out the error we get. in Table 5.7 there are the main results.

Our suppositions were wrong. The choice of the microphones did not affect the algorithm, indeed the PPAM algorithm can find the best affine projection possible for each microphone distance available.

5.5 Final results for the improved algorithm

Here, we can resume all the results we got:

- We found out that the Deleforge code works well also using other coordinate systems. Deleforge used the PPAM task only to find out the source direction, but we can think to use this algebraic tool also to localize a point in a plane. For example, using the polar coordinate system, we can search for the distance and the angular position of the sound source, we can suppose the point where the source is.
- The main idea of the PPAM method is that the algorithm acts also on the frequency response of the room and not only to the direct component speech signal. Also the reverberating part of the source

signal due to the influence of the environment are used by the Deleforge algorithm to get the mapping between training and test set. The direction/position retrieved by the speech localization system is the result of a mapping between room responses.

- The Deleforge algorithm works better with training signals with a high Direct-to-Reverberant Ratio. It needs a training set composed by signal that have no reverberation nor long attacks. So, to execute this algorithm in a non-anechoic room, you need to introduce a prefiltering stage that must separate the direct sound to the reflections. We propose a simple spectral mask based on the estimate of the spectral power of the incoming sound, we do this filtering, we save 1 second of the clean sound and use it as a training signal. Only in this conditions we can have good performances, unless the PPAM algorithm could also not converge and the angular error could reach values of 70°

- The minimum dimension of the affine transformation to have good localization results is $k = 24$. The best results is achieved using $k = 32$, but the execution slows down.

- The ideal sampling frequency is the maximum achievable by our experimental setup: $fs = 16KHz$. If you have some restrictive conditions (e.g., limited power consumption or computational power), you can think to use $fs = 8KHz$. The localization performance are worse, but always less than 4°.

- The best $time\_window$ value is $96ms$. This length slows down the training stage of the system (in one case in a Matlab simulation it reaches the convergence of the PPAM after 10 minutes). $time\_window = 64ms$ could be the minimum threshold to have useful results, so we can chose a value between 64ms and 96 ms in base of the goal of the application you can build.

- The prefiltering stage introduced is fundamental. We improved the results thanks to the ERB-filter and the dereverberation stage we proposed. The improvement at $fs = 8KHz$ is dramatic, the impact at $fs = 16KHz$ is less evident, but is always present.

The lowest mean localization error we got using the optimal parameters we found out and all the new stage we introduced is:

- $\rho = 1.6cm \pm 1.7cm$ and $\alpha = 2.3° \pm 2.1°$ using the polar coordinate system

- $\alpha = 2.1^\circ \pm 1.9^\circ$ for the azimuth using an angular coordinate system
- $x = 4.5cm \pm 3.6cm$ and $y = 5.3cm \pm 5.9cm$ using the cartesian coordinate system

We could reach this conclusions after having executed 235 experiments acting each time on a different parameter, using different training and test set, comparing and averaging out the results obtained.

8K_64			
k	dataset 1	dataset 2	dataset 3
	azimuth err.		
16	$4.5° \pm 5°$	$4.3° \pm 4.6°$	$3.5° \pm 4.1°$
20	$6° \pm 6.6°$	$4.9° \pm 5.2°$	$3.9° \pm 3.9°$
24	$7° \pm 6°$	$6.1° \pm 5.5°$	$5.3° \pm 4.4°$
28	$5.3° \pm 6.6°$	$4.6° \pm 5.9°$	$4.2° \pm 6.3°$
32	$4.2° \pm 5.8°$	$3.2° \pm 4.8°$	$3.5° \pm 5.5°$

8K_64_bank			
k	dataset 1	dataset 2	dataset 3
	azimuth err.		
16	$3.3° \pm 3.2°$	$3.6° \pm 3.4°$	$3.5° \pm 2.6°$
20	$3.9° \pm 4.4°$	$3.6° \pm 5.2°$	$3.9° \pm 4.3°$
24	$3.7° \pm 5.1°$	$3.2° \pm 5.3°$	$3.8° \pm 5.3°$
28	$2.8° \pm 2.9°$	$\mathbf{2.5° \pm 2.5°}$	$3.1° \pm 3°$
32	$3.6° \pm 4.5°$	$3.9° \pm 5.5°$	$3.1° \pm 4.2°$

16K_64			
k	dataset 1	dataset 2	dataset 3
	azimuth err.		
16	$4.7° \pm 4.6°$	$3.4° \pm 2.9°$	$4.6° \pm 4.1°$
20	$3.3° \pm 2.5°$	$3.2° \pm 3°$	$2.9° \pm 2.5°$
24	$2.8° \pm 2.4°$	$3.2° \pm 3.4°$	$2.6° \pm 2.8°$
28	$3.8° \pm 3.3°$	$3.6° \pm 3.8°$	$3.1° \pm 2.9°$
32	$\mathbf{2.2° \pm 2.1°}$	$4.8° \pm 5.6°$	$\mathbf{2.1° \pm 2°}$

16K_64_bank			
k	dataset 1	dataset 2	dataset 3
	azimuth err.		
16	$2.8° \pm 2.7°$	$3.2° \pm 2.9°$	$2.5° \pm 2.4°$
20	$4.2° \pm 4.1°$	$3.7° \pm 3.7°$	$3.1° \pm 3.1°$
24	$2.9° \pm 3°$	$\mathbf{2.3° \pm 2.9°}$	$2.9° \pm 2.8°$
28	$3.5° \pm 4.6°$	$3.1° \pm 3.4°$	$3.2° \pm 4.3°$
32	$\mathbf{2.4° \pm 2.8°}$	$3.1° \pm 3.9°$	$\mathbf{2.3° \pm 2.6°}$

Table 5.3: A deeper analysis on the inference of the dimension of the affine transformation used by PPAM on the mean angular direction error. The experiment was executed using different sampling frequencies (8KHz or 16KHz), different k values in the interval 16-32 and a window length of 64ms. The term bank *means that in that execution of the algorithm we used the ERB-filter. The best results obtained are in bold.*

16K_64						
k	dataset 1		dataset 2		dataset 3	
	x err. (cm)	y err. (cm)	x err. (cm)	y err. (cm)	x err. (cm)	y err. (cm)
16	7.5 ± 6.1	9.3 ± 12.9	7.8 ± 6.5	8.7 ± 8.2	8.4 ± 8.9	8.5 ± 11.2
20	7.7 ± 6.3	7.6 ± 8	6.4 ± 5.9	8 ± 9.2	6.3 ± 5.4	6.5 ± 8.4
24	7.7 ± 7.4	6.2 ± 6.4	6.5 ± 6.9	9.6 ± 16.9	7.6 ± 8	6.1 ± 7.3
28	8.5 ± 7.4	8.1 ± 9.4	7.9 ± 6.9	10 ± 12.2	8.5 ± 9.5	7.7 ± 10
32	9.5 ± 19.8	9 ± 16.1	5.8 ± 4.1	8.2 ± 11.4	7.6 ± 17.2	8.1 ± 14.5
16K_64_bank						
k	dataset 1		dataset 2		dataset 3	
	x err. (cm)	y err. (cm)	x err. (cm)	y err. (cm)	x err. (cm)	y err. (cm)
16	6.3 ± 5.4	7.1 ± 9.5	7.2 ± 6.1	9.9 ± 13.3	5.8 ± 5.4	6.3 ± 8.3
20	8.1 ± 6.6	9.5 ± 10.7	6.5 ± 5.4	9.7 ± 11.9	6.5 ± 5.1	7.8 ± 9.3
24	6.8 ± 4.9	7.4 ± 10.6	6.3 ± 7.6	6.3 ± 10.3	5.8 ± 4.4	7.2 ± 10.1
28	10 ± 14.4	7.4 ± 10.4	7.1 ± 9.5	5.8 ± 8	8.2 ± 13.1	6.5 ± 9.5
32	$\mathbf{4.7 \pm 4.2}$	$\mathbf{5.7 \pm 9.1}$	$\mathbf{3.7 \pm 2.8}$	$\mathbf{5.7 \pm 8.7}$	$\mathbf{4.8 \pm 4.2}$	$\mathbf{5.7 \pm 9.3}$

Table 5.4: Inference of the choice of the cartesian coordinate system and of the parameter k of the PPAM on the mean localization error. The term bank *means that in that execution of the algorithm we used the ERB-filter. The best results obtained are in bold.*

k	dataset 1		dataset 2		dataset 3	
	ρ err. (cm)	θ err. (°)	ρ err. (cm)	θ err. (°)	ρ err. (cm)	θ err. (°)
16	1.6 ± 1.4	3 ± 2.5	$\mathbf{1.3 \pm 1.9}$	$\mathbf{2.2 \pm 2.3}$	1.8 ± 1.6	3.2 ± 2.9
24	1.4 ± 1.4	3.3 ± 3.6	2.1 ± 2	3 ± 3.7	1.7 ± 1.5	3.3 ± 3.4
32	$\mathbf{1.4 \pm 1.3}$	$\mathbf{2.6 \pm 2.3}$	2 ± 1.8	2.7 ± 2.9	$\mathbf{1.6 \pm 1.4}$	$\mathbf{2.2 \pm 2.2}$

Table 5.5: Inference of the choice of the polar coordinate system and of the parameter k of the PPAM on the mean localization error. The best results obtained are in bold.

16K_96_bank						
k	dataset 1		dataset 2		dataset 3	
	x err. (cm)	y err. (cm)	x err. (cm)	y err. (cm)	x err. (cm)	y err. (cm)
16	12 ± 11	8.5 ± 9.4	9.2 ± 9.9	6.6 ± 6.4	9.4 ± 10.1	8.4 ± 8.6
24	$\mathbf{4.5 \pm 3.7}$	5.8 ± 6.6	5.2 ± 4.1	7.9 ± 8	$\mathbf{4.5 \pm 3.3}$	5.6 ± 7.2
32	4.8 ± 4.1	$\mathbf{5.4 \pm 5.9}$	$\mathbf{4.5 \pm 3.6}$	$\mathbf{5.3 \pm 5.9}$	$\mathbf{4.4 \pm 3.7}$	$\mathbf{4.7 \pm 5.2}$
polar_16K_96_bank						
k	dataset 1		dataset 2		dataset 3	
	ρ err. (cm)	θ err. (°)	ρ err. (cm)	θ err. (°)	ρ err. (cm)	θ err. (°)
16	2.7 ± 3.1	8 ± 9.1	2.8 ± 2.8	7.4 ± 7.7	2.5 ± 2.7	7 ± 7.9
24	$\mathbf{1.5 \pm 1.3}$	$\mathbf{2.7 \pm 2.4}$	$\mathbf{2.1 \pm 1.8}$	3.3 ± 2.4	$\mathbf{1.7 \pm 1.6}$	2.6 ± 2.3
32	$\mathbf{1.4 \pm 1.6}$	$\mathbf{2.7 \pm 2.6}$	$\mathbf{2.1 \pm 2.2}$	$\mathbf{2.3 \pm 2.2}$	$\mathbf{1.6 \pm 1.7}$	$\mathbf{2.3 \pm 2.1}$

Table 5.6: Inference of the choice of the coordinate system and of the parameter k of the PPAM on the mean localization error using a window length of 96ms instead of 64ms. In 16K_96_bank we used a cartesian coordinate system, in polar_16K_96_bank we used a polar coordinate system. The best results obtained are in bold.

mic. couple	$time\_window = 64ms$		$time\_window = 96ms$	
	ρ err. (cm)	θ err. (°)	ρ err. (cm)	θ err. (°)
2-1	$\mathbf{1.6 \pm 1.4}$	$\mathbf{2.2 \pm 2.2}$	$\mathbf{1.6 \pm 1.7}$	$\mathbf{2.3 \pm 2.1}$
3-1	1.8 ± 1.5	3.4 ± 3.2	$\mathbf{1.6 \pm 1.7}$	2.8 ± 2.4
4-1	1.8 ± 1.7	$\mathbf{2.4 \pm 2.1}$	1.9 ± 1.8	$\mathbf{2.1 \pm 1.9}$
5-1	1.8 ± 1.5	3.2 ± 2.9	1.9 ± 1.6	2.6 ± 2.9

Table 5.7: Inference of the choice of the couple of microphones and of the window length on the localization error. The best results are in bold.

Chapter 6

Conclusions and further development

The initial goal of this research was to do a deep introduction into the field of speech source localization problem for robotics, starting from the state-of-the-art algorithms and from the theoretical knowledge required to have a full comprehension of this task. This work is thought as a first approach to this world, so that a future new branch of research can rise up in our laboratory. We wish we obtained an articulated thesis in which we presented both the theoretical and the experimental processes based mainly on digital signal processing, physics, maths and psychoacoustics.

6.1 Final considerations

In Chapter 2 we described the most important and complete examples of the already existing algorithms that try to solve the problem of sound localization for robots. We did a deep analysis, so to understand which knowledge is required to work in this field of research and we showed the main theoretical basics we need to study to work on this topic.

Then, in Chapter 3, we focused our attention on the two most convincing works we found: [15] and [45]. We analyzed step-by-step the two algorithms, in order to reveal which ideas their authors used to create a working sound source localization system. There are some important differences between these two algorithms; we studied their pros and cons and we proposed ways to improve them.

Chapters 4 and 5 described the experimental results we obtained from

more than 300 executions of the systems, thanks also to the complete recording audio setup. First, we simulated the original code with the datasets used by the original authors and we tested our preliminary hypothesis taking the algorithms as they are. Then we modified them, testing a new dereverberation filtering stage, some filters, among which an ERB-filter and different types of band pass filters. We tested all the parameters used by the original authors (e.g., the sampling frequency, the windowing parameters, the signal length) and we showed all the results we obtained. With the improvements we proposed in Chapter 4, we reduced the localization error of the 5% in azimuth and of the 27% in the elevation.

We also built a completely new audio database, composed by white noise signal and different kind of speech signals, recorded in a real reverberant room using 5 MEMS microphones and 2 different speakers in different positions. We used them to reconstruct in a real environment the experiments and validate our first assumptions. The setup and the parameters we proposed in Chapter 5 allowed the algorithm to work in a realistic world with the minimum azimuth localization error of about 2°, with some important strong conclusions on the signal processing used, on the algebraic tools and on some physical and geometrical assumptions that could have a general validity.

In other words, we can think about this work as a stand-alone research, in which starting from zero we took the state-of-the-art algorithms and we proposed an assessment and an improvement, but these ideas can be also the starting point for new future works. We can think to optimize and implement this system in a robot or we can go more in depth on studying the inference between the localization task and the environment. Our results on the preprocessing stage can be used as a starting scheme for a new localization algorithm, maybe also for non-speech signals or for multiple speakers. The PRADO database we built up and the setup available in the laboratory could be used for any kind of audio signal processing work (e.g., localization, speech recognition, room analysis).

In the next section we proposed one of the possible future development of this research. We used the results we got in Chapter 5 to implement and improve the Willert algorithm. The ideas applied to improve the performance of the Deleforge algorithm allowed us to implement the Willert algorithm with some important and interesting results. The algorithm we proposed has worse result than the one proposed by the INRIA group, but it is easily portable, has a simpler training stage that is not room-dependent and uses a very simple algebraic tool (basically a distance measure between matrices). Some future research could be done studying more powerful clustering

systems that can substitute the distance measure we selected.

6.2 An example of further development: the improvement of the Willert algorithm

As we described deeply in Chapter 3, the authors describe a sound localization system based on some strong biological and physical studies. They propose an algorithm based on two interaural cues, the *ILD* and the *ITD*, calculated through a cochlear model designed to emulate the human auditory system. Their goal is to achieve good localization performances building a simple mathematical model of the human perception system.

Our proposal is to start from their work, developing a Matlab implementation of the algorithm they described and we would like to compare the result we can obtain with the ones listed in the paper. The code will be tested using the experimental setup and the PRADO dataset (see Chapter 4 and 5), both the training and the test signals, so we can check for the stability and for the reliability of this system also in a realistic reverberating chamber. Then we would like to improve it, using all the results we got from the whole previous analysis and experiments we did on the Deleforge algorithm. We add both the prefiltering stage, the ERB-filter, we test the impact of the parameters on the system (e.g., sampling frequency, feature vector dimension, window parameters). Finally, we analyze the final result, that represent our sound source algorithm proposal, in which we put together all the researches done previously.

First, we focused our attention on the original probabilistic model proposed in [45]. We followed strictly the parameters and the maths definition the researchers used in the paper to execute the code in our room with our setup. The measurement they proposed is based on a simple geometrical consideration, described by Fig 6.1. The source direction is linked to a time mismatch between the perceived signal by the two microphones. This is the basilar definition of ITD.

Remembering the room characteristics in which we created the PRADO dataset, we did other basilar hypothesis:

- We could do a simple rough estimation of the difference in time between the direct sound and the first reflection. The microphones array was positioned in the center of the room with the loudspeaker at 2m. The distance between one MEMS and the nearer wall was at least 3m. supposing to emit in front of the array, the direct path was 2m instead the first reflection path was long about 6.3m with a simple geometric

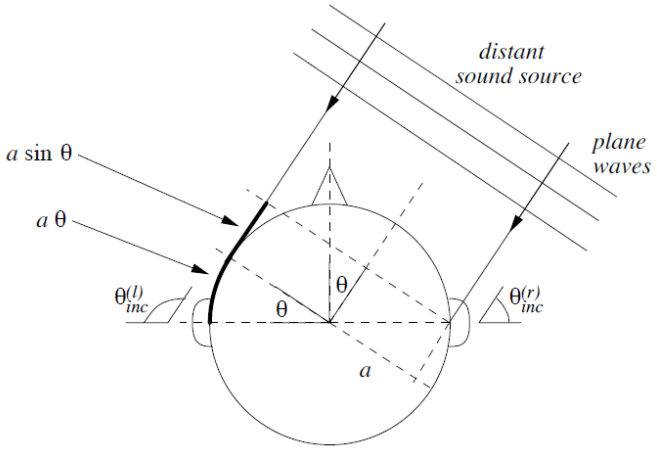

Figure 6.1: The geometrical basics of the Willert system

calculation. So, the time difference due to the different path length was about 13ms. If we would like to work separately with direct sound and late reverberation, we need to select the signal processing parameter so that we can observe a minimum useful amount of sampled data in each window. In our case, using our standard $fs = 16KHz$ and $time\_window = 96ms$ with an overlap factor of 1/8, the shift between each window is long 192 samples and the 13ms signal corresponds to 208 samples. This is a very rough calculation, but it is useful to understand that our framework using our parameters could have a physical meaning.

- For the same geometrical assumption and using the formulation described in Chapter 2 and 3, we calculated the hypothetic localization precision of the algorithm. Using Eq 3.10, with $d = 15cm$ and $c = 343m/s$, the minimum angular resolution achievable for separating two frontal sound source is 8°

- As described in the paper and resumed in Chapter 3, the main algebraic tool used here is the Cross Correlation to measure the distance between the incoming signal and a training set to estimate the direction. We proposed how to calculate it and we need to think also to a distance measure used to understand how much the measured test sound is near the training ones.

6.2.1 1st implementation, analysis and results

We started developing a first implementation of the algorithm, to understand deeply its original parameters and formulation, remembering the preliminary hypothesis we described above. The mathematical details were defined in Chapter 3. We decided to use the ideal parameters which gave the best results in Chapter 5, we set the dimension of the patch to $11x11$, we decided to respect the *Duplex theory*, so we set as frequency threshold the value $f = 1500Kz$ and, for our simple application, we take only the ITD as feature vector. For the training set, we used 19 signals taken from the so-called PRADO 3S training dataset. The selection is done so to have a train signal about every 10°.

The first approach failed. The maths definition described in the article were in some case wrong. We found out some error in their definitions that cause some Matlab dimension mismatch errors. The matrix and the point-wise multiplication could not work because the matrix dimensions were not matching. In particular, the parameters μ and σ they defined and used in Eq 3.13 and Eq 3.14 were wrong. Initially, we though that they were defining some particular parameters, but, after having noticed this problem, we deduced from that the μ and σ they would like to use were the standard definition of *mean* and *variance* functions. So, we changed those equations, using the standard function already available in the Matlab framework for their calculation.

With this correction, we executed the training stage of the algorithm and obtain the training set. You can find the Matlab script used in Appendix B.

The code works in the following way:

- It loads the training sounds and the ground truth, all the parameters required (sampling frequency, window parameters, FFT length) and the gaussian patch.
- Each incoming sound is read, resampled, zero-padded, passed through the ERB-filter and then the STFT is calculated.
- The frequency threshold required by the Duplex theory is used to filter out the above calculated left and right STFT and, for each of them, the gaussian patch is applied.
- The *STFT_patch* function calculates all the parameters useful to calculate in the next step the ITD: the patches signal X, μ and σ. X is a 4D matrix whose dimension in our application is $(11, 11, f, t)$, where f

is the number of useful frequency bin (it depends on the Duplex theory frequency threshold) and t is the number of time-frames that compose the signal (basically it's the number of windows given the length of the signal). We put in a unique multidimensional matrix the values of the patched signal for each frequency bin and for each time-step. Using the same definitions, μ and σ are (f, t) matrix. They represent the mean and the variance of all the patched signal at each bin and frame.

- The final ITD is calculated via a cross-correlation function which take as input all the left and the right patches available and saved in a *.mat* file. The ITD is a (f, t_1, t_2) matrix: each point represents the ITD calculated at a frequency bin, correlating the left patch at the time-step t_1 with the right patch at the time-step t_2

Once the training stage is executed, the saved ITD compose the training set required by the algorithm. They will be always valid for each new future unknown signal in each kind of room, so, this stage must be executed only once. This is the most powerful point of this algorithm: it's not room-dependent, so it can work anywhere after the first calibration.

The test stage works in a similar way. The ITD is calculated with the same functions described above and then a distance measurement is executed to estimate the sound source position. This is the core of the algorithm: it loads the training set and, for each time-frame of the test signal it calculates the distance between the test ITD and each of the training one. For our first implementation we used the available Matlab function *pdist2* that simply calculates the Euclidean distance between 2 functions. We extracted the test set from the PRADO test dataset, selecting 100 signals randomly as we did in the previous experiments.

Surprisingly ,the result we got with any other kind of processing and using this measuring system were very bad: the average localization error reaches the values of $47.1° \pm 36.3°$.

We were expecting worse result than the Deleforge algorithm, but we obtained a very poor result. The problem was the dimension of the data and how these matrix were built. Always in the Matlab framework we can check that the ITD matrix we defined following the Willert instruction are highly sparse an are badly conditioned. This creates important computational problem; some Matlab basilar function that involves matrix multiplication or inversion could not converge if the input matrix are not well conditioned or their integrity is not more guaranteed, so there is no more certainty about the correctness of the data. Another problem for us is that the original

code was not available. The paper they published contains also a complete simulation and an analysis on the result they found out, but there is not any relationship to how they implement the system. Seeing their results and the description of the algorithm, we started working on this believing that an easy Matlab implementation could be feasible; we would like also to compare our results with them, but how can we do it if we do not know how they got it? Nevertheless, we would try to change some definitions and modify this model to really understand its strength.

6.2.2 The final new proposal

We remember that the aim of this part of the research is to propose a localization algorithm that is easy to understand, not computational heavy and that have a localization error sufficiently low. We were supposing now that a resolution similar to the one the INRIA system found out (about 2°) is not achievable without any kind of complex algebraic tool. The system we were proposing is composed by a training set whose resolution is 10°, so our goal could be a final average error comparable with this number.

In Appendix B we put also our last proposal. We changed a bit how to store the data and, in particular, we tested different measurement system for the distance we need to measure between the test ITD and the training set. We selected always randomly a subgroup of 100 test sound from the PRADO dataset and we calculated their ITD with the Willert method. Instead of using the *pdist2* function, we calculated directly the pointwise distance between each matrix using the Euclidean, the Mahalanobis and the Canberra distance and, for each of them we did the same test with or without normalizing the measurement for each time frame. The Mahalanobis distance is defined as:

$$Mahalanobis = \sqrt{(X-Y)^T * \Sigma^{-1} * (X-Y)}. \tag{6.1}$$

It's based on the correlation between the variables thanks to that different pattern can be analyzed and identified. In other words, it calculate the similarity between a training space and an unknown one, accounting also for the correlation between the data (called Σ in Eq 6.1). The Canberra distance is defined as:

$$Canberra = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|}. \tag{6.2}$$

This equation defines a distance only if all the input are positive (and this is our situation). The advantage of this distance measure is that it is only a bit sensitive to not symmetrically distributed variables and to outliers.

Distance measure	Normalization	Azimuth error (°)
Euclidean	no	47.1 ± 36.3
	yes	$\mathbf{42.5 \pm 26.6}$
Mahalanobis	no	52 ± 29.4
	yes	43.2 ± 25
Canberra	no	44.9 ± 33.4
	yes	$\mathbf{41 \pm 27.2}$

Table 6.1: Inference of the distance measure and of the usage of a normalization factor in the measurement on the average localization error.

Distance measure	Normalization	Azimuth error (°)
Euclidean	no	28 ± 20
	yes	23.3 ± 15.4
Mahalanobis	no	23.2 ± 17.2
	yes	$\mathbf{22.4 \pm 16}$
Canberra	no	26.9 ± 20.6
	yes	24.4 ± 14.7

Table 6.2: Inference of the distance measure and of the usage of a normalization factor in the measurement on the average localization error. We focused the experiment only on the speech sources located in front of the listener, in a range between $50°$ *and* $130°$.

Table 6.1 shows the average localization error we get.

We do also other experiments focusing only to the 50 sound emitted in front of the microphones array, between 50° and 130°. Table 6.2 are the results

The localization error went down dramatically for these test sound. This is due to the binaural cue we chose and we were expecting this kind of result. Using only the ITD can cause some problem for sources emitting on the side of the listening object (as described in Chapter 2). In these points, the time-delay resolution of the ITD cue is not more sufficient and the approximation used to calculate the ITD is not more valid. That's why ITD is not more useful and this is the reason why we got a high average error for the whole test set we used instead the precision concerning only the frontal sources doubles.

Bibliography

[1] Xavier Alameda-Pineda and Radu Horaud. A geometric approach to sound source localization from time-delay estimates. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22(6):1082–1095, 2014.

[2] Xavier Alameda-Pineda and Radu Horaud. Vision-guided robot hearing. *The International Journal of Robotics Research*, page 0278364914548050, 2014.

[3] Murat Aytekin, Cynthia F Moss, and Jonathan Z Simon. A sensorimotor approach to sound localization. *Neural Computation*, 20(3):603–635, 2008.

[4] Glen Ballou. *Handbook for sound engineers*. Taylor & Francis, 2008.

[5] Jon Barker and Xu Shao. Energetic and informational masking effects in an audiovisual speech recognition system. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(3):446–458, 2009.

[6] Siouar Bensaid, Antony Schutz, and Dirk TM Slock. Single microphone blind audio source separation using em-kalman filter and short+ long term ar modeling. In *Latent Variable Analysis and Signal Separation*, pages 106–113. Springer, 2010.

[7] Patricia Besson, Vlad Popovici, Jean-Marc Vesin, Jean-Philippe Thiran, and Murat Kunt. Extraction of audio features specific to speech production for multimodal speaker detection. *Multimedia, IEEE Transactions on*, 10(1):63–73, 2008.

[8] David T. Blackstock. *Fundamentals of Physical Acoustics*. Wiley-Interscience, July 2000.

[9] Michael S Brandstein and Harvey F Silverman. A robust method for speech signal time-delay estimation in reverberant rooms. In *Acoustics,*

Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on, volume 1, pages 375–378. IEEE, 1997.

[10] Christophe Couvreur. octave, 1997.

[11] Antoine Deleforge, Florence Forbes, and Radu Horaud. Variational em for binaural sound-source separation and localization. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 76–80. IEEE, 2013.

[12] Antoine Deleforge, Florence Forbes, and Radu Horaud. Acoustic space learning for sound-source separation and localization on binaural manifolds. *International journal of neural systems*, 25(01):1440003, 2015.

[13] Antoine Deleforge and Radu Horaud. Learning the direction of a sound source using head motions and spectral features. Technical report, INRIA - INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 2011.

[14] Antoine Deleforge and Radu Horaud. The cocktail party robot: Sound source separation and localisation with an active binaural head. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 431–438. ACM, 2012.

[15] Antoine Deleforge, Radu Horaud, Yoav Y Schechner, and Laurent Girin. Co-localization of audio sources in images using binaural features and locally-linear regression. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(4):718–731, 2015.

[16] M Ferras. Multi-microphone signal processing for automatic speech recognition in meeting rooms. Master's thesis, ICSI, Berkeley, California, 2005.

[17] Bradford W Gillespie and Les E Atlas. Strategies for improving audible quality and speech recognition accuracy of reverberant speech. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–676. IEEE, 2003.

[18] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Element of Statistical Learning*. Spring, 2008.

[19] Simon Haykin and Zhe Chen. The cocktail party problem. *Neural computation*, 17(9):1875–1902, 2005.

[20] Simon Haykin and Michael Moher. *Introduzione alle telecomunicazioni analogiche e digitali*. Casa Editrice Ambrosiana, 2007.

[21] Martin Heckmann, Tobias Rodemann, Frank Joublin, Christian Goerick, and Bjorn Scholling. Auditory inspired binaural robust sound source localization in echoic and noisy environments. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 368–373. IEEE, 2006.

[22] Fakheredine Keyrouz. *Efficient Binaural Sound Localization for Humanoid Robots and Telepresence Applications*. PhD thesis, TECHNISCHE UNIVERSITÄT MÜNCHEN, 2008.

[23] Fakheredine Keyrouz. Advanced binaural sound localization in 3-d for humanoid robots. *Instrumentation and Measurement, IEEE Transactions on*, 63(9):2098–2107, 2014.

[24] Fakheredine Keyrouz and Klaus Diepold. An enhanced binaural 3d sound localization algorithm. In *Signal Processing and Information Technology, 2006 IEEE International Symposium on*, pages 662–665. IEEE, 2006.

[25] Fakheredine Keyrouz, Werner Maier, and Klaus Diepold. Robotic localization and separation of concurrent sound sources using self-splitting competitive learning. In *Computational Intelligence in Image and Signal Processing, 2007. CIISP 2007. IEEE Symposium on*, pages 340–345. IEEE, 2007.

[26] Hyun-Don Kim, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Binaural active audition for humanoid robots to localise speech over entire azimuth range. *Applied Bionics and Biomechanics*, 6(3-4):355–367, 2009.

[27] Charles H Knapp and G Clifford Carter. The generalized correlation method for estimation of time delay. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 24(4):320–327, 1976.

[28] Jonathan Le Roux and Emmanuel Vincent. A categorization of robust speech processing datasets. Technical report, Mitsubishi Electric Research Labs, 2014.

[29] Jonathan Le Roux, Emmanuel Vincent, John R Hershey, and Daniel PW Ellis. Micbots: collecting large realistic datasets for speech and audio research using mobile robots. In *Acoustics, Speech and Signal*

Processing (ICASSP), 2015 IEEE International Conference on, pages 5635–5639. IEEE, 2015.

[30] Soo-Yeon Lee and Hyung-Min Park. Multiple reverberant sound localization based on rigorous zero-crossing-based itd selection. *Signal Processing Letters, IEEE*, 17(7):671–674, 2010.

[31] Xiaofei Li, Laurent Girin, Radu Horaud, and Sharon Gannot. Binaural sound source localization based on direct-path relative transfer function. *arXiv preprint arXiv:1509.03205*, 2015.

[32] Yan-Chen Lu and Martin Cooke. Binaural estimation of sound source distance via the direct-to-reverberant energy ratio for static and moving sources. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(7):1793–1805, 2010.

[33] Ewan A Macpherson and John C Middlebrooks. Listener weighting of cues for lateral angle: the duplex theory of sound localization revisited. *The Journal of the Acoustical Society of America*, 111(5):2219–2236, 2002.

[34] Maria Irene Miranda. Clustering methods and algorithms, 1999.

[35] Sanjit K. Mitra. *Digital Signal Processing - A computer based approach.* McGraw-Hill, 2008.

[36] Joan Mouba and Sylvain Marchand. A source localization/separation/respatialization system based on unsupervised classification of interaural cues. In *Proceedings of the Digital Audio Effects (DAFx06) Conference*, pages 233–238, 2006.

[37] Kazuhiro Nakadai, Tino Lourens, Hiroshi G Okuno, and Hiroaki Kitano. Active audition for humanoid. In *AAAI/IAAI*, pages 832–839, 2000.

[38] Kazuhiro Nakadai, Daisuke Matsuura, Hiroshi G Okuno, and Hiroshi Tsujino. Improvement of recognition of simultaneous speech signals using av integration and scattering theory for humanoid robots. *Speech Communication*, 44(1):97–112, 2004.

[39] Hiromichi Nakashima and Toshiharu Mukai. 3d sound source localization system based on learning of binaural hearing. In *Systems, Man and Cybernetics, 2005 IEEE International Conference on*, volume 4, pages 3534–3539. IEEE, 2005.

[40] Martin Raspaud, Harald Viste, and Gianpaolo Evangelista. Binaural source localization by joint estimation of ild and itd. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(1):68–77, 2010.

[41] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of Biometrics*, pages 827–832, 2015.

[42] Scott Rickard and Özgür Yilmaz. On the approximate w-disjoint orthogonality of speech. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–529. IEEE, 2002.

[43] Andreas Schwarz and Walter Kellermann. Coherent-to-diffuse power ratio estimation for dereverberation. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(6):1006–1018, 2015.

[44] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. Performance measurement in blind audio source separation. *Audio, Speech, and Language Processing, IEEE Transactions on*, 14(4):1462–1469, 2006.

[45] Volker Willert, Julian Eggert, Jürgen Adamy, Raphael Stahl, and Edgar Körner. A probabilistic model for binaural sound localization. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(5):982–994, 2006.

List of Figures

List of Tables

Appendix A

Deleforge code

Here, you can find the original and the modified Matlab codes used in all the experiments linked to the Deleforge work.

A.1 Original code

These are the main scripts used by Deleforge in [12] and [15]. The details about the meaning of each variable and function are analyzed in Chapter 3 and 4.

A.1.1 EXAMPLE_ training1source

```
%%%%%%%%%%%%%%%%%%%%%%%%%% Training 1 Source %%%%%%%%%%%%%%%%%%
%%% Author: Antoine Deleforge (March 2015) -
%% deleforge.antoine@gmail.com %%
% This file shows how to train a single-source sound-localizer
% with annotated white-noise recordings
%
% It uses the algorithm PPAM as presented in:
% Deleforge, A., Forbes, F., & Horaud, R. (2014). "Acoustic space
%     learning for sound-source separation and localization on
%     binaural manifolds". International Journal of Neural Systems.
%
% This code has been succesfully tested with the following datasets:
%CAMIL v1.1 (http://perception.inrialpes.fr/~Deleforge/CAMIL_Dataset/)
%AVASM v1.0 (http://perception.inrialpes.fr/~Deleforge/AVASM_Dataset/)
% The example below is tuned for the AVASM training dataset v1.0,
% but can be used with CAMIL up to minor adaptations.
%
% WARNING: this code requires the GLLiM matlab toolbox to work.
% This toolbox is available at
```

```
% https://team.inria.fr/perception/gllim_toolbox/

verb = 1; % Level of output displays

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PATHS TO DATA %%%%%%%%%%%%%%%%%%%%
% Path to the AVASM dataset:
PATH = 'the_AVASM_dataset/';

% Code name of the project (set by user):
proj_name='AVASM_1S';

% Path where the training data will be saved (set by user):
DATA_PATH=[PATH 'Data/'];
if(~exist(DATA_PATH,'dir')); mkdir(DATA_PATH); end;

% Path to white-noise sounds:
WN_SOUND_PATH=[PATH,'annotated_white_noise/Recorded/'];

% Path to the white-noise sound positions file:
WN_POS_FILE= [PATH,'annotated_white_noise/pixel_positions'];

%%%%%%%%%%%%%%%%%%%% Create white-noise ILD and IPD data %%%%%%%%
%%% Spectrogram parameters:
fs=16000; % Frequency of sampling used (in Hz)
time_window=64; % Spectrogram windows (ms)
overlap=87.5; % Spectrogram % overlap
%%% Create ILD and IPD data from white noise recordings:
createInterauralData(WN_SOUND_PATH,DATA_PATH,proj_name,...
                     time_window,overlap,fs,verb);

%%%%%%%%%%%%%%%%%%% TRAINING USING WHITE-NOISE AND PPAM %%%%%%%%
%%% Load white-noise ILD and IPD data:
s = load([DATA_PATH,proj_name,'_ILD']);
ILD_data = s.ILD_data;
s = load([DATA_PATH,proj_name,'_IPD']);
IPD_data = s.IPD_data;
freq_idx_ILD = 1:513; % Keep all frequencies for ILD
freq_idx_IPD = 2:513; % Remove f=0 for IPD because it is null
ILD_data=ILD_data(freq_idx_ILD,:);
IPD_data=IPD_data(freq_idx_IPD,:);
% Interaural training data:
Y_data=[ILD_data;real(exp(1i*IPD_data));imag(exp(1i*IPD_data))];

%%% Load source position data: (Ground truth data)
wn_position = ReadMatrixData(WN_POS_FILE);

%%% Train the sound-localizer using PPAM and white-noise ILPD data:
x=wn_position(2:3,:);
y=Y_data(:,wn_position(1,:));
```

```
Maxiter = 30; % Maximum number of iterations
% Multiscale training:
for i=0:5
    K = 2^i; % Number of piecewise-affine components
    [Theta,~] = ppam(x,y,K,Maxiter,verb);
    % Save learned parameters:
    parameter_file = [DATA_PATH,proj_name,'_Theta_ILPD_K',...
                      int2str(K)];
    save([parameter_file,'.mat'],'Theta');
end
```

A.1.2 EXAMPLE_ test1source

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%% Test 1 Source %%%%%%%%%%%%%%%%%%%%%%%
%%% Author: Antoine Deleforge (March 2015) -
%% deleforge.antoine@gmail.com %%
% This script shows how to use a trained single-source sound-
% localizer to localize individual speech recordings.
%
% It uses the algorithm PPAM as presented in:
% Deleforge, A., Forbes, F., & Horaud, R. (2015). "Acoustic
%   space learning for sound-source separation and localization
%   on binaural manifolds". International Journal of Neural Systems.
%
% This code has been succesfully tested with the following datasets:
%CAMIL v1.1 (http://perception.inrialpes.fr/~Deleforge/CAMIL_Dataset/)
%AVASM v1.0 (http://perception.inrialpes.fr/~Deleforge/AVASM_Dataset/)
% The example below is tuned for the AVASM training dataset v1.0,
% but can be used with CAMIL up to minor adaptations.
%
% WARNING: this code requires the GLLiM matlab toolbox to work.
% This toolbox is available at
% https://team.inria.fr/perception/gllim_toolbox/

verb = 1; % Level of output displays

%%%%%%%%%%%%%%%%%%%%%%%%%%%% PATHS TO DATA %%%%%%%%%%%%%%%%%%%%%%
% Path to the AVASM dataset:
PATH = 'the_AVASM_dataset/';

% Code name of the project (set by user):
proj_name='AVASM_1S';

% Path to training data:
DATA_PATH=[PATH 'Data/'];

% Path to test sounds:
```

```
TEST_SOUND_PATH=[PATH,'annotated_speech/Recorded/'];

% Path to the ground truth positions of test sounds:
TEST_POS_FILE= [PATH,'annotated_speech/pixel_positions'];


%%%%%%%%%%%%%%%%%%% SINGLE SOURCE LOCALIZATION TEST %%%%%%%%%%
% Parameters:
fs=16000; % Frequency of sampling used (in Hz)
time_window=64; % Spectrogram windows (ms)
overlap=87.5; % Spectrogram % overlap
K = 32; % Number of piecewise-affine components
Ntest = 20; % Number of sounds tested

freq_idx_ILD = 1:513; % Keep all frequencies for ILD
freq_idx_IPD = 2:513; % Remove f=0 for IPD because it is null

% Load training data and ground truth positions:
test_position = ReadMatrixData(TEST_POS_FILE);
parameter_file = [DATA_PATH,proj_name,'_Theta_ILPD_K',...
                    int2str(K)];
load([parameter_file,'.mat']); % Theta

er_az=zeros(Ntest,1);
er_el=zeros(Ntest,1);
for t=1:Ntest;
    %%% pick a test sound at random:
    posidx=randi(size(test_position,2));
    sound_nr = test_position(1,posidx);

    true_az = test_position(2,posidx);
    true_el = test_position(3,posidx);

    [sound,rfs] = wavread([TEST_SOUND_PATH,int2str(sound_nr)]);

    fprintf(1,'Localization test %d (Sound %d.wav at ...
            (%2.2f,%2.2f):\n',t,posidx,true_az,true_el);

    %%% Compute interaural spectrograms:
    mic=[1,2]; % Pair of microphones used
    threshold=-20; % Threshold for missing data (in dB)

    % Resample the sound and select microphones:
    sound=[resample(sound(:,mic(1)),fs/gcd(rfs,fs),...
            rfs/gcd(rfs,fs)),resample(sound(:,mic(2)),...
            fs/gcd(rfs,fs),rfs/gcd(rfs,fs))];

    [~,~,alpha,phi,Chi] = ...
             binaural_spectrograms(sound,fs,time_window,...
```

```
            overlap,threshold);

    phi=phi(freq_idx_IPD,:);    % Interaural spectrogram
    alpha=alpha(freq_idx_ILD,:);
    Y=[alpha;real(exp(1i*phi));imag(exp(1i*phi))];
    % Indicator of missing data:
    Chi=[Chi(freq_idx_ILD,:);Chi(freq_idx_IPD,:);...
            Chi(freq_idx_IPD,:)];

    %%% Spectrogram inversion using the mapping parameters Theta:
    xmap = ppam_spectrogram_inverse(Y,Chi,Theta,verb); % 2x1

    %%% Computer errors:
    er_az(t) = abs(xmap(1)-true_az);
    er_el(t) = abs(xmap(2)-true_el);
    fprintf(1,'Single-source localization error (Pixels): ...
            az = %2.2f, el = %2.2f\n\n', er_az(t),er_el(t));
end
fprintf(1,'\n==== Average Localization Errors (In Pixels) ====\n');
fprintf(1,'azimuth = %2.2f +- %2.2f, el = %2.2f +- %2.2f\n',...
          mean(er_az),std(er_az),mean(er_el),std(er_el));
```

The function called *createInterauralData* loads and reads all the training sounds, it resamples them, it builds the binaural cues through the fucntion *binaural_ spectrograms*and it saves them in *.mat* files. The function called *ppam* implements the calculation of the PPAM training parameters. The function called *ppam_ spectrogram_ inverse* computes the affine transformation starting from the training parameters. The complete toolbox and their datasets are freely available at the following website: https://team.inria.fr/perception/research/binaural-ssl/.

A.2 Modified code

In this section you can find the final proposal we created for the Deleforge algorithm.

A.2.1 PRADO_ dataset_ training1source

```
clear all
close all
clc
addpath('octave');
addpath('the_GLLiM_toolbox\');

verb = 1; % Level of output displays
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PATHS TO DATA %%%%%%%%%%%%%%%%%%%
% Path to the AVASM dataset:
PATH = 'PRADO dataset/';

% Code name of the project (set by user):
proj_name='polar_PRADO_3S_16k_96_bank_newGT_newILPD';

% Path where the training data will be saved (set by user):
DATA_PATH=[PATH 'Data/'];
if(~exist(DATA_PATH,'dir')); mkdir(DATA_PATH); end;

% Path to white-noise sounds:
WN_SOUND_PATH=[PATH,'training_sound_3/'];

%%% Load source position data: (Ground truth data)
load([PATH,'training_genelec.mat']);
wn_position=pos_gen';
load([PATH,'mic.mat']);

%%%%%%%%%%%%%%%%%%%% Create white-noise ILD and IPD data %%%%%%%
%%% Spectrogram parameters:
fs=16000; % Frequency of sampling used (in Hz)
time_window=96; % Spectrogram windows (ms)
overlap=87.5; % Spectrogram % overlap
limit=256; %valid frame for ITD
%%% Create ILD and IPD data from white noise recordings:
createInterauralData(limit,WN_SOUND_PATH,DATA_PATH,proj_name,...
                     time_window,overlap,fs,verb);

%%%%%%%%%%%%%%%%%%% TRAINING USING WHITE-NOISE AND PPAM %%%%%%%%
%%% Load white-noise ILD and IPD data:
s = load([DATA_PATH,proj_name,'_ILD']);
ILD_data = s.ILD_data;
s = load([DATA_PATH,proj_name,'_IPD']);
IPD_data = s.IPD_data;
freq_idx_ILD = 1:size(ILD_data,1); % Keep all frequencies for ILD
freq_idx_IPD = 2:size(IPD_data,1); % Remove f=0 for IPD
ILD_data=ILD_data(freq_idx_ILD,:);
IPD_data=IPD_data(freq_idx_IPD,:);
% Interaural training data:
Y_data=[ILD_data;real(exp(1i*IPD_data));imag(exp(1i*IPD_data))];

 rho=sqrt((wn_position(2,:)-real_mic(7,1)).^2+...
            (wn_position(3,:)).^2);
 alpha=wn_position(10,:);
 x=[rho;alpha];

y=Y_data(:,wn_position(1,:));
```

```
Maxiter = 30; % Maximum number of iterations
flag=32;
% Multiscale training:
%for i=1:length(flag)
    K = flag; % Number of piecewise-affine components
    [Theta,~] = ppam(x,y,K,Maxiter,verb);
    % Save learned parameters:
    parameter_file = [DATA_PATH,proj_name,'_Theta_ILPD_K',...
                      int2str(K)];
    save([parameter_file,'.mat'],'Theta','-v7.3');
%end
```

A.2.2 PRADO_ dataset_ test1source_ filter

```
clear all
close all
clc

addpath('the_GLLiM_toolbox\');
addpath('octave\');

verb = 1; % Level of output displays

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PATHS TO DATA %%%%%%%%%%%%%%%%%%%%
% Path to the AVASM dataset:
PATH = 'PRADO dataset/';

% Code name of the project (set by user):
proj_name='polar_PRADO_3S_16k_96_bank_newGT_newILPD';

% Path to training data:
DATA_PATH=[PATH 'Data/'];

% Path to test sounds:
TEST_SOUND_PATH=[PATH,'test_sound_3/'];

%%%%%%%%%%%%%%%%%%%% SINGLE SOURCE LOCALIZATION TEST %%%%%%%%%%%
% Parameters:
fs=16000; % Frequency of sampling used (in Hz)
time_window=96; % Spectrogram windows (ms)
overlap=87.5; % Spectrogram % overlap
K =32; % Number of piecewise-affine components
limit=256; % valid frame for ITD

% Load training data and ground truth positions:
load([PATH,'testing_genelec_3.mat']);
test_position=new_pos_test';
load([PATH,'mic.mat']);
```

```
parameter_file = [DATA_PATH,proj_name,'_Theta_ILPD_K',...
                  int2str(K)];
load([parameter_file,'.mat']); % Theta
load([DATA_PATH,proj_name,'_ILD']);
load([DATA_PATH,proj_name,'_IPD']);

freq_idx_ILD = 1:size(ILD_data,1);  % Keep all frequencies
                                    % for ILD [=size(ILD/IPD,1)]
freq_idx_IPD = 2:size(IPD_data,1);  % Remove f=0 for IPD
                                    % because it is null

Ntest = 100; % Number of sounds tested

er_az=zeros(Ntest,1);
er_el=zeros(Ntest,1);
for t=1:Ntest;

    posidx=t;
    sound_nr = test_position(1,posidx);

    true_az= sqrt((test_position(2,posidx)-real_mic(7,1)).^2 ...
            +(test_position(3,posidx)).^2);
    true_el= test_position(10,posidx);

    [test_sound,rfs] = audioread([TEST_SOUND_PATH,...
                        int2str(sound_nr),'.wav']);

    fprintf(1,'Localization test %d (Sound %d.wav at ...
            (%2.2f,%2.2f):\n',t,posidx,true_az,true_el);

    %%% Compute interaural spectrograms:
    mic=[1,2]; % Pair of microphones used
    threshold=-20; % Threshold for missing data (in dB)

    %ERB-filterbank
    [x_1,~]=PRADO_oct3bank(test_sound(:,1));
    [x_2,~]=PRADO_oct3bank(test_sound(:,2));

    test_sound=[x_1,x_2];

    % Resample the sound and select microphones:
    test_sound=[resample(test_sound(:,mic(1)),fs/gcd(rfs,fs),...
            rfs/gcd(rfs,fs)),resample(test_sound(:,mic(2)),...
            fs/gcd(rfs,fs),rfs/gcd(rfs,fs))];

    %dereberberation stage
    reverb1=[test_sound(:,1), test_sound(:,1)];
    reverb2=[test_sound(:,2), test_sound(:,2)];
```

```
    dereverb=[];
    dereverb(:,1)=PRADOdereverb(reverb1,fs);
    dereverb(:,2)=PRADOdereverb(reverb2,fs);

    [~,~,alpha,phi,Chi] = binaural_spectrograms(dereverb,...
                            fs,time_window,overlap,threshold);

    %interaural spectrogram
    phi=phi(freq_idx_IPD,:);
    alpha=alpha(freq_idx_ILD,:);
    Y=[alpha;real(exp(1i*phi));imag(exp(1i*phi))];
    % Indicator of missing data:
    Chi=[Chi(limit+freq_idx_ILD,:);Chi(freq_idx_IPD,:); ...
            Chi(freq_idx_IPD,:)];

    %%% Spectrogram inversion using the mapping parameters Theta:
    xmap = ppam_spectrogram_inverse(Y,Chi,Theta,verb);

    %%% Computer errors:
    er_az(t) = abs(xmap(1)-true_az);
    er_el(t) = abs(xmap(2)-true_el);
    fprintf('rho=%2.2f alpha=%2.2f\n', xmap(1),xmap(2));
    fprintf(1,'Single-source localization error : rho = %2.2f,...
             alpha = %2.2f\n\n',er_az(t),er_el(t));
end

fprintf(1,'\n==== Average Localization Errors  ====\n');
fprintf(1,'rho = %2.2f +- %2.2f, alpha = %2.2f +- %2.2f\n',...
          mean(er_az),std(er_az),mean(er_el),std(er_el));
```

You can notice the main improvements we introduced: the new parameters, the new coordinate system, the ERB filtering stage done by *PRADO_oct3bank* and the dereverberation stage executed by *PRADOdereverb* based on [43]. Their influence on the final result are explained in a detailed way in Chapter 4 and 5.

A.2.3 PRADO_ oct3bank

```
function [P,F] = PRADO_oct3bank(x)

Fs = 16000;                 % Sampling Frequency
N = 3;                   % Order of analysis filters.
F = [ 100 125 160, 200 250 315, 400 500 630, 800 1000 1250, ...
    1600 2000 2500, 3150 4000 5000 ]; % Preferred labeling freq.
ff = (1000).*((2^(1/3)).^[-10:7]);  % Exact center freq.
P=zeros(length(x),1);

% Design filters in 1/3-oct. bands
```

```
% 5000 Hz band to 100 Hz band, direct implementation of filters.
for i = 18:-1:1
   [B,A] = oct3dsgn(ff(i),Fs,N);
   y = filter(B,A,x);
   P = P+y;

end

end
```

A.2.4 PRADOdereverb

```
 %DEMO_CDR_DEREVERB
%
% Demonstration of CDR-based noise and reverberation suppression.
%
% To use this with your own recordings:
% 1. Change wave filename
% 2. Adapt microphone spacing (cfg.d)
% 2. Adapt cfg.TDOA, or use the DOA-independent estimator
%    (estimate_cdr_nodoa)
%
% Reference:
% Andreas Schwarz, Walter Kellermann, "Coherent-to-Diffuse Power
% Ratio Estimation for Dereverberation", IEEE/ACM Trans. on Audio,
% Speech and Lang. Proc., 2015 (under review); preprint available:
% arXiv:1502.03784
% PDF: http://arxiv.org/pdf/1502.03784
%
% Andreas Schwarz (schwarz@lnt.de)
% Multimedia Communications and Signal Processing
% Friedrich-Alexander-Universitaet Erlangen-Nuernberg (FAU)
% Cauerstr. 7, 91058 Erlangen, Germany

function y = PRADOdereverb(x,fs_in,alpha)

addpath(genpath('lib'));

%% filterbank initialization
cfg.K = 512; % FFT size
cfg.N = 128; % frame shift
cfg.Lp = 1024; % prototype filter length

load('lib/filterbank/prototype_K512_N128_Lp1024.mat');

%% algorithm and scenario configuration
cfg.fs = 8000;      % sampling rate [Hz]
cfg.c = 342;         % speed of sound [m/s]
```

```
cfg.d_mic = 0.15;   % mic spacing [m]

% all estimators except estimate_cdr_nodoa require the TDOA of
% the signal; make sure to adapt this when loading another wave
% file
cfg.TDOA = abs(cfg.d_mic*sin(alpha-pi/2)/cfg.c);

cfg.nr.lambda = 0.68; % smoothing factor for PSD estimation
cfg.nr.mu = 1.3;     % noise overestimation factor
cfg.nr.floor = 0.1;  % minimum gain
cfg.nr.alpha = 1; cfg.nr.beta = 1; % power subtraction
%cfg.nr.alpha = 2; cfg.nr.beta = 0.5; % magnitude subtraction
%cfg.nr.alpha = 2; cfg.nr.beta = 1; % Wiener filter

%cfg.estimator = @estimate_cdr_unbiased;
    % unbiased estimator (CDRprop1)
cfg.estimator = @estimate_cdr_robust_unbiased;
    % unbiased, (CDRprop2) "robust" estimator
%cfg.estimator = @estimate_cdr_nodoa;
    % DOA-independent estimator (CDRprop3)
%cfg.estimator = @estimate_cdr_nodiffuse;
    % noise coherence-independent estimator (CDRprop4)
    % does not work for TDOA -> 0!

%% preparation
x = resample(x,cfg.fs,fs_in);

%% Signal processing

% analysis filterbank
X=DFTAnaRealEntireSignal(x,cfg.K,cfg.N,p);

% estimate PSD and coherence
Pxx = estimate_psd(X,cfg.nr.lambda);
Cxx = estimate_cpsd(X(:,:,1),X(:,:,2),cfg.nr.lambda)./...
        sqrt(Pxx(:,:,1).*Pxx(:,:,2));

frequency = linspace(0,cfg.fs/2,cfg.K/2+1)'; % frequency axis

% define coherence models
Css = exp(1j * 2 * pi * frequency * cfg.TDOA);
    % target signal coherence; not required for estimate_cdr_nodoa
Cnn = sinc(2 * frequency * cfg.d_mic/cfg.c);
    % diffuse noise coherence; not required for
    % estimate_cdr_nodiffuse

% apply CDR estimator (=SNR)
SNR = cfg.estimator(Cxx, Cnn, Css);
SNR = max(real(SNR),0);
```

```
weights = spectral_subtraction(SNR,cfg.nr.alpha,...
            cfg.nr.beta,cfg.nr.mu);
weights = max(weights,cfg.nr.floor);
weights = min(weights,1);

% postfilter input is computed from averaged PSDs
% of both microphones
Postfilter_input = sqrt(mean(abs(X).^2,3)) .* ...
                    exp(1j*angle(X(:,:,1)));

% apply postfilter
Processed = weights .* Postfilter_input;

% synthesis filterbank
y = DFTSynRealEntireSignal(Processed,cfg.K,cfg.N,p);
y=y';

end
```

Appendix B

Original code based on Willert algorithm

Here, you can find our Matlab implementation of the Willert algorithm with all the modifications as described in Chapter6.

B.1 Training stage

```
clear all
close all
clc

train_sound_path='training/';
load('Prado dataset/training_genelec_3.mat');

sounddir=dir([train_sound_path, '*.wav']);
nsounds = size(sounddir,1);

fs=16000;
N=floor(0.096*fs);
overlap=floor(7/8*N);
win=hamming(N);
N_fft = 2^(ceil(log2(N)));

load('gaussian_patch.mat');
flag=size(gaussian_patch,1);

for i=1:nsounds

[s,soundfs]=audioread([train_sound_path, num2str(i),'.wav']);
```

```
s=s(round(0.5*soundfs):2*round(0.5*soundfs),:);

if soundfs>fs
    s=resample(s,fs,soundfs);
end

frame=ceil(length(s)/(N-overlap));
dim=size(s);
new_length=frame*(N-overlap)+overlap;
s=[s;zeros(new_length-dim(1),dim(2))];

if dim(:,2)==1;
    fprint('Mono signal - Localization impossible\n');
else
    sl=s(:,1);
    sr=s(:,2);
    [sl,~]=PRADO_oct3bank(sl,fs);      %bark filterbank
    [sr,~]=PRADO_oct3bank(sr,fs);
    [SL,SR]=STFT(sl,sr,N,overlap,win,N_fft,frame);

    useful_high_f=round(size(SL,1)*1500/8000)+5;
    SL=SL(1:useful_high_f,:);
    SR=SR(1:useful_high_f,:);
    [XL,muL,sigmaL]=STFT_patch(SL,flag,gaussian_patch);
    [XR,muR,sigmaR]=STFT_patch(SR,flag,gaussian_patch);

    ITD=final_ITD(sigmaL,sigmaR,XL,XR,gaussian_patch);
    save(['ITD',num2str(i)],'ITD');

end

end
```

B.2 Test stage: 1st proposal

```
clear all
close all
clc

fs=16000;
N=floor(0.096*fs);
overlap=floor(7/8*N);
win=hamming(N);
N_fft = 2^(ceil(log2(N)));
```

```
load('gaussian_patch.mat');
flag=size(gaussian_patch,1);

[s,soundfs]=audioread('test_sound.wav');

s=s(round(0.5*soundfs):2*round(0.5*soundfs),:);

if soundfs>fs
    s=resample(s,fs,soundfs);
end

frame=ceil(length(s)/(N-overlap));
dim=size(s);
new_length=frame*(N-overlap)+overlap;
s=[s;zeros(new_length-dim(1),dim(2))];

if dim(:,2)==1;
    fprint('Mono signal - Localization impossible\n');
else
    sl=s(:,1);
    sr=s(:,2);
    [sl,~]=PRADO_oct3bank(sl,fs);      %bark filterbank
    [sr,~]=PRADO_oct3bank(sr,fs);
    [SL,SR]=STFT(sl,sr,N,overlap,win,N_fft,frame);

    useful_high_f=round(size(SL,1)*1500/8000)+5;
    SL=SL(1:useful_high_f,:);
    SR=SR(1:useful_high_f,:);
    [XL,muL,sigmaL]=STFT_patch(SL,flag,gaussian_patch);
    [XR,muR,sigmaR]=STFT_patch(SR,flag,gaussian_patch);

    test_ITD=final_ITD(sigmaL,sigmaR,XL,XR,gaussian_patch);

end

ITD2=permute(test_ITD,[3,2,1]);

for i=1:19

    load(['ITD',num2str(i),'.mat']);
    ITD=permute(ITD,[3,2,1]);
    test_ITD2=ITD2(:,:,i);
    train_ITD=ITD(:,1:size(test_ITD2,2),i);
    D=pdist2(test_ITD2,train_ITD);

end
```

B.3 Test stage: final proposal

```
clear all
close all
clc

fs=16000;
N=floor(0.096*fs);
overlap=floor(7/8*N);
win=hamming(N);
N_fft = 2^(ceil(log2(N)));

load('gaussian_patch.mat');
flag=size(gaussian_patch,1);

for i=1:100

[input,soundfs]=audioread(['test_sound/', num2str(i),'.wav']);

s=input(round(0.5*soundfs):2*round(0.5*soundfs),:);

if soundfs>fs
    s=resample(s,fs,soundfs);
end

frame=ceil(length(s)/(N-overlap));
dim=size(s);
new_length=frame*(N-overlap)+overlap;
s=[s;zeros(new_length-dim(1),dim(2))];

if dim(:,2)==1;
    fprint('Mono signal - Localization impossible\n');
else
    sl=s(:,1);
    sr=s(:,2);
    [sl,~]=PRADO_oct3bank(sl,fs);      %bark filterbank
    [sr,~]=PRADO_oct3bank(sr,fs);
    [SL,SR]=STFT(sl,sr,N,overlap,win,N_fft,frame);

    useful_high_f=round(size(SL,1)*1500/8000)+5;
    useful_low_f=round(size(SL,1)*300/8000)-5;
    SL=SL(useful_low_f:useful_high_f,:);
    SR=SR(useful_low_f:useful_high_f,:);
    [XL,muL,sigmaL]=STFT_patch(SL,flag,gaussian_patch);
    [XR,muR,sigmaR]=STFT_patch(SR,flag,gaussian_patch);

    ITD=final_ITD(sigmaL,sigmaR,XL,XR,gaussian_patch);
    save(['test_sound_ITD/',num2str(i),'.mat'],'ITD');

end
```

```
end

load('testing_genelec.mat');
final_position=zeros(100,1);

for l=26:75

 load(['test_sound_ITD/',num2str(l),'.mat']);
 test=ITD;
 ITD2=permute(test,[1,3,2]);
 prova=zeros(154,size(ITD2,2)-1);
 result=zeros(154,9,size(ITD2,2)-1);

for j=1:9

    load(['ITD',num2str(5+j),'.mat']);
    ITD=permute(ITD,[1,3,2]);

    for i=1:size(ITD2,2)-1

        test_ITD2=ITD2(:,:,i);
        train_ITD=ITD(33:end,1:size(test_ITD2,2),i);
        test_ITD2=test_ITD2(1:end-1,1:end-1);
        train_ITD=train_ITD(1:end-1,1:end-1);
        d=mahal(train_ITD,test_ITD2);
        d=d/sum(d(:));
        s=sum(d,2);
        prova(:,i)=s;

    end

    result(:,j,:)=prova;

end

position=zeros(size(result,3),1);

for k=1:size(result,3)
    my_result=result(:,:,k);
    [a,b]=min(my_result,[],2);
    estimate=sum(b)/length(b);
    position(k)=estimate;
end

final_position(l)=10*(sum(position)/length(position)+5);

end
```

```
error=abs(final_position(26:75)-new_pos_test(26:75,5));
mean(error)
std(error)
```

B.4 Complementary functions

B.4.1 STFT

```
function [SL,SR] = STFT(sl,sr,N,overlap,win,N_fft,frame)

SL=zeros(frame,N_fft/2);
SR=zeros(frame,N_fft/2);
zero_pad=zeros((frame-1)*(N-overlap)+N-length(sl),1);
sl=[sl; zero_pad];
sr=[sr; zero_pad];

for m=0:frame-1
    [~,SL]=frame_fft(sl,SL,m,N,overlap,win,N_fft);
    [~,SR]=frame_fft(sr,SR,m,N,overlap,win,N_fft);
end

SL = abs(fliplr(SL)');
SR = abs(fliplr(SR)');

end
```

B.4.2 frame_fft

```
function [s,S]=frame_fft(s,S,m,N,overlap,win,N_fft)

s=s(m*(N-overlap)+1:m*(N-overlap)+N);
y_=win.*s;
s=fft(y_,N_fft);
S(m+1,:)=s(1:N_fft/2);
s=s(2:N_fft/2-1)';

end
```

B.4.3 STFT_patch

```
function [X,mu,sigma]=STFT_patch(S,flag,window)

[A,B]=size(S);
C=round(flag/2);
```

```
X=zeros(flag,flag,A-flag+1,B-flag+1);
mu=zeros(A-flag+1,B-flag+1);
sigma=mu;

for i=C:A-C
    for j=C:B-C
        temp=S(i-(C-1):i+(C-1),j-(C-1):j+(C-1));
        X(:,:,i-(C-1),j-(C-1))=temp;
        temp=temp.*window;
        mu(i-(C-1),j-(C-1))=mean(temp(:));
        sigma(i-(C-1),j-(C-1))=var(temp(:));
    end
end

end
```

B.4.4 final_ITD

```
function ITD=final_ITD(sigmaL,sigmaR,XL,XR,gaussian_patch)

ITD=zeros(size(XL,3),size(XL,4),size(XL,4));
sigmaN=0.1*mean(sigmaL(:)+sigmaR(:));

for k=1:size(ITD,1)
for i=1:size(ITD,2)
    for j=1:size(ITD,3)
        rho=my_correlation(XL(:,:,k,i),XR(:,:,k,j),...
            gaussian_patch);
        ITD(k,i,j)=my_ITD(sigmaR(k,j),rho, sigmaN);
    end
end
end

end
```

B.4.5 my_correlation

```
function rho=my_correlation(XL,XR,window)

temp=corr(XL,XR);
temp=temp.*window;
rho=mean(temp(:));

end
```

B.4.6 my_ITD

```
function ITD=my_ITD(sigmaR,rho, sigmaN)

ITD=exp(-0.5*(sigmaR/sigmaN)*(1-rho^2))/(sqrt(sigmaN*2*pi));
end
```