Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2022

José Alberto Barros dos Reis

Sistema Anti-Melgas I: Identificação e localização de fontes sonoras

Anti-Mosquito System I: Identification and localization of acoustic sources

Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática
2022

José Alberto Barros dos Reis

Sistema Anti-Melgas I: Identificação e localização de fontes sonoras

Anti-Mosquito System I: Identification and localization of acoustic sources

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Doutor António Guilherme Campos, Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Daniel Filipe Albuquerque, Professor adjunto no Departamento de Engenharia Electrotécnica do Instituto Politécnico de Viseu Escola Superior de Tecnologia e Gestão.

Palavras Chave

Mosquitos, Trajectória, Microfones, Angle of Arrival, Time Difference of Arrival, Correlação cruzada, Azimuth, Far-field, Tempo real, Simulação de sala

Resumo

Esta dissertação aborda o desenvolvimento de um sistema de localização acústica com o objectivo de detectar mosquitos dentro de casa. Começou com um breve estudo do som produzido pelos insectos, especialmente os mosquitos fêmea, com o objectivo de compreender as características espectrais; Foi realizada uma revisão do nosso sistema auditivo e da sua capacidade de localizar espacialmente fontes sonoras. As principais pistas 2D são ITD (interaural time difference) e ILD (interaural level difference). O exemplo da audição humana mostra como a diversidade espacial dos sensores é indispensável para a localização do som; Assumiu-se um cenário 2D, reduzindo assim o problema da estimativa de azimute, que requer dois microfones. Assumindo que a distância da fonte ao receptor é muito maior do que a distância entre microfones (aproximação "far-field"), o ângulo de azimute procurado pode ser obtido através de uma fórmula aproximada. Foi avaliado o erro intrínseco causado pela própria aproximação "far-field", bem como o impacto de possíveis erros na estimativa dos parâmetros de cálculo: velocidade do som, espaçamento entre microfones e atraso temporal; O trabalho de desenvolvimento, realizado no ambiente MATLAB, foi baseado num simulador existente. O elemento central do sistema é o processamento digital dos sinais recebidos nos dois microfones. O método de correlação cruzada é utilizado para calcular o tempo de espera entre eles. A interpolação foi aplicada para aumentar a resolução da estimativa do pico de correlação cruzada; Foi desenvolvido um script com uma interface gráfica para combinar o preditor com o simulador. Facilita ao utilizador a especificação da trajectória a reproduzir no simulador. O ficheiro de áudio a ser injectado é também escolhido pelo utilizador. O simulador devolve um ficheiro estéreo com os sinais do microfone. O script gera um ponteiro que se move em tempo real para indicar a posição estimada da fonte; Foram realizadas simulações e testes experimentais, numa sala anecóica sem fontes adicionais de ruído. O erro da estimativa de azimute medido na simulação confirmou o comportamento previsto, tendo em conta as fontes de erro intrínsecas à aproximação "far-field". O erro é menor quando a fonte se situa entre 45° e 135°. Fora deste intervalo, aumenta, atingindo um pico nos extremos (0° e 180°). Aproxima-se de zero quando a fonte está a 90°, formando um padrão simétrico em forma de U em torno deste valor. Quando o ruído é introduzido, as estimativas feitas perdem qualidade, como esperado; para SNR inferior a -10 dB, o erro ultrapassa os 10°; Os testes experimentais consistiram em dois microfones, um altifalante e uma interface de áudio para comunicar com o computador. Foi criada uma câmara de absorção para reduzir os reflexos acústicos e o ruído externo. Foram feitas gravações para cada ângulo de azimute, com longa duração. Com todos os ficheiros processados, o padrão do erro de estimativa do azimute também teve a forma de U, embora não tenha tido uma simetria perfeita.

Keywords Mosquitoes, Trajectory, Microphones, Angle of Arrival, Time Difference of Arrival, Cross-correlation, Azimuth, Far-field, Real-time, Room Simulation

Abstract

This dissertation addresses the development of an acoustic localisation system with the aim of detecting mosquitoes indoors. It starts with a brief study of the sound produced by insects, with special focus on the case of female mosquitoes, aimed at understanding the spectral characteristics; A review was carried out on our auditory system and its ability to spatially locate sound sources. The main 2D cues are ITD (interaural time difference) and ILD (interaural level difference). The example of human hearing shows how spatial diversity of sensors is indispensable for sound localisation; A 2D scenario was assumed, thus reducing the problem to azimuth estimation, which requires two microphones. Assuming that the distance from the source to the receiver is much greater than the distance between microphones (far-field approximation) the sought azimuth angle can be obtained by an approximate formula. The intrinsic error caused by the far-field approximation itself was assessed, as well as the impact of possible estimation errors in the calculation parameters: speed of sound, microphone spacing and time delay; The development work, carried out on a MATLAB environment, was based on an existing simulator. The central element of the system is the digital processing of the signals received at the two microphones. The cross-correlation method is used to work out the time delay between them. Interpolation was applied to increase the resolution of the cross-correlation peak estimate; A script featuring a graphical interface was developed to combine the predictor with the simulator. It makes it easy for the user to specify the trajectory to be reproduced in the simulator. The audio file to be injected is also chosen by the user. The simulator returns a stereo file with the microphone signals. The script generates a pointer moving in real time to indicate the estimated position of the source; Several other simulations and experimental tests were carried out, based on an anechoic room without additional sources of noise. The azimuth estimation error measured in simulation confirmed the predicted behaviour taking into account the sources of error intrinsic to the far-field approximation. The error is smaller when the source is between 45° and 135°. Outside this range, it increases, peaking at the extremes (0° and 180°). It approaches zero when the source is at 90°, forming a symmetric U-shaped pattern around this value. When noise is introduced, the estimations made lose quality, as expected; for SNR less than -10 dB, the error exceeds 10°; The experimental tests involved two microphones, a loudspeaker and an audio interface for communication with the computer. An absorbing chamber has been created to reduce sound reflections and external noise. Recordings of long duration were made for each azimuth angle. With all the files processed, the pattern of the azimuth estimation error was also U-shaped, although not perfectly symmetric.

Contents

List of Figures

List of Tables

Glossary

AoA	Angle of Arrival
SNR	Signal-to-noise ratio
TDoA	Time Difference of Arrival
IR	Impulse response
MR	Multiple reflections
3D	Three-dimensional
2D	Two-dimensional
GUI	Graphical User Interface
IDE	Integrated Development Environment
LMB	Left mouse button
RMB	Right mouse button
ADC	Analog-to-digital converter
ITD	Interaural Temporal Difference
IPD	Interaural Phase Difference
ILD	Interaural Level Difference
HSE	Head-shadow effect
FFT	Fast Fourier transform
MUSIC	Multiple signal classification
MVDR	Minimum Variance Distortionless Response
ESPRIT	Estimation of Signal Parameters via Rotational Invariance Techniques
DoA	Direction of Arrival
HRTF	Head-related transfer function

CHAPTER 1

Introduction

At some point in their lives, everybody gets to know how annoying mosquitoes can be – especially at night[1], when most people aim for an undisturbed rest. Some may try to ignore them, hoping that they will go away, but most vow to eliminate every single mosquito, to ensure their sleep will not be disturbed.

The real problem lies not in the actual killing of the mosquitoes, but in locating them in a quick, reliable and safe way. Some may argue that chemical repellents remove the necessity of locating them. However, mosquitoes mutate and evolve to adapt to those chemicals[2], making it a temporary solution. The ultimate goal of this project is to solve that problem buy building a system that is inexpensive, inconspicuous, and comfortable to use.

Various aspects, both theoretical and practical, must be considered for this purpose. For instance, the biology, behaviour and environmental impact of mosquitoes as well the understanding of the human hearing system can help determine how the system should be built. Obviously, it is also mandatory to review the state of the art in terms of existing anti-mosquito systems, to help shape the development of a prototype and guide its critical evaluation.

1.1 Objectives

The idea of the dissertation proposal[3] was to trace the trajectories of the mosquito and corresponding landing points (e.g. on the wall or ceiling of the room) based on the acoustic signals it emits while flying. The landing point information should be presented to the user with minimal delay and without alerting the mosquito. To achieve this, the proposal envisaged a system of combining two blocks, as shown in figure 1.1: the acoustic locator to trace the mosquito position and a light pointer to estimated landing points. The user would only need to observe the output of block II, since the acoustic locator – the block to be addressed in this project – would be fully automatic, with signal processing algorithms capable of recognising a specific sound and estimate its source location.

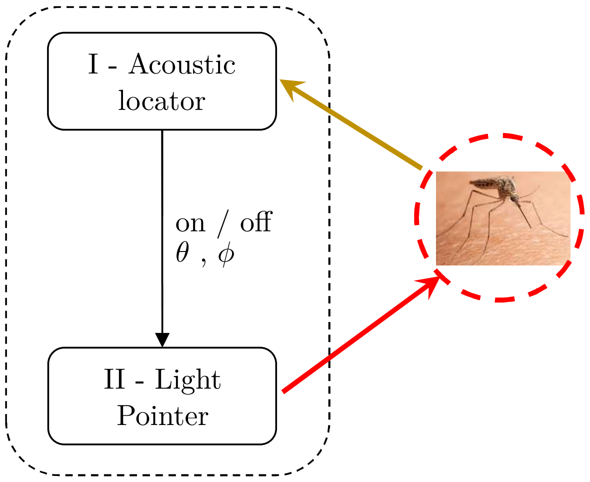


Figure 1.1: Block diagram of the proposed system – adapted from [3].

Following each landing point estimation, it should output a pair of results: the flying status – *on/off* (flying/standing) – and the mosquito position in terms of azimuth (θ) and elevation (ϕ) angles (a reminder of the spherical coordinate system is given by figure 1.2). It is wirth noting that since the mosquito is to be pointed while at rest on a room boundary, it is not necessary to determine distance (ρ).

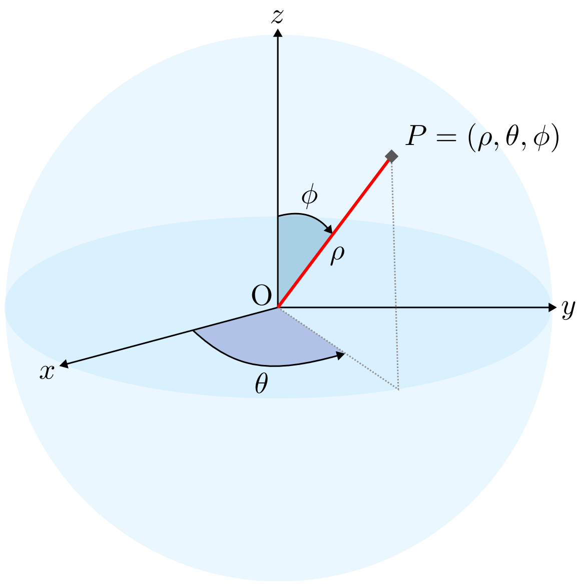


Figure 1.2: Spherical coordinates system reference.

The acoustic locator would essentially comprise a set of acoustic sensors in the audible range (i.e. microphones) with corresponding Analog-to-digital converter (ADC) converters (i.e. digital sound card) and a digital signal processing unit. The captured audio could be filtered to suppress out-of-band noise (i.e. frequency components outside the band emitted by the mosquito), as suggested in figure 1.3.

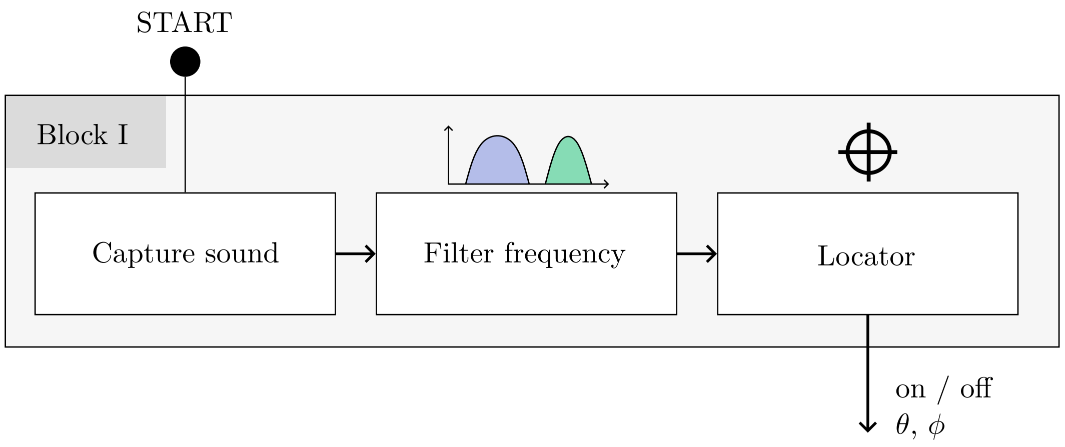


Figure 1.3: Tasks embedded in block I - acoustic locator.

The proposal targeted a specific application (mosquito detection), but it was clear from the outset that the technology to be explored (acoustic location) is by no means restricted to that particular application, which adds to the interest of the topic.

1.2 Insect sounds

1.2.1 Sound production

It is well known that sound is a vibration that transmits through a medium. Many insects emit vibrations, either purposely (for communicating) or incidentally (as a result of their activity), they emit a sound.[4]. Some examples can be observed in table 1.1, where there are two categories of sound generated by insects mentioned previously.

Incidental	**Communication**
walking	mating signals
flying (adults)	courtship signals
chewing	territorial displays
swimming	social & subsocial signals
breathing	anti-predator signals
heartbeats	warning signals

Table 1.1: Categories of sound generated by insects – adapted from [4].

However, sound poses as cue for predators and the insects suffer a process of natural selection, where those who attract less attention survive. This means that insects with a low profile in incidental sounds will be favourable – the sound category determined mostly by their biology.

Consequently, the characterization and occurrence of both categories is heavily influenced, as it can be observed in table 1.2. The contrast between both categories is noticeable although, the points of interest are that:

- The incidental sounds are not necessarily specific for each specie i.e. the specie complex[1] can be characterized the same incidental behaviour;
- Incidental sounds have a "broad" frequency range which makes the detection of insects hard – for not having a spectral "signature".[2]

	Incidental	**Communication**
Species Specificity	low	high
Frequency Range	broad	narrow
Localization	difficult	easy
Distance Travelled	short	long
Duration	short	long
Timing	unpredictable	predictable

Table 1.2: Generalized comparison of characteristics between both sound categories – adapted from [4].

1.2.2 General biology

Similar to humans, insects have an internal biological clock that regulates the sleep-wake cycle among other activities and behaviours – such as daily rhythms of locomotion, laying eggs, feeding and mating. While in humans the clock follow a circadian rhythm ($\approx$ 24 hours), in insects the clock ranges from circa-tidal rhythm ($\approx$ 12 hours) to circa-annual rhythm ($\approx$ 1 year). Individual insects often have a spontaneous change of the biological clock rhythm, which reinforces the *unpredictability* in *incidental* behaviours pointed out in table 1.2.[5]

Besides considering the biological clock, insects can have different behaviours when facing a situation of hunger/deficit of nutrients. This induces an increase of effort to gain resources and other nutrients that would be ignored by less desperate individuals.[6] For many such cases it involves the movement of the insect which causes the production of *incident* sounds. Nonetheless, the contrary can also be applied: if an insect or population has abundance of food, the hunting and foraging decreases which also reduces their movements (and the resulting sound).

Another important factor is where the insects live and how they interact with each other, if living in colonies. Density and dispersion are the metrics that describe the conditions of their colony. A denser colony makes mating easier and depending on their dispersion, it can generate more *communication* sounds. Meanwhile, dispersion is used to describe a dense colony – spatial distribution of species. It serves as a protection measure for certain temperatures and as an isolation barrier from certain species (while at the same time increasing the risk of being targeted). This means that an aggregated colony favours the population by decreasing the *communication* sounds (but increasing the *incident* sounds) if the predator hunts by sensing sounds.[6]

[1]Group of species similar in appearance and behaviour but nevertheless, distinct from each other.

[2]The spectrum of a sound (relative power at different frequencies) is dependent of the distance from its source and so it can be mixed with the background noise.

Lastly, seasonality plays an important role on the presence and growth of insects. As the environmental conditions change, such as the temperature, terrain conditions and nutrient abundance, insects are not exposed to their optimal conditions. This triggers their survival mechanism, diapause, where their development is delayed. During this stage some insects are immobilized whereas others are active, but with reduced feeding and breeding. [6], [7] Either by being immobilized or by reducing their routines to survive, this mechanism decreases the production of any sound made by the insect.

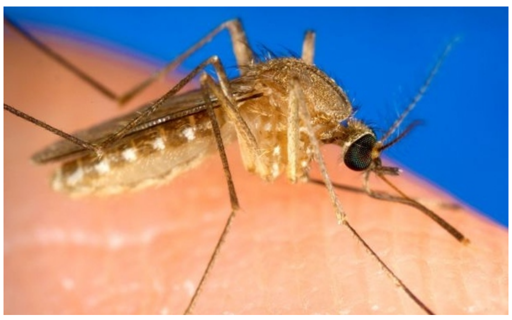
1.2.3 Mosquitoes

Just as the majority of insects, mosquitoes are part of a species complex, as it can be observed in figure 1.4. Being part of the *Culicidae* family, they belong to a group of approximately 3600 species.[8].

By analysing the previous diagram in figure 1.3, the mosquito frequencies must be known a-priori to implement a frequency filter. However, gathering all the frequencies for all species presents a difficult task. To circumvent this problem, three main genera [3] from the *Culicidae* family are considered: the *Aedes*, *Culex* and *Anopheles*. These genera represent the most common (and invasive) mosquitoes among America and Europe [9]–[12] as well it represents the majority of mosquitoes that carry diseases.[13]

(a) *Aedes albopictus.*

(b) *Anopheles minimus.*


(c) *Culex quinquefasciatus.*

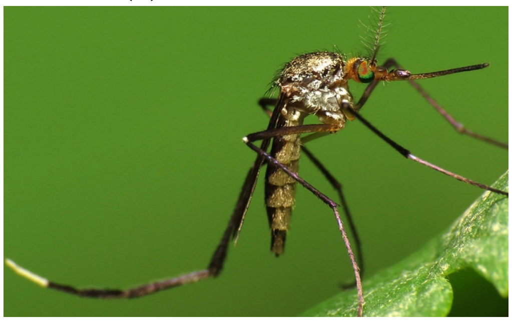

(d) *Psorophora ferox.*

Figure 1.4: Example of a mosquito specie complex – with 4 distinct species (CC license by 4.0).

[3]Genus (plural: genera) is a taxonomic rank used to classify organisms. Family is on top of the hierarchy, followed by genus and later by the species.

Going into detail, mosquitoes produce a sound when they are either flying – by the speed of the beating of their wings – or by *incidental* sounds such as touching the wing with its "legs".[14], [15] When alone, a mosquito does not make any audible sound. However, when in a group, they will produce an audible sound[14] – which strongly relates to the communication between them and reinforces the sound characterization in table 1.2. Male mosquitoes should not be ignored because, while they consume flowers and nectars to survive[16], they can tune their wing-beat frequency to match the one that a female mosquito sends – with the intent to mate.[17] If they are not mating they can be filtered due to the fact that their wing-beat frequency has a higher pitch that their female counterpart.[14]

So far, the most useful information about mosquitoes lies on the sound they produce when they use their wings. For a more robust frequency characterization of the mosquito species it is important to gather data from various recordings measurements. The first data was gathered using a smartphone's acoustic sensors, which accurately captured the mosquito's wing-beats without any distortion[18], whose results produced by the article can be observed on table 1.3 and figure 1.5.

Specie	Frequency(Hz)	Recordings
aegypti	450 – 620	3190
albopictus	540 – 650	4855
mediovittatus	395 – 455	1505
sierrensis	350 – 455	7734

Table 1.3: Female *Aedes* mosquito wing-beat frequencies – adapted from [18].

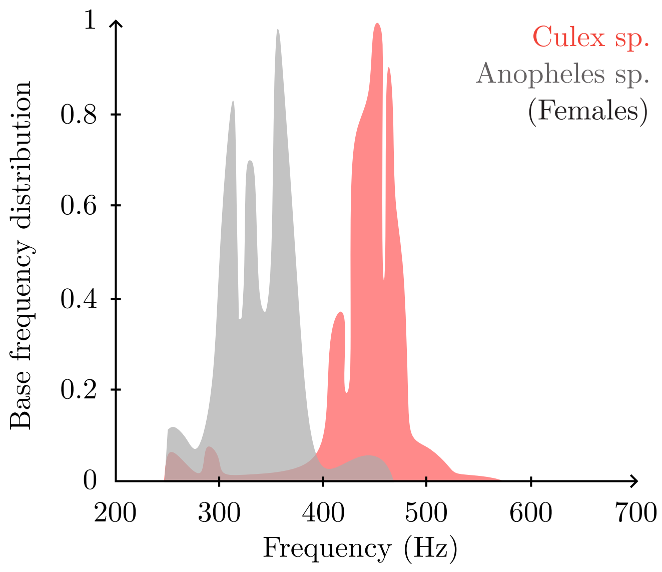


Figure 1.5: Distribution (estimation) of the *Culex* and *Anopheles* mosquito species frequency generated by their wing-beat – adapted from [18].

The second frequency data comes mostly from Clements[19], and the rest complemented by Brogdon[20], [21] and Cator, Arthur, Ponlawat, *et al.*[22]. Besides presenting the frequency range for the mosquito species, as observed in figures 1.6, 1.7 and on table 1.4, it also demonstrates how the wing-beat frequency can change depending on the temperature of the environment where the mosquito is flying – as observed in figure 1.8.

Another information omitted in the following results is that the mosquito age has been annotated and in fact it does contribute the wing-beat frequency value.[19]. It was spared from the figures and tables to make the results simpler.

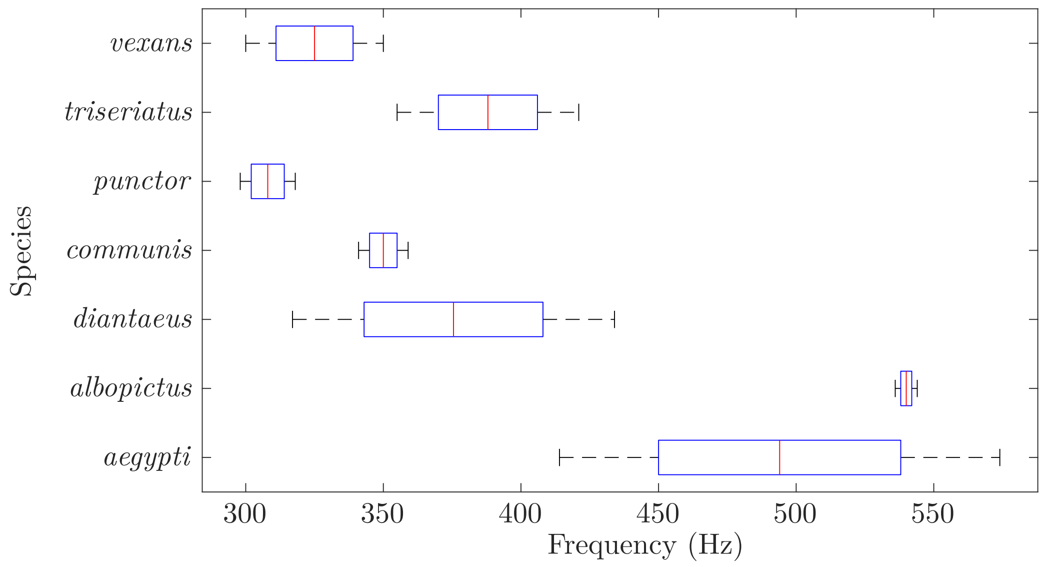


Figure 1.6: Female *Aedes* mosquito frequency range values (Clements).[19]

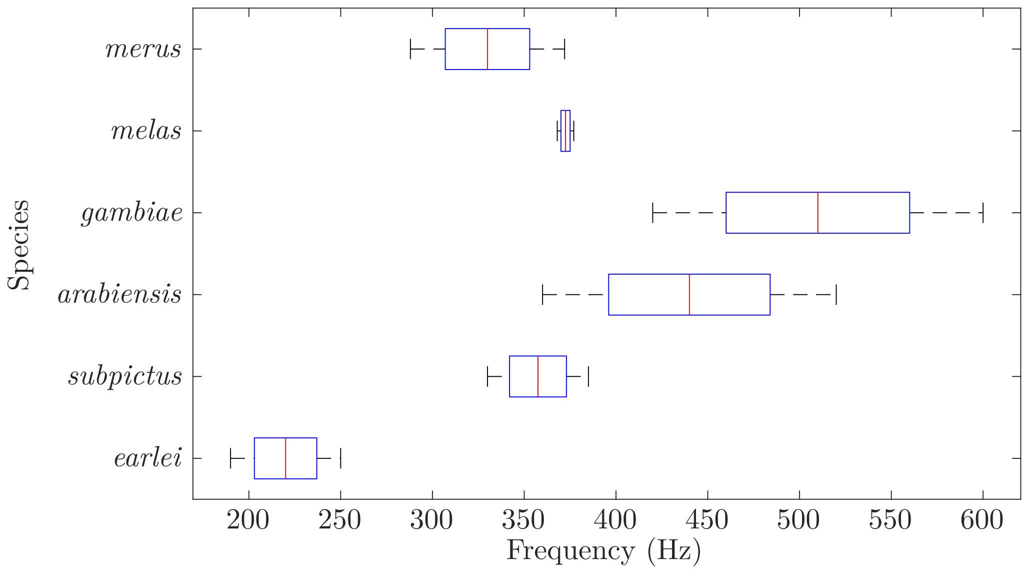


Figure 1.7: Female *Anopheles* mosquito frequency range values (Clements).[19]

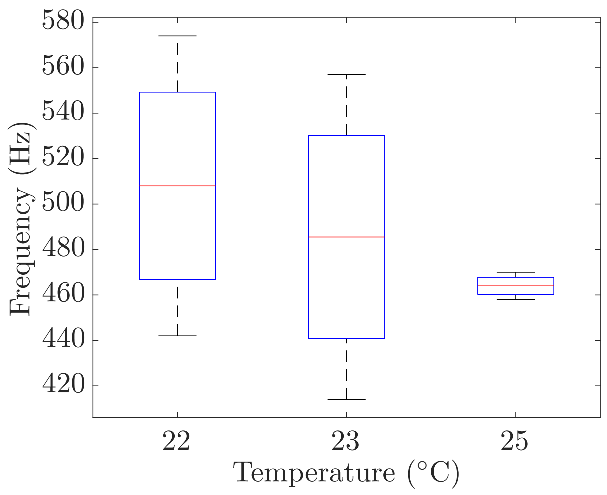


Figure 1.8: Female *Aedes aegypti* frequency range on different temperatures (Clements).[19]

Specie	Frequency (Hz)
pipiens	329 – 370

Table 1.4: Female *Culex* mosquito species frequency (Clements).[19]

Merging the two data sets gives a more generalized frequency range for the three main genera of the female mosquito species. The data gathered by Mukundarajan, Hol, Castillo, *et al.* complements the Clements dataset, since this last dataset is not diverse on the *Culex* species. Considering the worst-case scenarios i.e. choosing the furthest frequency limits, the frequency range for the mosquito genus species are listed in table 1.5.

Genus specie	Frequency (Hz)
Aedes	350 – 650
Anopheles	190 – 600
Culex	250 – 450

Table 1.5: Generalized frequency range for female mosquitoes.

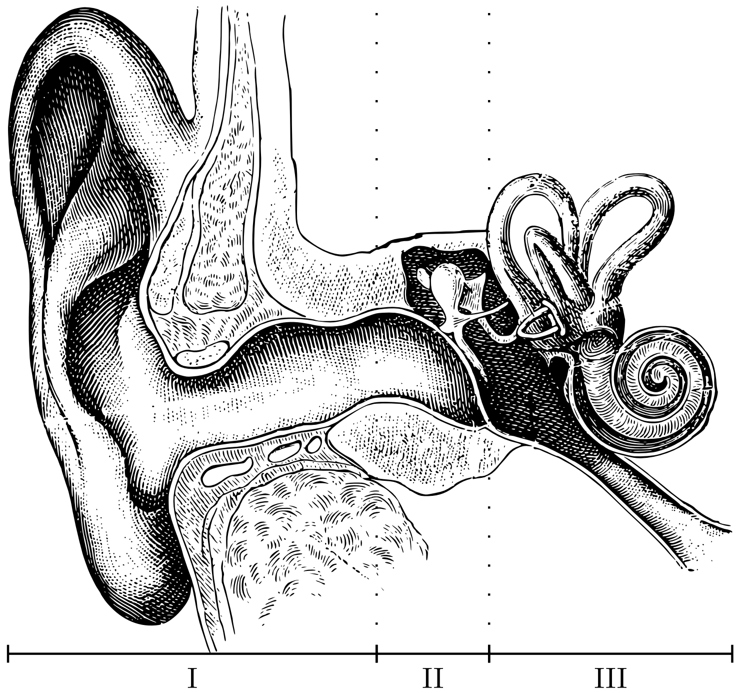
Other aspect to be aware is that mosquitoes wing-beat waveforms contain a harmonic series, where their wing-beat frequency (as the ones demonstrated in table 1.5) represent the fundamental frequency.[15]

In the end, to filter only the mosquito frequencies a band-pass filter can be applied – with a range of [190 650] Hz – which facilitates the predictor or locator algorithm. If, however, the user pretends to create a species-specific locator, it can implement various band-pass filters based on the mosquitoes respective frequencies. What matters is that the frequency ranges are listed and the filter implementation is flexible and is up to the user.

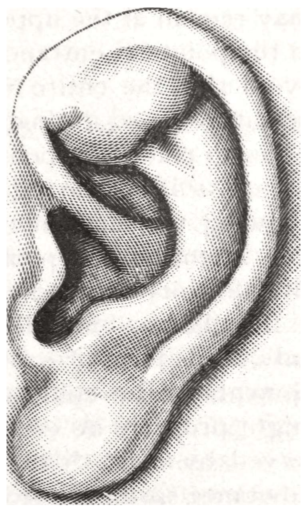
1.3 Human auditory system

One aspect that often goes unnoticed is the geometry and spatial configuration of the acoustic sensor structure. As nature is the master of creativity, where multiple engineering feats were inspired on them (e.g. see Leonardo da Vinci inventions), the inspiration for an acoustic structure is also going to be inspired on something that already exists in nature – the human ear.

In a human ear there are three main areas: the outer, middle and inner ear, as observed in figure 1.9a. For the purpose of the structure, the anatomy of the middle and inner ear is discarded, since they do not possess any evident spatial information. That being said, the outer ear is characterized by its auricle (*pinna*) and the ear canal, which can be observed in figure 1.9. Its function is to reflect and focus the sound waves into the middle ear. While in some animals the *pinna* can be moved to search for sound sources (e.g. cats and dogs), in humans that is not possible. To have a similar spatial information a human needs to move its head. [23]



(a) Human ear represented into three main areas: I) Outer ear; II) Middle ear; III) Inner ear.



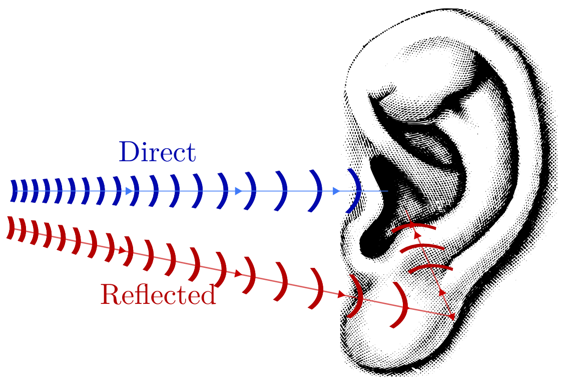
(b) Outer ear when observed in front (*pinna*).

Figure 1.9: The anatomy of an human ear (CC license by 4.0).

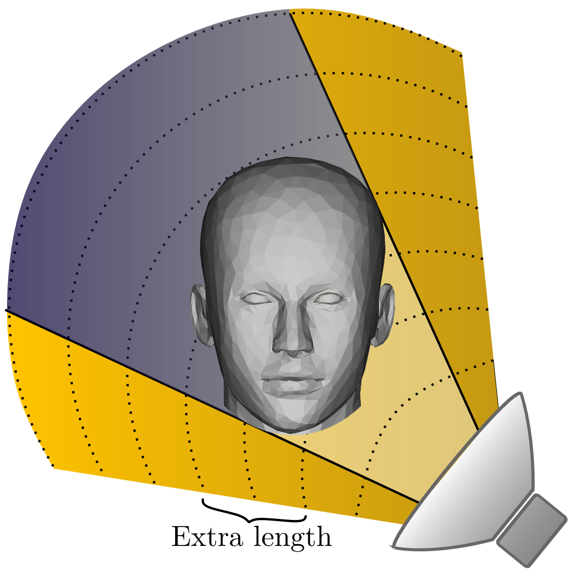
When sound waves hit the *pinna*, they can either travel directly or reflect until they enter the ear canal as observed in figure 1.10a. This forms a cue for *monaural*[4] prediction systems, where they are based on the asymmetry and reflections that occur in the receiver. [24] Besides the human body, the pinna geometry and its resulting filtering effects also pose a cue that help in locating sounds in the vertical plane (elevation).[25], [26]

[4]Sound localization based on one acoustic sensor.

Furthermore, there is the Head-shadow effect (HSE) where the sound is obstructed by the head and results in an attenuated signal at the respective shadow zone.[27] An attenuation that occurs mostly on high frequency waves, since its wavelength is too short to bend over the head.[28] The ear located on the shadow zone contains a time delay since the sound wave propagates more distance – as observed in figure 1.10b. Moreover, this cue is useful for *monaural* control listeners (e.g. person deaf in one ear) but not completely adequate for detecting sounds in a Two-dimensional (2D) environment. [29]



(a) Sound waves entering a ear.



(b) Shadow region of the sound waves caused by the head (dark area).

Figure 1.10: Sound waves hitting the *pinna*.

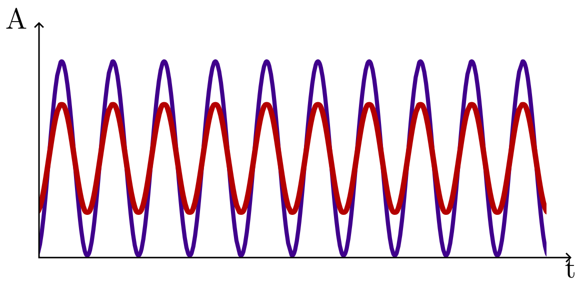
Going back to considering a normal-hearing person[5], they rely on *binaural*[6] control to detect the sound sources in a 2D environment.[30] The cues that are part of the lateral sound localization are due to the HSE and the distance that separates both ears.[31]

An ear located in the shadowed zone receives the same signal of the opposite ear but with an attenuated intensity, which makes up for the Interaural Level Difference (ILD). This can also be caused by the attenuation of sound waves of multiple reflections before arriving at the ear. The second clue is the time delay that an ear suffers in relation to the other, either caused by the HSE, sound reflections or simply by a source that is closer the the other ear – Interaural Temporal Difference (ITD). The final cue is Interaural Phase Difference (IPD), which is strongly dependent on the sound wave frequency and ITD, refers to the phase difference between the two waves. Moreover, the ILD cue is accurate at locating sound sources at high frequencies while the IPD and ITD cues are accurate at low frequencies.[32] Better yet, if both ITD and ILD cues are combined, humans can predict the sound source faster and more

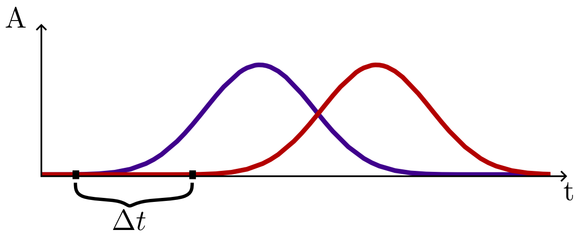
[5]Someone absent of any hearing loss and that does not have any anomaly in the ears, e.g. number of ears not equal to two, deformation in the outer ear, etc.

[6]Sound localization based on at least two acoustic sensors.

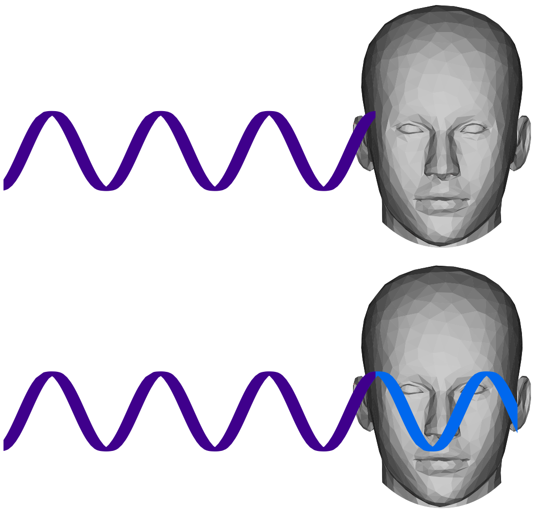
accurately than using each clue individually.[31] For a better understanding of the cues, there is a demonstration of different sound waves in figure 1.11.



(a) Same acoustic waves but with different amplitudes (ILD).



(b) Similar acoustic waves where the red wave has a time delay (ITD).



(c) Example of the same acoustic wave but with a different phase at the entry of both ears.(IPD).

Figure 1.11: Demonstration of the *binaural* control cues.

In the end, when designing a receiver structure for the acoustic sensors one could add some shapes in a way that creates more spatial diversity, as humans have with their *pinnas* (auricles) and the head separating them, acting as a spacer and an obstacle. Given the *binaural* cues that humans use to locate the sound sources, one could exploit a configuration that boosts each cue, e.g. increasing the distance between sensors to have a reasonable time delay or even by adding an object between the both sensors which would create a lower intensity on the other microphone and possibly a different phase – increasing the audio entropy information for the predictor.

1.4 State-of-the-art

There are already solutions to some of the objectives presented in 1.1 but they either are not fit for real-time usage due to the necessity of high computational resources or do not predict with good accuracy in a realistic environment. One solution consisted of a structure with two microphones fixed on a rod, that rotated horizontally with a specified angular velocity and height or elevation angle (that were setup before testing the results).[33] Its model combines the ITD and angular speed into a differential equation, where it is able to predict azimuth and elevation in real-time. At the end the results show that the azimuth is predicted accurately but the elevation is predicted with high uncertainty. However, changing the elevation creates a trade-off between the azimuth and elevation measurements – increasing the elevation improve the elevation measurements at the cost of azimuth measurements.

A simpler solution is based on a passive acoustic system of five microphones and it displays the azimuth of a sound source in real-time.[34] The received signals are interpolated and then compared via cross-correlation (via hardware and software). It predicts almost perfectly the sound sources location in an anechoic room but when the system was exposed to a normal room i.e. with walls and glass panels the results quickly degraded.

It already exist solutions that can predict the Direction of Arrival (DoA) with good accuracy and resolution, like Multiple signal classification (MUSIC), Minimum Variance Distortionless Response (MVDR)[35] and Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT), which is less computationally expensive than MUSIC.[36] The only problem is that these methods cannot estimate the elevation with good accuracy. Due to that limitation, Damarla[37] created a new algorithm that predicts the elevation more accurately – by using the Angle of Arrival (AoA) from each pair of microphones. The setup consisted of a tetrahedral array of microphones and the target used for detection was an helicopter. The results were good, where the AoA was estimated almost perfectly and the elevation, while not following exactly the expected results, produced way better results than MVDR.

To finish, a solution used spatial diversity whose aim consisted of reducing the number of microphones while estimating AoA with a good accuracy.[38] It was based on the auditory cue ILD and used Head-related transfer function (HRTF) along some time delay estimation methods. With all of this, by adding an half-sphere in each microphone he reduced the number of microphones needed in a 2D environment – from three to two – with good results.

As mentioned previously, it is recommended to filter the desired frequencies before applying location prediction algorithms. It becomes important to learn their general behaviour and when and how they produce any sound – aiming to facilitate the search of the desired frequency range.

CHAPTER 2

Sound Source Localisation

This chapter addresses the problem of detecting sound source direction, starting by a 2D scenario i.e. considering only two spatial dimensions (x and y). This means source elevation is disregarded and the problem simplifies to azimuth estimation. Receiver configuration is made as simple as possible, with only two pickup points, the minimum number ensuring the spatial diversity required in 2D. Also, for the time being, the room is assumed to be anechoic (i.e. without reflections) and noise free.

2.1 Azimuth estimation

2.1.1 Formulation of the problem

Figure 2.1 illustrates the 2D scenario to be explored. Point S represents the source (our annoying mosquito flying in the room); M_1 and M_2 represent the two pickup points (microphones) forming the receiver. For convenience, the reference axes are chosen so that M_1 and M_2 be placed symmetrically on the x-axis. Let the spacing between them be Δx and their distances to the source d_1 and d_2, respectively.

Our goal is to work out the source azimuth θ (angle between OS and the positive x-axis) from the signals picked up at M_1 and M_2. As highlighted in the figure, the source-receiver sound path lengths differ by Δd:

$$\Delta d = d_1 - d_2. \tag{2.1}$$

Consequently, there is a delay between the Times of Arrival (ToA) at receivers M_1 and M_2. This delay, known as Time Difference of Arrival (TDoA), and denoted Δt here, is directly proportional to the path-length difference:

$$\Delta t = \frac{\Delta d}{c}, \tag{2.2}$$

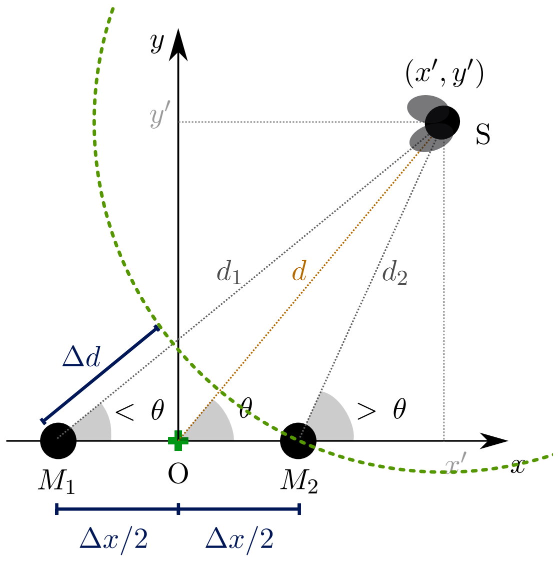


Figure 2.1: 2D source azimuth detection scenario.

where c represents sound propagation speed. This depends on atmospheric conditions; considering only the most influential factor, temperature τ, it is given by:

$$c \approx 20.05 \times \sqrt{273.15 + \tau}, \tag{2.3}$$

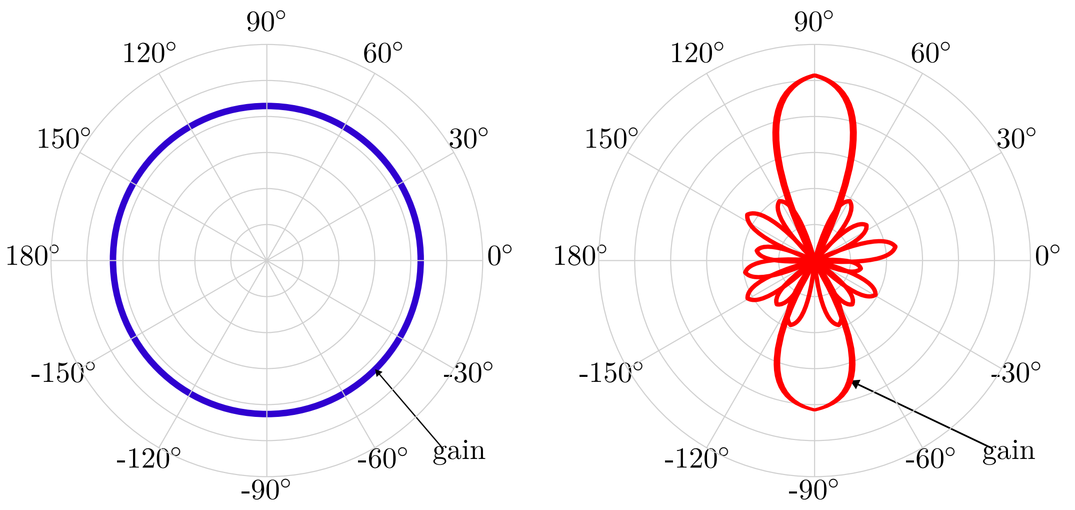
where c is in m/s and τ is in °C [39]. Imagine that in two distinct seasons, like winter and summer, the temperatures can reach as low as 8 °C and as high as 28 °C inside a house. This difference of 20 °C is enough to create an error of 20 μs in the TDoA estimation in a microphone spacing of 20 cm for example. That is why the temperature must be measured for each experiment, to reduce the error propagation caused by the estimations.

Under the assumed conditions (anechoic room, no noise) and so long as M_1 and M_2 are identical and omnidirectional (gain at a certain frequency is equal in all directions – see figure 2.2), the signal at M_1 will be essentially a replica of the signal at M_2: delayed by Δt and multiplied by an attenuation factor according the inverse square law (see annex B.1) governing the variation of sound wave amplitude with travelled distance in free-field (anechoic) propagation:

$$M_1 = \alpha \cdot M_2(t - \Delta t). \tag{2.4}$$

This can be observed on figure 2.3, where two signals f and g are identical but one has a lower amplitude and is shifted to the right. All this conditions favour the cross-correlation of both waveforms, which in signal-processing is a measure of similarity between similar waveforms as a function of the time-lag (delay) applied to them. In discrete systems is represented by:

$$(f \star g)[n] = \sum_{-\infty}^{\infty} \overline{f[m]} \cdot g[m+n]. \tag{2.5}$$



(a) Omnidirectional **(b)** Bidirectional with side lobes (most common)

Figure 2.2: Radiation patterns of two different microphones (at a certain frequency) – (CC license by 4.0).

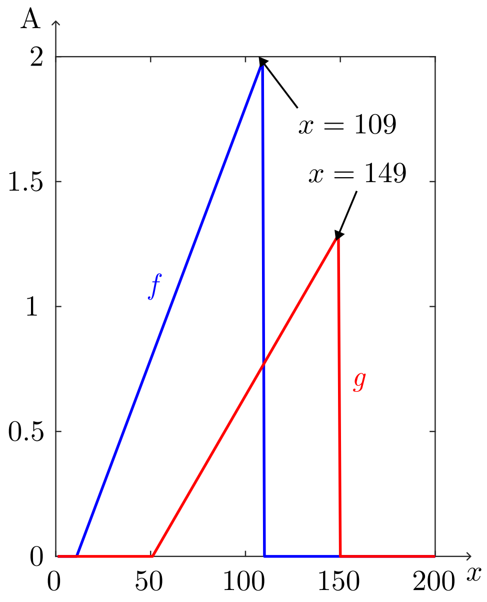


Figure 2.3: Signal f (blue) and its replica g (red).

Basically, equation (2.5) represents the sliding dot product for both functions and this can be represented in figure 2.4. After sliding the function g (since it is $f \star g$; to determine the delay of g in reation to f it should be: $g \star f$) it returns a list of values. The maximum absolute value represents the time delay of g in relation to f and it can be positive – g is delayed – or negative – g arrived first.

Therefore, cross-correlation between the signals picked up at M_1 and M_2 will exhibit a maximum for Δt, as both signals are discrete and have a sampling frequency associated. Converting the maximum value to time units will work out the TDoA (equation (2.2)) and hence the source distance difference Δd (equation (2.1)).

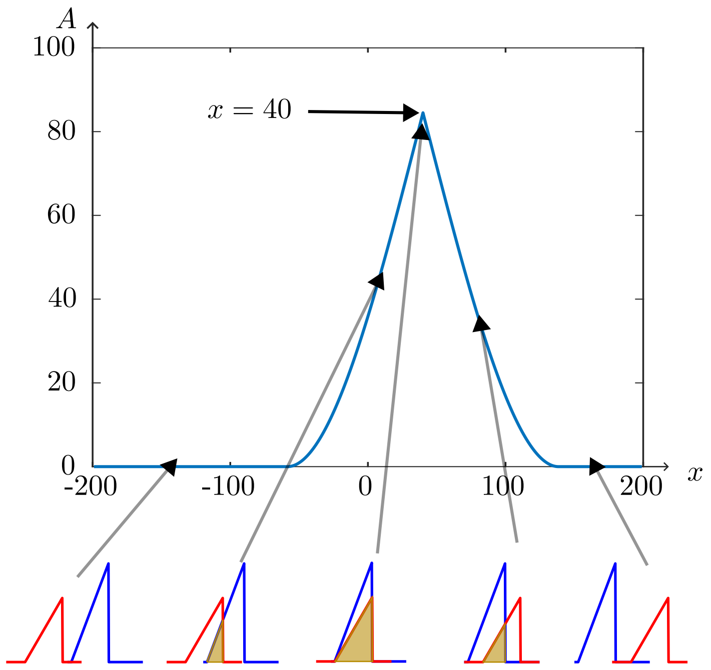


Figure 2.4: Example of cross-correlation in signal processing – applied to signals f and g of figure 2.3 – in this case g is the sliding function: $g \star f$, and this can be observed on the negative x scale of the plot.

The parameter Δd identifies the set of possible source locations - a hyperbola. (see annex A.1) It consists of two branches whose focus is one of the receivers (figure 2.5), where the source is at the left branch when the TDoA is negative (otherwise it is located on the right branch), since Δd is the same as the hyperbola distance difference. As the source moves along the branch and far away from the source, it follows a straight path - an asymptote – as it can be observed in figure 2.6.

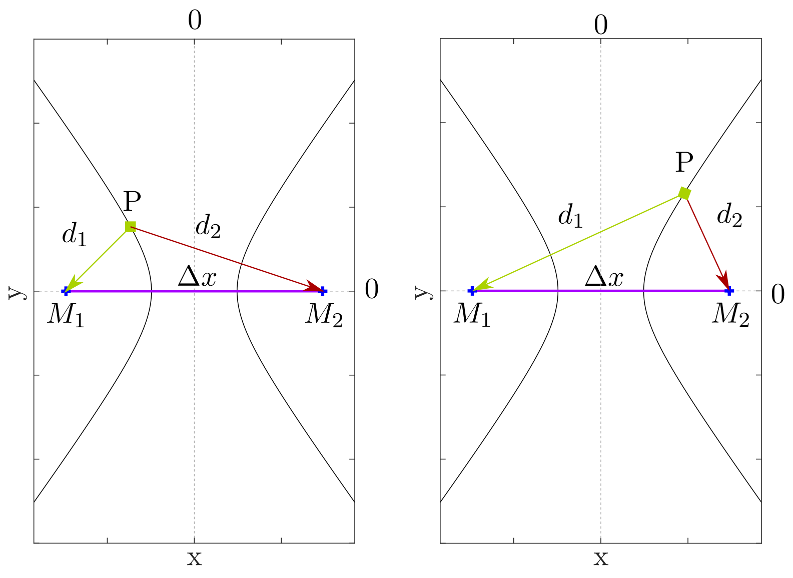


Figure 2.5: Source location on both branches of the hyperbola.

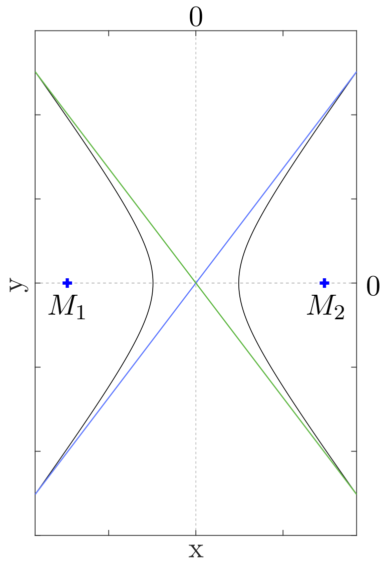


Figure 2.6: Asymptotes close to both branches of the hyperbola.

The main advantage of using an asymptote over an hyperbola is its complexity, having a linear behaviour which translates in less computation resources. However, there is an disadvantage: the closer the source is to the receiver the worse the prediction becomes. This is due the asymptote not following the branch in the near-field.

By changing the distance difference between both hyperbolas the branches change their direction (and shape), as it can be observed in figure 2.7. There are two things that stand out: the density near the horizontal line and the two hyperbola deformations at $\Delta d = 0$ cm and $\Delta d = \Delta x$ cm. Following the deformations, the acceptable Δd must be between $]0 \;\; \Delta x[$ cm. Furthermore, by analysing figure 2.1 it becomes easy to understand that the AoA is 180° when $\Delta d = -\Delta x$ cm and 0° when $\Delta d = \Delta x$ cm. However, while the asymptotes help in estimating the direction from where the mosquito comes from – AoA – it does not give its exact position in space, giving only the positions along the branch.

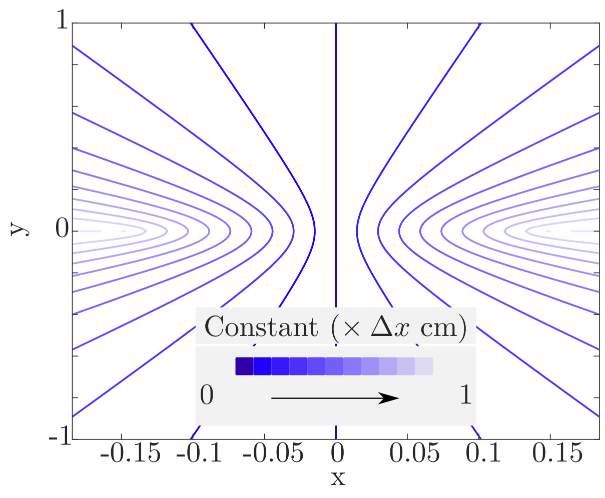


Figure 2.7: Family of hyperbolas by varying its constant value, ranging from 0 to Δx cm, where $\Delta x = 29.2$ cm.

A new solution to the problem consists of applying the far-field approximation to all sound waves (see in figure 2.1), which implies that all sources are located far away from the receiver hence allowing for the use of the previously mentioned asymptotes. The consequences are that the sound waves become parallel to each other, causing each wave to have the same AoA – as observed in figure 2.8.

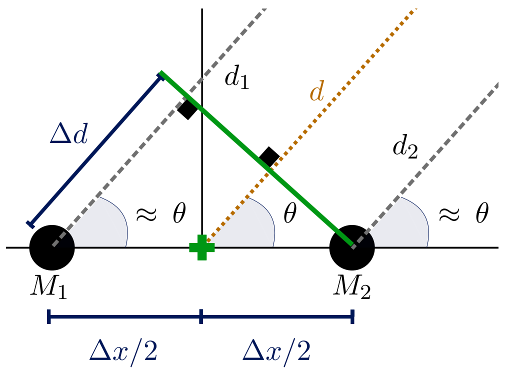


Figure 2.8: Far-field applied to the azimuth detection scenario.

The green circumference – the mosquito radius – is now approximated to a straight line, perpendicular to all distances d, d_1 and d_2. This creates a right triangle with a base of Δd cm and an hypotenuse of Δx cm.

2.1.2 Formula deduction

Taking advantage of figure 2.8 simplification, it becomes much easier to determine the cosine of an angle, which is the ratio of a triangle's base to its hypotenuse:

$$cos(\theta) = \frac{\Delta d}{\Delta x}. \tag{2.6}$$

Integrating the TDoA equation (2.2) and solving in order to θ, the AoA is given by the following equation:

$$\theta = \arccos \frac{c \times \Delta t}{\Delta x}. \tag{2.7}$$

The AoA reference is the same as figure 2.1, where its 0° when the mosquito is on the right, 90° when the mosquito is in the front and 180° when the mosquito is on the left.

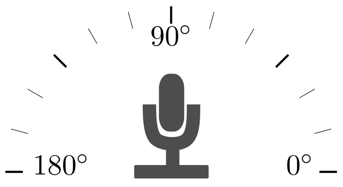


Figure 2.9: AoA reference for the system's receiver.

2.2 Algorithm limitations

So far the limitations encountered are derived from the properties of the hyperbola and from the far-field approximation applied to the incoming sound waves (from the flying mosquito). This made the problem easier to solve and the trigonometric equation less complex.

By studying an algorithm purely based on this equation is is possible to get conclusions about its limitations that have not been pointed out yet. Considering that a mosquito performs a semi-circular trajectory around the receiver, with a distance *d* of 5 meters, the algorithm has an error pattern of:

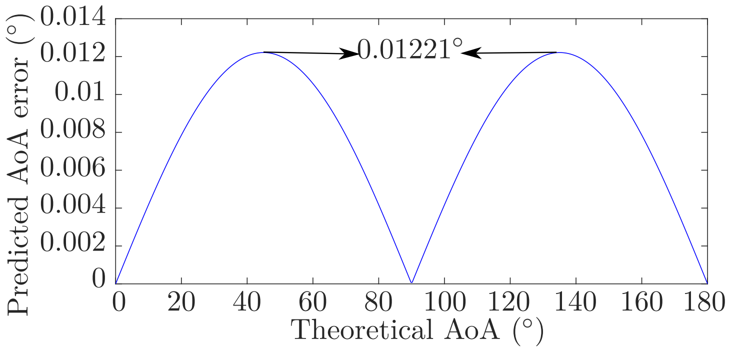


Figure 2.10: Pattern of the algorithm's AoA absolute error (5 m from the source).

This figure shows that the pattern has two peaks, one at 45° and other at 135°, meaning that the algorithm performs worse when the mosquito is located at these AoA values. To confirm the far-field approximation limitations, the mosquito now is located at 20 cm from the receiver:

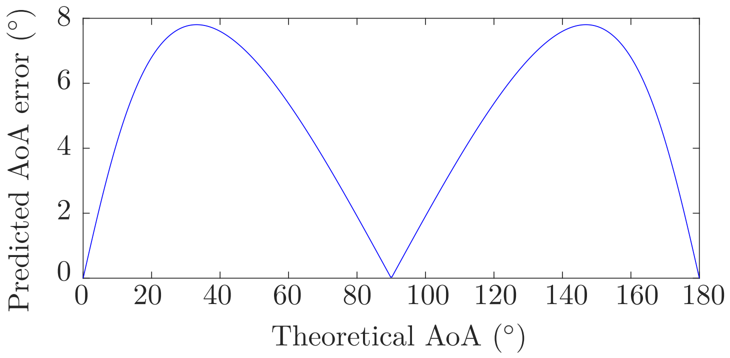


Figure 2.11: AoA error pattern when the source is close to the receiver (20 cm).

Not only the error amplitude is much higher, but the peaks are located at an AoA different from the previous ones (45° and 135°), with a shift of roughly 10°.

The algorithm sensitivity to both extremes is not shown on the figures because the values are truncated when plotting the results. This will be explored on later chapters.

2.3 Spatial parameters

The parameters in question are the ones related to space: the microphone spacing Δx – derived from the receiver's properties – and d – which is the distance from the receiver to the flying mosquito that is being tracked.

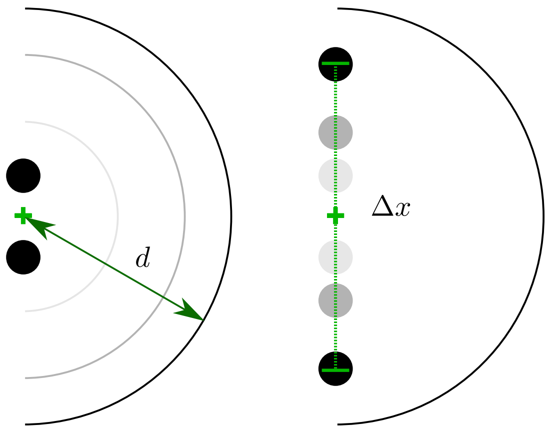


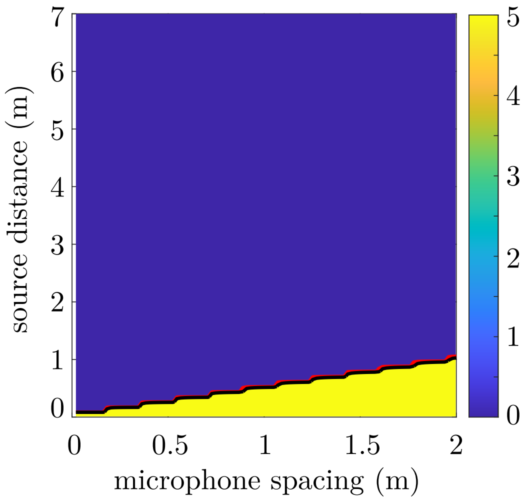
Figure 2.12: Visual representation of the spatial parameters: distance d on the left and microphone spacing Δx on the right.

Similar to the previous test, the mosquito flies a semi-circular trajectory around the receiver, with the intent of creating an AoA range of $[0 \;\; 180]^\circ$ to the output of the algorithm while maintaining the same distance.

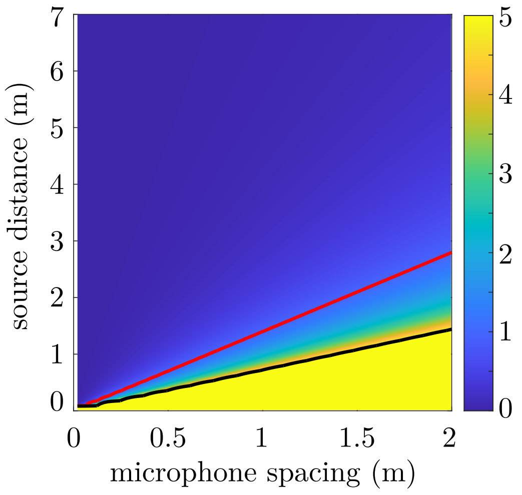
2.3.1 Colour maps

The best way to demonstrate the impact of the spatial parameters is by using a colour map, where different colours indicate the magnitude of the AoA prediction error. The map has a red curve overlapped, which indicates the part of the map where the error is equal to 1° and a black curve overlapped, indicating an error of 5°.

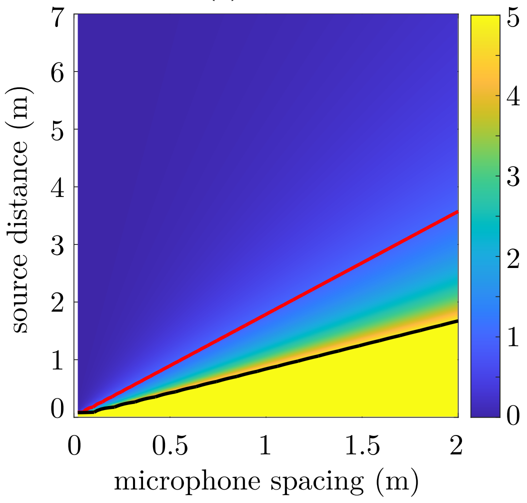
Taking a look at the error pattern in figures 2.10 and 2.11, it becomes clear that there is a symmetry at an AoA of 90°. If the TDoA (or time delay) is estimated without any error, the AoA range of $[0 \;\; 90]^\circ$ is enough to take conclusions about the parameters. The tests are made with steps of 15°, starting from 0° and ending in 90°, showing a specific case in 135° to check the error symmetry. All the colour maps that demonstrate the AoA error pattern with no measurements errors can be observed in figures 2.13 and 2.14.



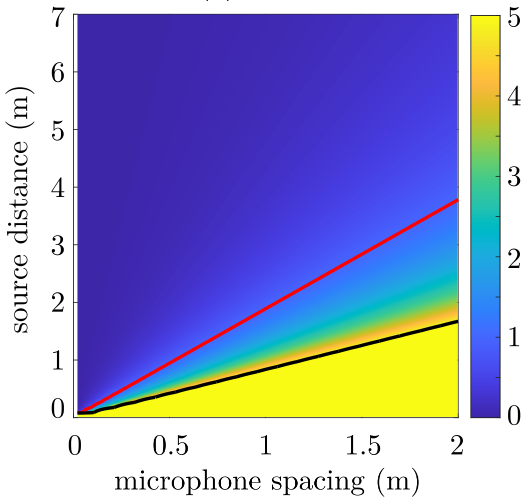
(a) $\theta = 0°$



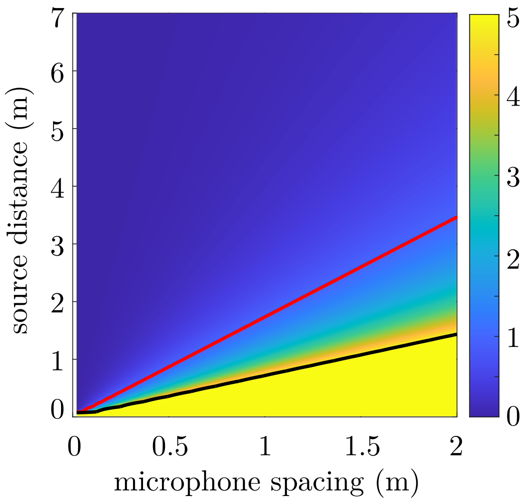
(b) $\theta = 15°$



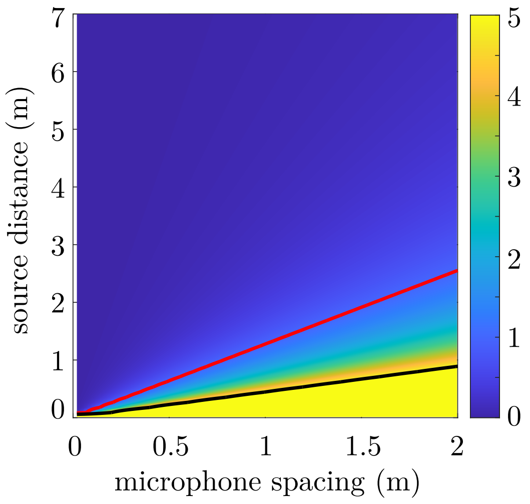
(c) $\theta = 30°$



(d) $\theta = 45°$



(e) $\theta = 60°$



(f) $\theta = 75°$

Figure 2.13: AoA error colour maps when the mosquito goes from 0° to 75°.

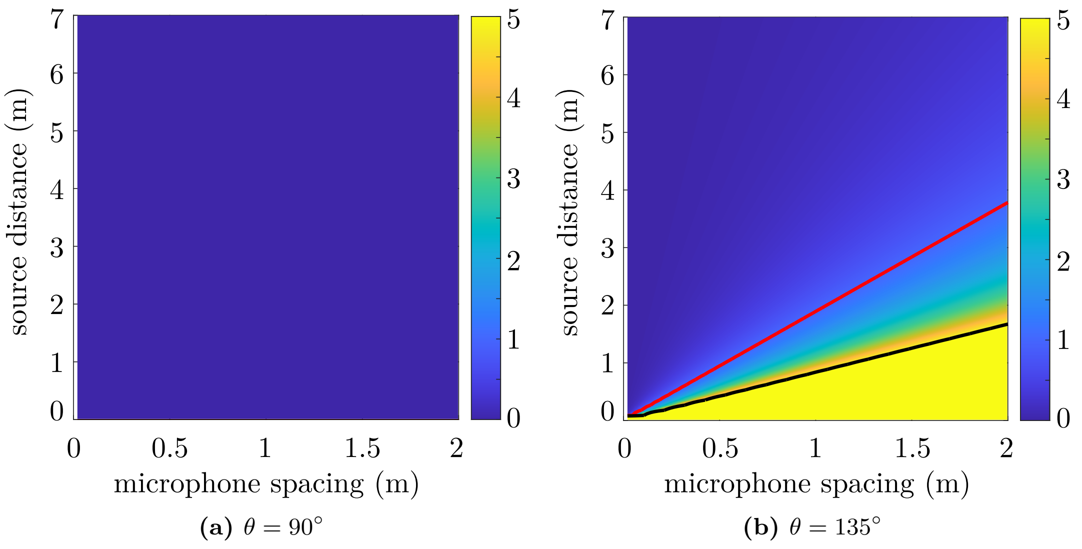


Figure 2.14: AoA error colour maps when the mosquito goes from 90° to 135°.

2.3.2 Analysis

The error pattern symmetry can be observed on figures 2.13d and 2.14b, when the mosquito is at 45° and 135° from the receiver. Both AoA are shifted 45° from the centre (90°) and produce the same error values.

When the mosquito flies near the extremes of the receiver, as seen in figure 2.13a, the algorithm predicts with an error of less than 1° when the distance is approximately

$$d > \frac{\Delta x}{2}. \tag{2.8}$$

Stepping in 15° the red curve changes drastically, meaning that the distance d required for a good predictor behaviour also increases:

$$d > \frac{3\Delta x}{2}. \tag{2.9}$$

This distance d will increase to its maximum value when the mosquito is located at 45° from the receiver, as it can be observed in figure 2.13d. This maximum value is roughly the double of the microphone spacing distance:

$$d > 2\Delta x. \tag{2.10}$$

As the mosquito continues to follow the trajectory, the minimal distance required decreases until it reaches 90°. This is the predictor's best AoA because it doesn't need a minimal distance and the predicted result is always precise – see figure 2.14a.

If the predictor is configured to have a prediction error of 5°, the spatial requirements for the receiver lower a little (see figure 2.15). This can be concluded by analysing the black curve of the colour maps above, where the distance d gets his maximum value at approximately

$$d > \frac{3\Delta x}{4}. \tag{2.11}$$

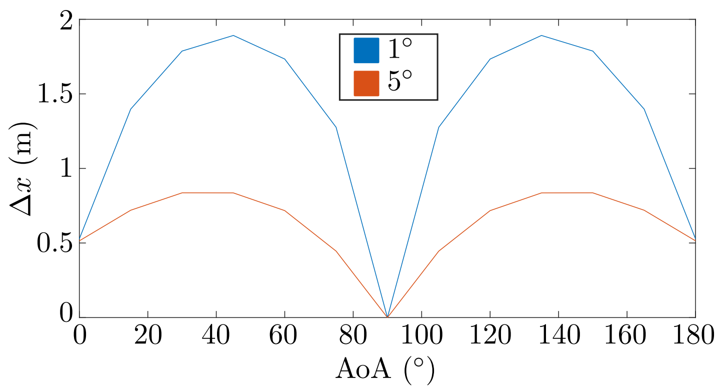


Figure 2.15: AoA error curves and minimal distances for Δx: 1° of error (blue curve) and 5° of error (orange curve).

2.4 Delay estimation

Until now it has been considered that the TDoA is measured without any errors. This is not always the case because there is always some residual errors from the measurement, being one of them dependent on the sampling frequency of the recordings and the other from the possible sound waves reflections. This error is given by the letter ϵ and is represented in seconds, being either negative or positive. By adding this delay estimation parameter to the main equation in (2.7), the equation becomes:

$$\theta = \arccos \frac{c \times (\Delta t \pm \epsilon)}{\Delta x}. \tag{2.12}$$

A delay estimation affected by a sampling frequency error of N samples can be calculated by the following way:

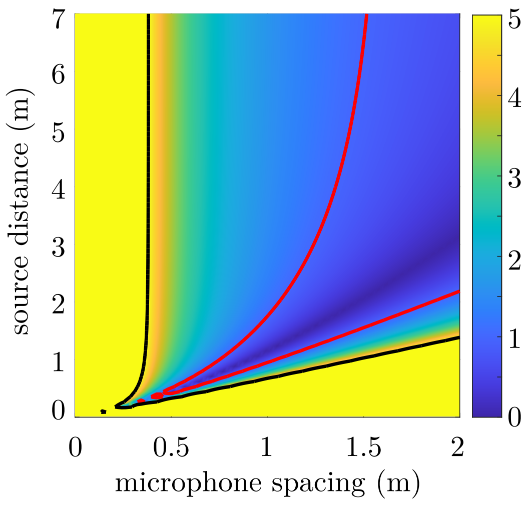
$$\epsilon = \frac{N}{f_s}, \tag{2.13}$$

and this value alone can affect the precision of the algorithm, since it destroys the symmetry of the AoA error pattern at 90°.

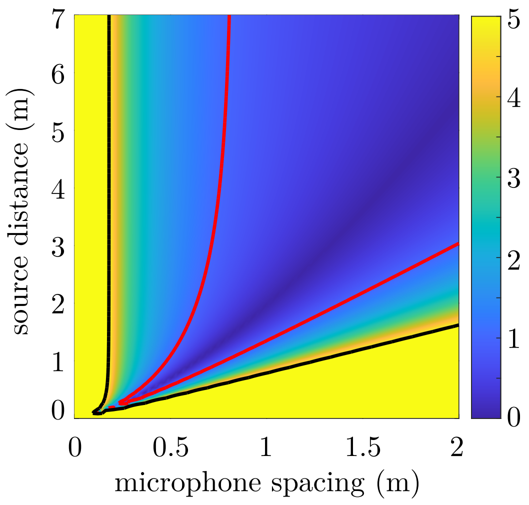
Other possibility is the presence of sound reflections, which may interfere in the cross-correlation of the samples by flattening its maximum curve (the peak is now undistinguishable from other values) or by creating fake spikes (maximums) – resulting in a corrupted time delay value.

2.4.1 Colour maps

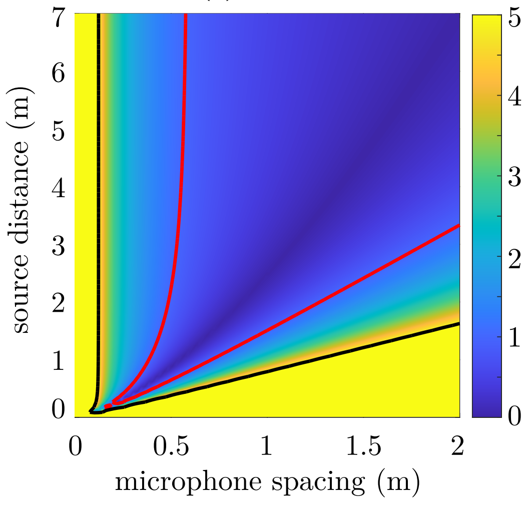
It is considered that the receiver captures sound waves at a rate of 48 KHz, which is the lowest sampling rate accepted by predictor thus, making this the best scenario to analyse since the other sampling frequencies will have a lower error. Following equation (2.13), the ϵ is equivalent to +21 μs and the respective colour maps for the AoA error pattern are presented in figures 2.16 and 2.17.



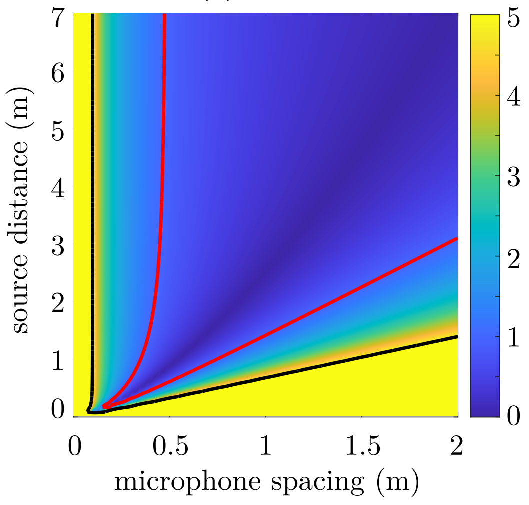
(a) $\theta = 15°$



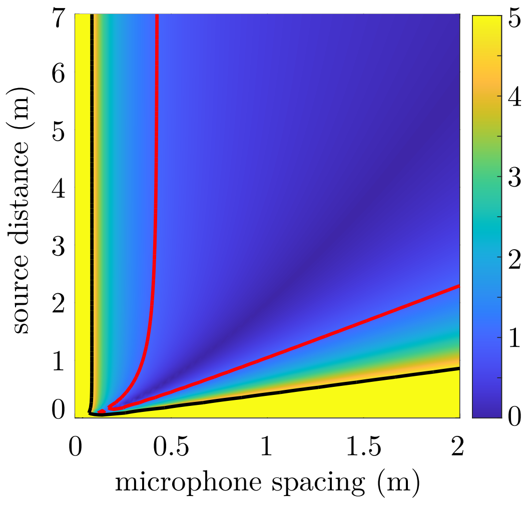
(b) $\theta = 30°$



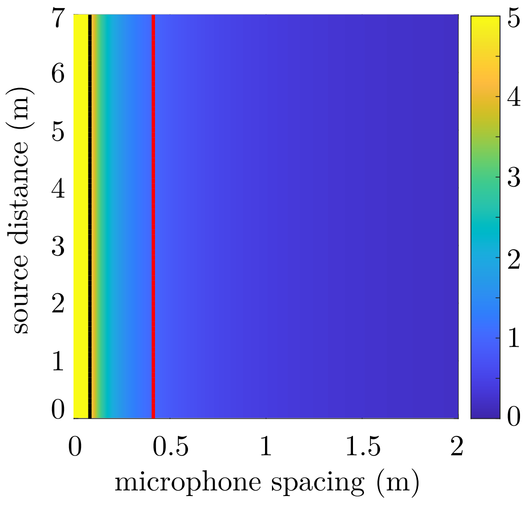
(c) $\theta = 45°$



(d) $\theta = 60°$

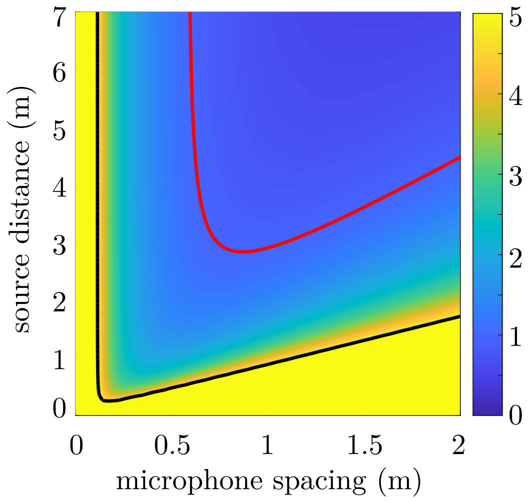


(e) $\theta = 75°$



(f) $\theta = 90°$

Figure 2.16: AoA error colour maps when the TDoA estimated has an error of $\epsilon = +21\ \mu s$: mosquito flying from 15° to 90°.



(a) $\theta = 135°$

Figure 2.17: AoA error colour maps when the TDoA estimated has an error of $\epsilon = +21\ \mu s$: mosquito flying from 135°.

2.4.2 Analysis

There is no longer a symmetry at $\theta = 90°$, which can be observed in figures 2.16c and 2.17a when comparing the AoA error in 45° and 135°. This measurement error affects the receiver's spatial configuration due to the fact that now it is imposed a microphone spacing restriction.

When the mosquito is located at $\theta = 15°$ (see figure 2.16a) which is very close to one of the extremes of the receiver, it can only be detected with an error of 1° if the microphone spacing is bigger than approximately 1.5 meters. It is also worth noting that the minimal distance to detect the mosquito has been lowered, to almost

$$d > \Delta x, \tag{2.14}$$

which is lowered by $0.5\Delta x$ when compared to the value discussed in the previous equation (2.9). In other ranges below 90°, the minimal distance is also lowered.

This restriction to the spacing of both microphones decreases when the mosquito flies to narrower ranges i.e. close to the front of the receiver. When the mosquito is located at $\theta = 90°$ is when the restriction reaches its lowest value, of approximately 40 cm. The opposite can be observed when $\theta = 135°$, which is when the mosquito is located at the other side of the receiver (equivalent to 45° when there is no TDoA measure errors, as seen in figures 2.13d and 2.14b). In this case the minimal distance increases as does the microphone spacing restriction: Δx needs to be bigger than approximately 80 cm.

The receiver (see figure 2.18 for its orientation) has a side where it predicts the mosquito more inaccurately and is totally dependent on the polarity of the epsilon value. If the epsilon was negative in this analysis, the conclusions made to the AoA of 135° would be made to the AoA of 45° and vice-versa, since the analysis would be made on the inaccurate side.

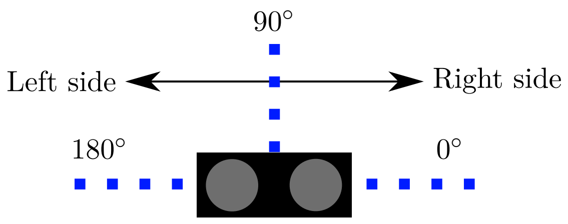


Figure 2.18: Orientation of the receiver structure.

2.5 Optimal design

So far the analysis points out distances that the receiver can detect without any significant AoA error, avoiding the near-field problem generated by the far-field approximation applied to all sound waves. It was also seen that a single sample error from the sampling frequency being used can completely change the error pattern studied in figure 2.10, causing one more restriction – in the microphone spacing parameter.

The receiver is a simple structure with two microphones, capable of recording and processing the sounds of the room where they are placed. It must have a configuration that allows for a good prediction accuracy without occupying too much space, which is the main challenge of this section due to the fact that wider microphone configurations generate more precise results, as seen in the previous colour maps.

When implemented in the corner of a room, the microphone spatial restrictions lower, allowing for a more compact configuration. This is due to the detection range narrowing down to a range of [45 135]° which in consequence lowers the Δx needed to have a lower prediction error.

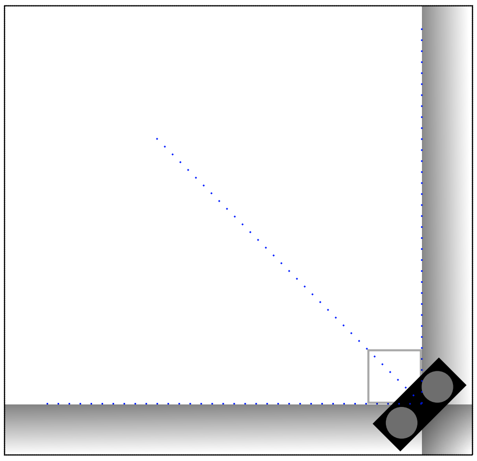


Figure 2.19: Receiver implemented on a corner of a room.

To choose the receiver's parameters the worst case scenario must be taken into account, to guarantee that it covers all the problems. This can be done by analysing figure 2.17a, which is located on the inaccurate side of the receiver (in that particular case the error was of $+21\mu s$ and it affected more the AoA range of [90 180]°) and needs a bigger distance d in order to predict with a lower error (see figure 2.15).

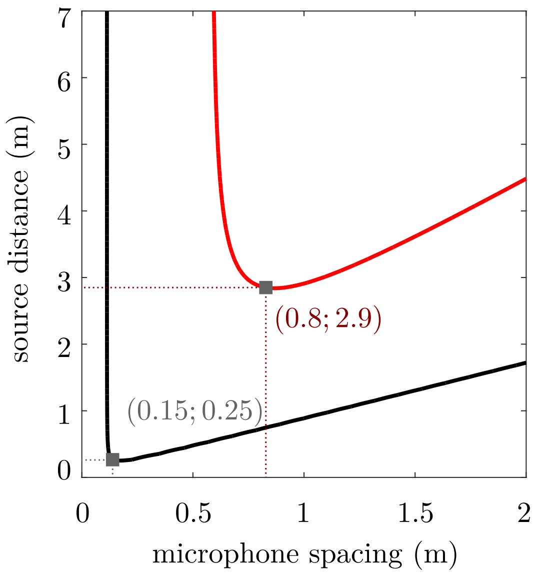


Figure 2.20: Detailed analysis for the error curves at $\theta = 135°$ – with an error of $\epsilon = +21\mu s$.

For a strict predictor, where the error must be no more than 1°, $\Delta x = 80$ cm. On the contrary, a more forgivable predictor where the error is around 5°, $\Delta x = 15$ cm. This shows that for a more accurate predictor the receiver has to be wider. However, a qualitative information about the whereabouts of the mosquito is enough in most cases, which allows for a more compact receiver. The final remark about figure 2.20 is that wider configurations need the mosquito to be located farther way than usual: 2.9 meters for an error of 1° and 0.15 meters for an error of 5°.

Going back to the workaround in figure 2.19, it becomes apparent the presence of shadow zones outside of the range [45 135]°. This shadow zones is where the predictor becomes inaccurate since the microphone spacing attributed to the receiver was based on the data of the narrower range. A solution for the problem caused by this workaround is by implementing other receiver in the opposing corner.

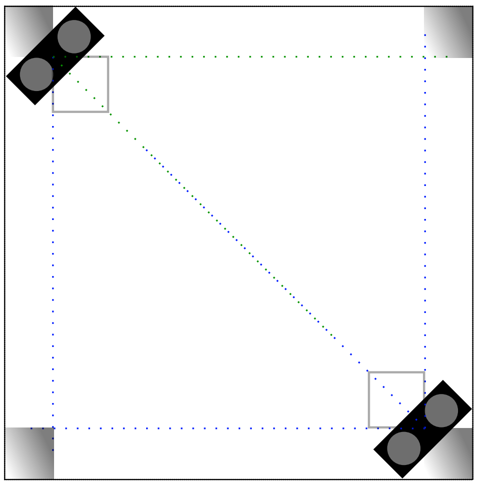


Figure 2.21: Two receivers in opposite directions.

There are shadow zones located on the back of the receiver, but those can be ignored since the mosquito will not go there. Now it remains two shadow zones located on the rest of the corners of the room. There are numerous ways to solve this, aiding the result with the history of the mosquito AoA predictions is one of them.

Implementing the ideal receiver while using only the azimuth plane becomes a trade-off game. A more accurate predictor causes the receiver to be wider while reducing the accuracy causes the receiver to be more compact. However, wider configurations start detecting the mosquito at longer distances to keep a lower prediction error while compact configurations start detecting at shorter distances.

CHAPTER 3

Implementation

The majority of the work done for this thesis is based on simulation. While the result figures and values are what matters in the end, the understanding of the simulator (and associated tools) being used is also important. This way the whole experiment can be reproduced easily.

3.1 Simulator

3.1.1 Origin

This experiment uses the "Ultrasonic Room Simulator", a tool created by my co-advisor Daniel Albuquerque. It was implemented for MATLAB and can be accessed freely on the MathWorks File Exchange [40]. It was sightly adapted for the context of the objectives and some applications were built around it.

3.1.2 Overview

All the components that make up the simulator are all thoroughly detailed in the paper “Indoor acoustic simulator for ultrasonic broadband signals with Doppler effect”[41]. This subsection only gives a very simplified overview of the simulator, by explaining how it works and what (and why) it has been changed.

The original architecture, which is located in figure 3.1, has three main stages - the transmitter block (ultrasonic transmitter), the room channel and the receiver block. The transmitter receives an electric signal x_m which is transformed intro an acoustic wave so it can travel through the Room Channel, where it propagates before arriving at the receiver. The receiver combines each received wave into a single signal, y_n.

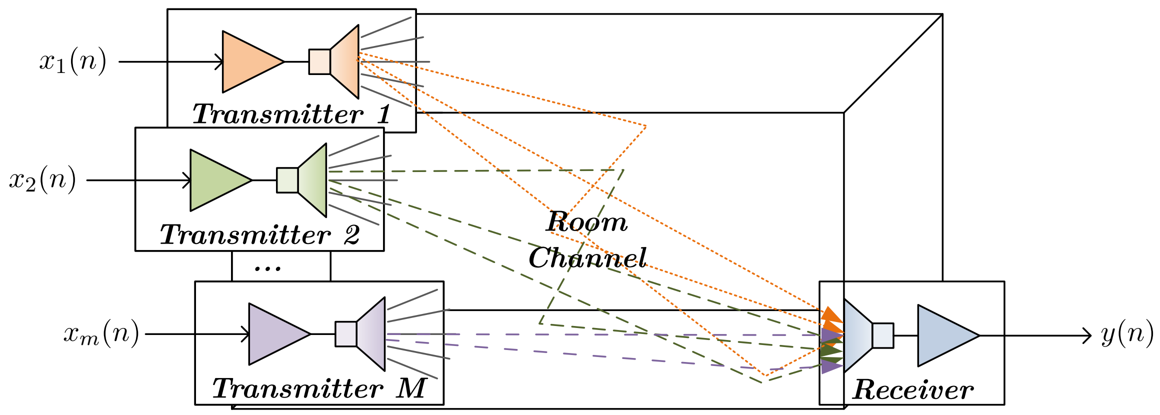


Figure 3.1: The original Ultrasonic Room Simulator architecture – adapted from [41].

Due to the fact that two microphones are being used, it would be convenient if the simulator could implement more than one receiver. This is done by inverting the architecture in 3.1 in a simple way: one transmitter but various receivers, as shown in figure 3.2. A single electric signal $x(n)$ enters into the only transmitter available, where it propagates in the Room Channel arriving at the inputs of various M receivers. Each receiver combines its received waves into a $y_m(n)$ signal, having a total of M output signals.

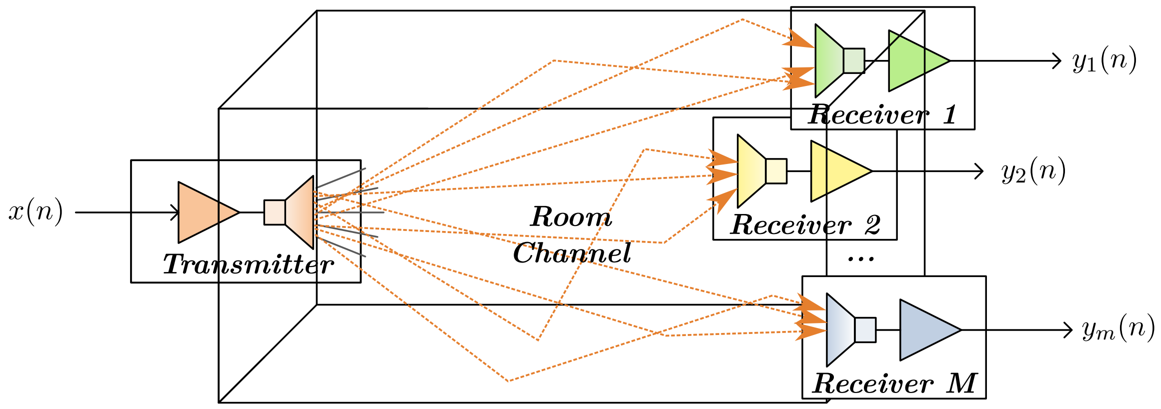


Figure 3.2: The modified Ultrasonic Room Simulator architecture.

The block diagram in figure 3.3 explains what happens in the simulator. Each $h_m(n)$ block represents the Impulse response (IR) that is applied to the signal $x(n)$ before reaching each receiver. It is composed by three IR: the transmitter $t(n)$, the sound propagation $p(n)$ and the receiver $r_m(n)$. This chain is demonstrated in figure 3.4.

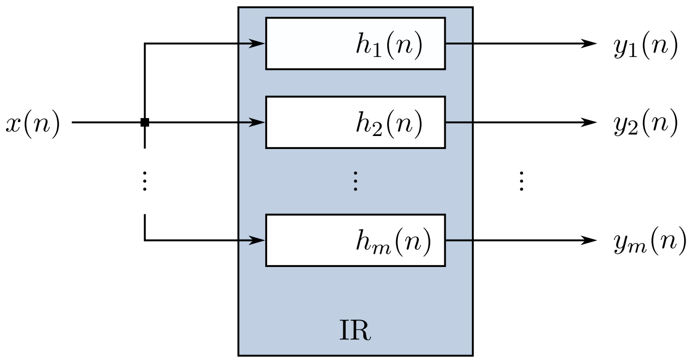


Figure 3.3: Block diagram of the modified simulator.

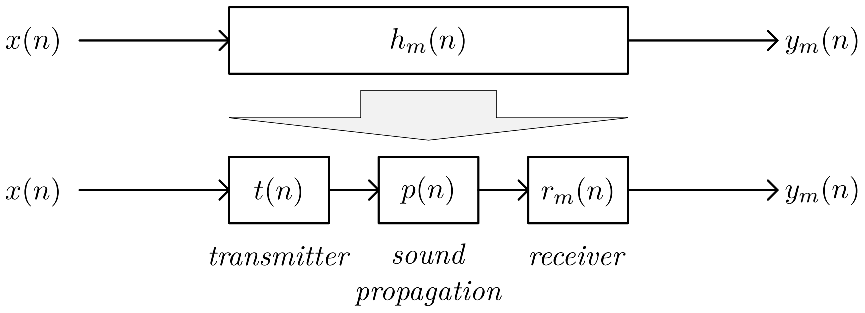


Figure 3.4: Detailed IR block of figure 3.3.

The transmitter $t(n)$ models the acoustic world for a digital signal and the receiver models the digital signal of the acoustic wave that has propagated inside the Room Channel. The middle block – the sound propagation $p(n)$ – models the acoustic wave in function of the room characteristics i.e. wave distance, delay, attenuation, etc., hence why the multiple IR responses.

3.1.3 Modelling

When the sound waves exit the transmitter, they propagate through a medium called Room Channel. It does not matter if this medium is in an open space or in a closed room, it needs to be modelled physically while respecting the characteristics of the materials. So, it is necessary to model the surfaces that form the shape of the room (or the floor in case of an open space) and the reflections within them.

Due to the fact that quadrilateral surfaces are less complex and simpler to implement, the simulator then uses a polygonal based modelling where the basic element of construction are quadrilaterals and not triangles – the standard of Three-dimensional (3D) modelling. Nevertheless, the shape of a room can be defined by a group of quadrilaterals, as demonstrated on figure 3.5.

The sound waves that are emitted travel radially in all directions, which means that some travel directly to the receiver while some of them are reflected on the surfaces until they reach the receiver or loose their energy. A surface modelled as quadrilateral is flat and an incident wave is reflected into a single wave like a mirror-like reflection – also known as specular reflections. Figure 3.6 demonstrates how the waves are propagated.

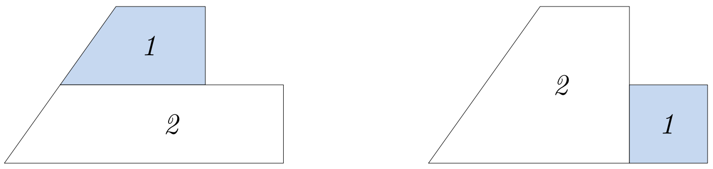


Figure 3.5: Possible divisions of a complex shape into two quadrilaterals – adapted from [41]

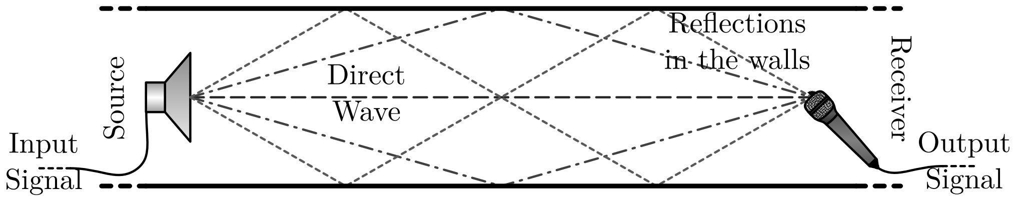


Figure 3.6: Typical sound wave propagation in a room that only has two surfaces (in this case, a floor and a ceiling) – adapted from [41].

Snell's Law describes the process on how wave are reflected (see figure 3.7). The refracted waves can be ignored however, the energy of the reflected wave is not totally equal to the energy of the incident wave, since some of it goes to the refracted wave. That being considered, the reflected wave energy is mostly dependent on the surface material.

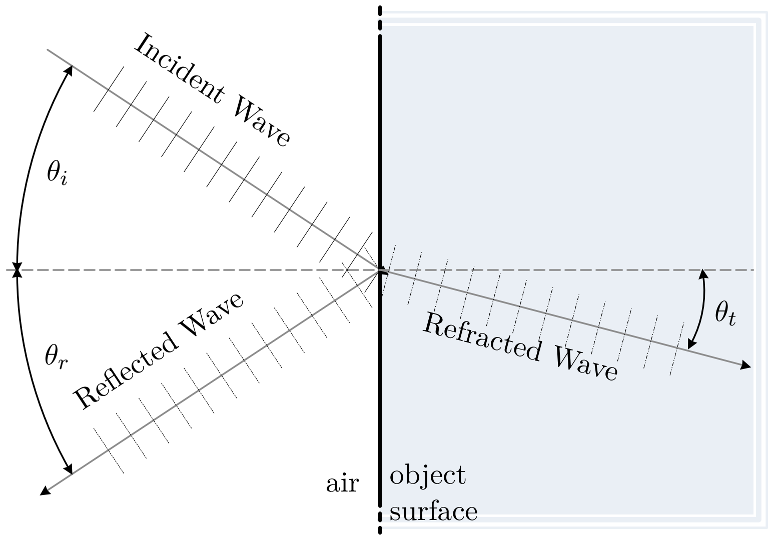


Figure 3.7: Sound wave reflection and refraction (Snell Law) – adapted from [41].

In the simulator each reflection is modelled by the Virtual Source Method. While the simulator was implemented for ultrasonic waves it works perfectly well with audible sound waves (frequencies below 20 kHz). It has a good accuracy at higher frequencies. It states that when a wave hits a surface it creates a virtual source on the other side (inside or through the surface). Applying the Snell's law it is possible to locate the virtual source, visualized in figure 3.8. The virtual source is located at the same distance d as the source, where the refracted angle is equal to the incident angle.

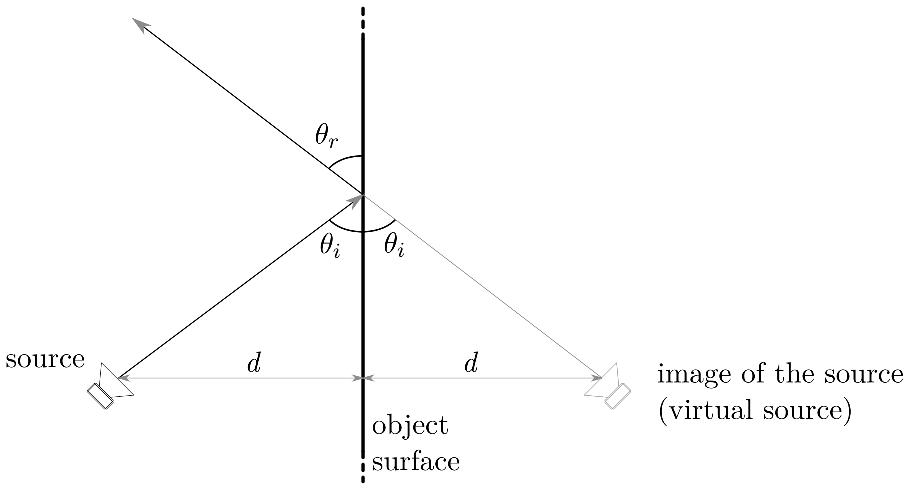


Figure 3.8: Virtual Source Method principle (with Snell Law applied) – adapted from [41].

This method is very accurate for the typical room shape, but its complexity is linear to the number of surfaces. With this, the number of virtual sources needed for a room is given by the already deducted formula

$$N_T \approx (N-1)^r, \tag{3.1}$$

where N_T symbolizes the number of virtual sources, N the number of different surfaces and r the reflection order. When the number of surfaces increase so the number of virtual sources. However, this starts to become computationally expensive when the reflection order increases which imposes an indirect limitation: the room model must be simple (rectangular e.g.). To better understand the impact of this formula, the plot tendency at figure 3.9 demonstrates how the simulation time is affected by these two variables.

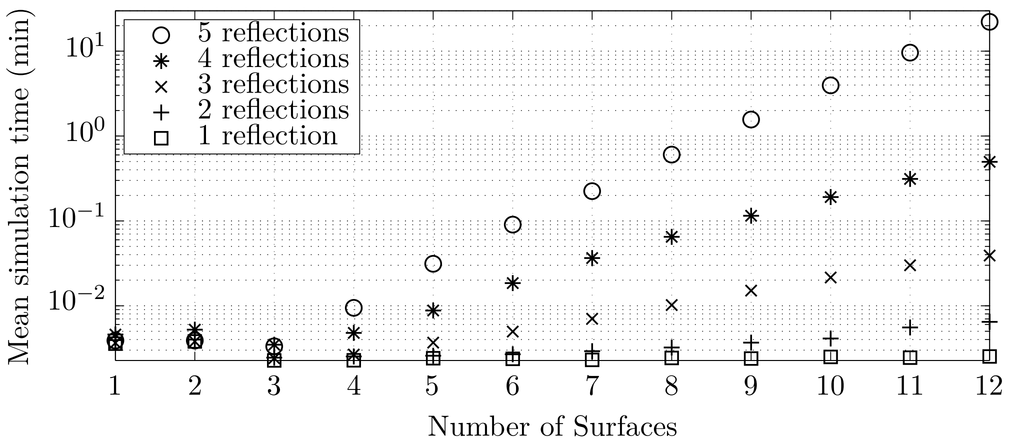


Figure 3.9: Simulation time as a function of the number of surfaces and the MR – adapted from [41].

Nevertheless, knowing the location of the receiver allows the simulator to remove certain virtual sources that do not contribute to the received signal thus reducing some complexity of the method. An example of a room with a reflection order of one can be seen in figure 3.10 while an reflection order of two can be seen in figure 3.11.

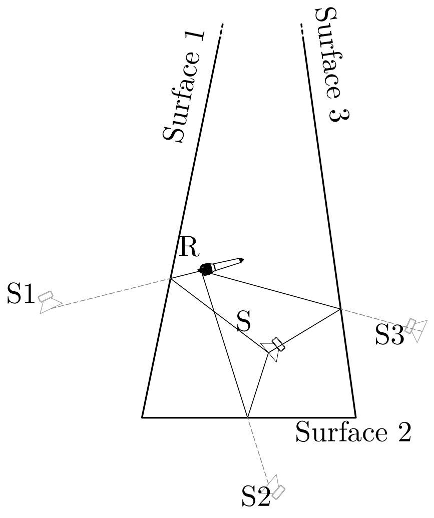


Figure 3.10: Example of a first order reflection on virtual sources – adapted from [41].

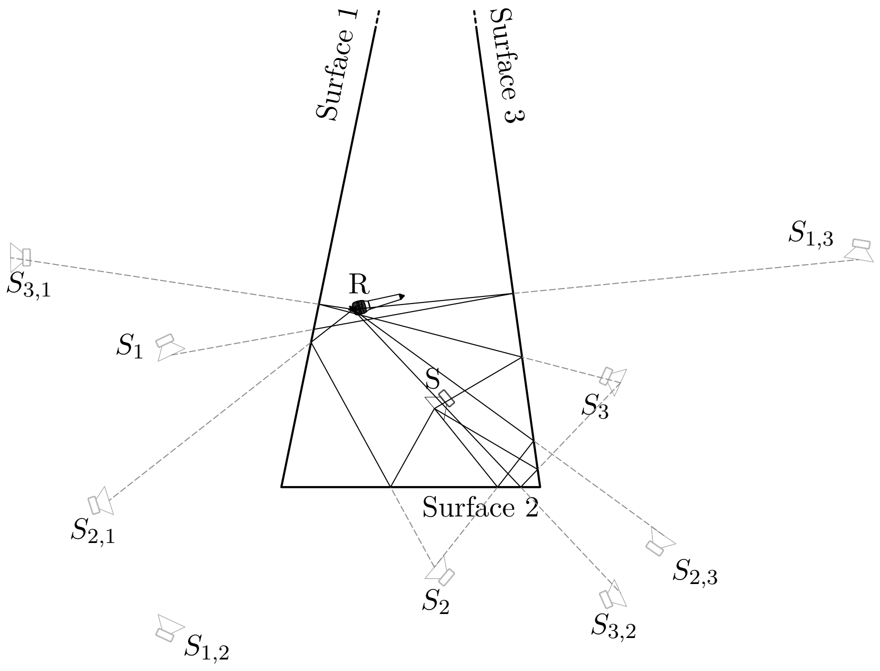


Figure 3.11: Example of a second order reflection on virtual sources – adapted from [41]
.

In figure 3.11, besides the addition of virtual sources created by the refections of the first order virtual sources there is also the presence of a non-contributing source: $S_{1,2}$. The full criteria and details of the accepted sources are thoroughly explained in the paper [41].

3.1.4 Limitations

By analysing how the simulator was modelled, some limitations stand out while other are implicit. That being said, the main limitations are:

1. The shape of a room must be simple by default;
2. Refractions and diffractions are not supported;
3. Reflection order must be small.

A room must be simple i.e. with an rectangular profile because each surface is modelled with quadrilaterals and if one tried to create a curved surface, it would require a huge number of surfaces and that would affect the performance of the simulator – limitation 1.

Now considering that the limitation 1 is obeyed, one must understand the specifications of the simulator and what is capable of. As discussed in 3.1.3, all reflections are of specular nature and this implies that there are no diffractions. Furthermore, as it states that all incident sound waves are absorbed by the surface's material, there are no diffractions occurring – limitation 2.

Finally, when the first two limitations are taken into consideration, the user must be careful when specifying the Multiple reflections (MR) order: when using the virtual source method, even in a simple room configuration with six surfaces, if the reflection order is high enough the simulator would need to calculate a gigantic number of virtual sources and it would be very slow (equation (3.1)). Not that it is not possible, only that it would take a long time – limitation 3.

3.1.5 Implementation

The simulator is modular which means that anyone can easily modify certain simulator behaviours without the risk of damaging it. It is formed by 6 modules and 7 methods, where a module is an object and the method serves to gather information from the other modules or to even create a new one. The changes made to the implementation are based on how the receiver and source are implemented, now being able to implement an array of receivers and a single source (as opposed to the implementation of the original simulator [41]), as it can be seen on figure 3.12.

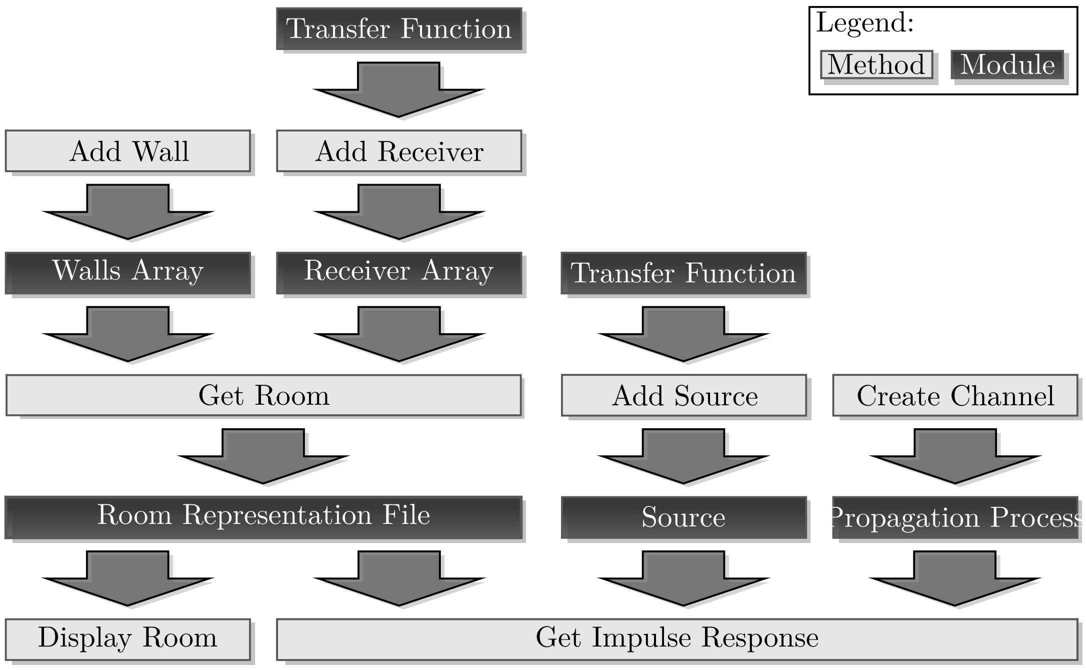


Figure 3.12: Modules and interaction between them (methods) that are embedded in the modified simulator – adapted from [41].

When designing a room for simulation, the first thing that comes to mind is to design the room itself i.e. its shape and inside surfaces that are relevant to the experiment. This can be done with the module *Walls Array* which is basically a structure that contains the four points (P1 to P4, respecting the sequence order) of a single (quadrilateral) surface along with its respective reflection coefficient (RCf). To call this module simply use the *Add Wall* method, where the user inputs all the necessary data for a single wall/surface structure (four points and reflection coefficient) followed by an empty array for the structure field (`AW=[]`) when defining the first surface or by the previous surface structure (`AW=previousAW`) when defining the rest of the surfaces.

```
function AW = addWall(AW, P1, P2, P3, P4, RCf)
```

After modelling the room, the user specifies the speaker and the receiver(s). However, these two modules need a dedicated transfer function module whose purpose is to output the IR (h) for specific azimuth (theta) and elevation (phi). To implement this module it is necessary to input the sampling frequency (fs) and the frequency range (from *fi* to *fe* Hz).

```
function [h,p] = transfer_function(theta, phi, fs, fi, fe)
```

There is also another output, p, which represents the index of the first sample. This is not important because it is not used for this experiment.

Moving on, both receivers (microphones) and source (speaker) are implemented in a identical way. As mentioned in the beginning, the original simulator's version was modified to implement multiple receivers but only a single source. This means that the receiver method (to implement the receiver module) behaves in a similar way to the *Add Walls* method, where it needs the previous structure when defining multiple receivers. That being said, the field necessary to produce a structure is the position in space (C: x,y,z coordinates). Other fields such as the transfer function handle (Tf: use @transfer_function) and radiation direction (D) are optional and in case the user omits them, the default is omnidirectional for both of them. To generate the speaker structure (S) the method *Add Receiver* is called (same as Add Speaker – *addSpk*).

```
function S = addSpk(C, Tf, D)
```

For the microphone structure (AM), this input field is empty when defining the first microphone and it uses the previous structure when defining the rest of them. So, using the method *Add Receiver* (same as Add Microphone – *addMic*):

```
function AM = addMic(AM, C, Tf, D)
```

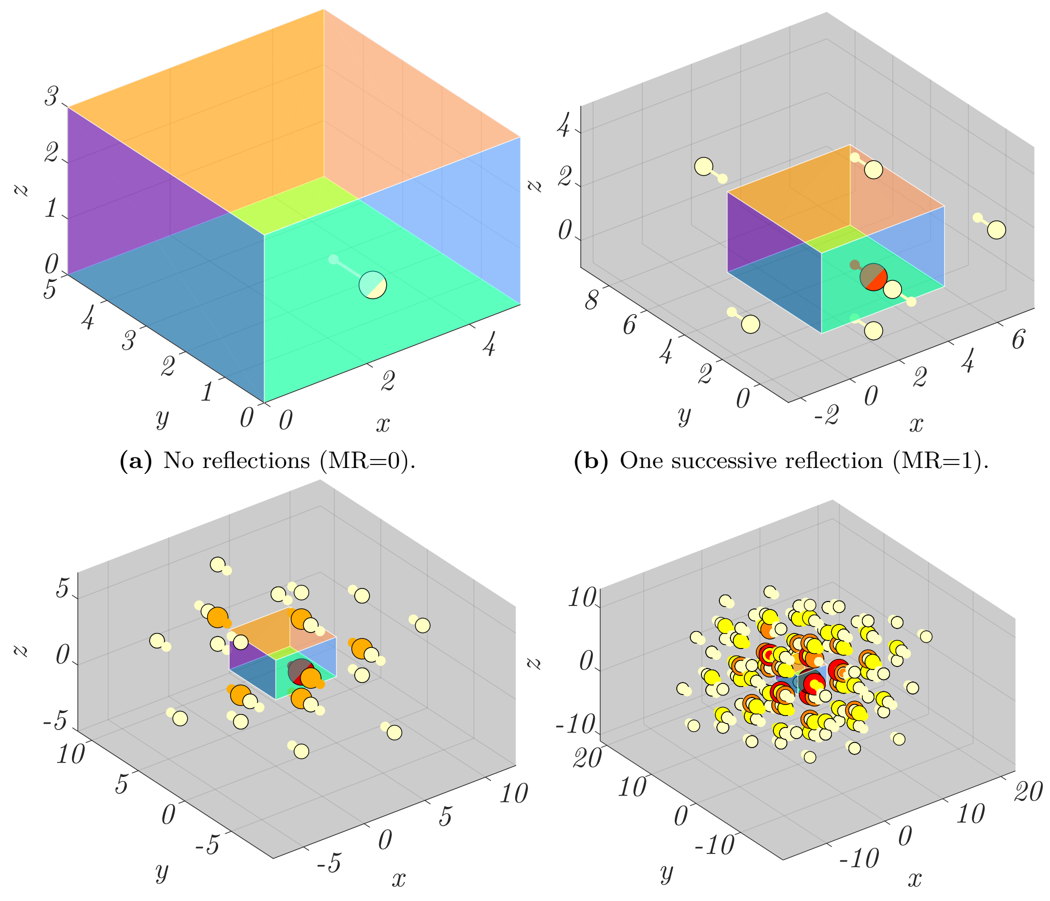
The middle step is to implement the *Room Representation File* module, which will later be needed to obtain the IR. Besides being obligatory to the simulation process, this module is useful to visualise the modelled room with the embedded method *display room*. Furthermore, to originate this file the constructor requires the structures created in *Walls Array* (AW) and *Receiver Array* (AM), the frequency bounds for frequency response optimization ([fl fh] Hz), the sampling frequency (fs) and finally, the maximum reflection order (MR) imposed to the room. While the original version uses an array of sources (AS), this constructor uses an array of microphones.

```
function makeFile(Name, AW, AS, MR, fs, fl, fh)
```

To use the method *Display Room* the user must specify the *Room Representation File*. There is only one operation (op) available for this method and it is to hide the virtual receivers (with the keyword `'HideVS'`). If the user only types one argument, by default it will show the virtual receivers (`op` is omitted in the arguments).

```
function [f,a] = displayRoom(Ffile, op)
```

An example of the usage of this method can be seen on figure 3.13, where there is a single receiver in a rectangular room. Similar to the original implementation, all the surfaces created are shown whereas virtual receivers are shown instead of virtual sources. In the initial room at sub-figure 3.13a there are no reflections and so the receiver is symbolized as an yellow sphere.



(a) No reflections (MR=0).

(b) One successive reflection (MR=1).

(c) Two successive reflections (MR=2).

(d) Four successive reflections (MR=4).

Figure 3.13: Graphic representation of the modelled room (using the *Display Room* method).

Going back at figure 3.4, the remaining block left to model is the *sound propagation* which corresponds to the module *Propagation Process*. Its main purpose is to give the IR for a given distance to the source (d), while depending on the ambient conditions such as temperature (T), relative humidity (H) and pressure (P) and on the frequency domain, such as the frequency operation band ([fl fh]) and sampling frequency (fs).

```
classdef Room < handle
```

This class creates a structure with pre-defined values whose values can be edited by the user.

At last, the most useful output of the simulator: the IR. By using the *Get Impulse Response* method, which uses all the modules described until now, it outputs an array of IR where each row is attributed to different receivers, whereas in the original version of the simulator each row represented different sources. The method is also known as *impR*.

```
function I = impR(file, R, C)
```

The variable *file* represents the *Room Representation File* module, C represents the *Propagation Process* resulting structure and R, which used to represent the single receiver, now represents a single source.

Having all the implementation blocks explained it allows the user to create a custom simulation environment with ease. Just to remind, this is a strategic summary of what has been written in the article of reference [41], giving enough information on how to use the (modified) simulator.

3.2 Custom applications

Due to the fact the simulator is modular thus being easily modifiable, some apps emerge from it (forked apps). Some apps are made to increase the productivity of the simulator, others have other functionality where the simulator is part of its core, etc. That being said, in here every app made using the simulator is going to be thoroughly detailed.

3.2.1 2D STS

Overview

Its name stands for "2D spatial trajectory simulation", where the user defines a 2D trajectory over a 3D modelled room and it outputs the respective spatial sound file. This application is basically an enhancement to the simulator interface, going all the way from configuring every parameter manually by a function in the command line to configuring every parameter in a Graphical User Interface (GUI) style, where the user can even create a trajectory while moving the mouse. While this feature appeals new users to the simulator, its main purpose is to increase the experiments productivity by reducing the configuration time.

Given the fact that the simulator was implemented in *MATLAB*, the interface was created using the same environment, specifically with the embedded tool *App Designer*[42]. For now the app can only be executed inside the Integrated Development Environment (IDE) and not in stand-alone mode because in the implementation there is a method that adds folders to the search path – and *MATLAB Compiler* does not support that feature.

Usage

Whenever a user wants to create a simulation it must follow the steps shown in figure 3.14, which covers the majority of the scenarios of the application.[1] This flow chart can be accompanied by the tutorial demonstrated in figure 3.15, giving a more complete guide to the user. Going back to the flow chart, there are three main decision markers which symbolize the main stages:

1. Trajectory definition;
2. Interpolation of trajectory;
3. Reconstruction of spatial sound.

Before even going through the definition of the trajectory, the application asks the user where the main folder is located – "Localizador-Acustico", which contains all the functions and structures necessary to run everything. There are also more assert warnings implemented, however since it is more important to create a simulation scenario, this is going to be demonstrated in the appendix.

Resuming the analysis of the flow chart in figure 3.14, to define a trajectory its necessary to have some structures defined, such as the room and receiver. When a trajectory exists its name appears on the list shown in the "Source" tab and if the user does not have selected a trajectory (or even have one created), it needs to go back and configure everything. The first thing to configure is the room or choose any that is already created, on the list shown in the "Room" tab. So far the application allows the creation of a rectangular room – by pressing the "NEW" button – where the height is declared as constant (3 meters) and the width and length (x and y), along with the room's environmental conditions such as temperature, humidity and pressure can be registered. With all the parameters registered, the user writes a name for the structure (or not, but a automated string with a date is generated: `rectangle%dd%mm%yy_%hh%mm_%ss.mat`) and saves it using the "SAVE" button. Depending on the previous configurations made by the user, a receiver can be attached to this structure. If the receiver is not configured yet (or even if the user intends to modify the current settings), on the "Receiver" tab there is a button and a knob present, where the button allows the user to input the receiver's location and the knob allows to adjust its rotation. By pressing the button it opens a figure with the room's shape and with the help of the mouse, the user places the receiver within its boundaries by pressing Left mouse button (LMB), which automatically exits the figure and updates the location. The other component, the knob, allows to rotate the receiver from $-180°$ to $180°$.

[1]There are some tweaks present whose branches would make the demonstration more complex to understand. It will be explained later what those tweaks are.

Now, considering that the user has already selected a room with a receiver properly configured, the user must draw the trajectory. In the "Source" tab the user writes a name for the structure (or not, but a automated string with a date is generated: `custom`*`%dd%mm%yy_%hh%mm_%ss.mat`*) and by pressing the "NEW" button a figure pops up, which contains the room's shape and the receiver's location and rotation. Inside this figure, the user chooses the velocity between two points (line segment) with the mouse wheel button – scroll up to increase velocity and scroll down to decrease it. To create a point the user simply presses LMB and to finish the drawing the user presses Right mouse button (RMB), which saves the drawing and exits the figure – completing the first main stage – *Trajectory definition*.

Having a trajectory defined the user may now interpolate it, to smooth the curves and to create a more realistic movement of the source. To do so it must choose a sound sample, which can be easily done by picking a file available in the drop-down menu and then by clicking on the "START" button next to the "Trajectory Interpolated" status text, where a progress bar appears displaying the remaining time for this operation. The time necessary to interpolate the trajectory is dependent on the sound sampling frequency and on the velocity chosen for each line segment. This completes the second main stage – *Interpolation of trajectory*.

The last (and most important) step is to generate a stereo sound file. This process is dependent on the interpolated trajectory and similarly to the interpolation operation, the operation time is dependent on the sampling frequency and velocities chosen for the line segments. After all of this, the user can view the demo of the AoA predictor by clicking on the *demo* button, as seen in figure 3.16 – completing the last main stage – *Reconstruction of spatial sound*.

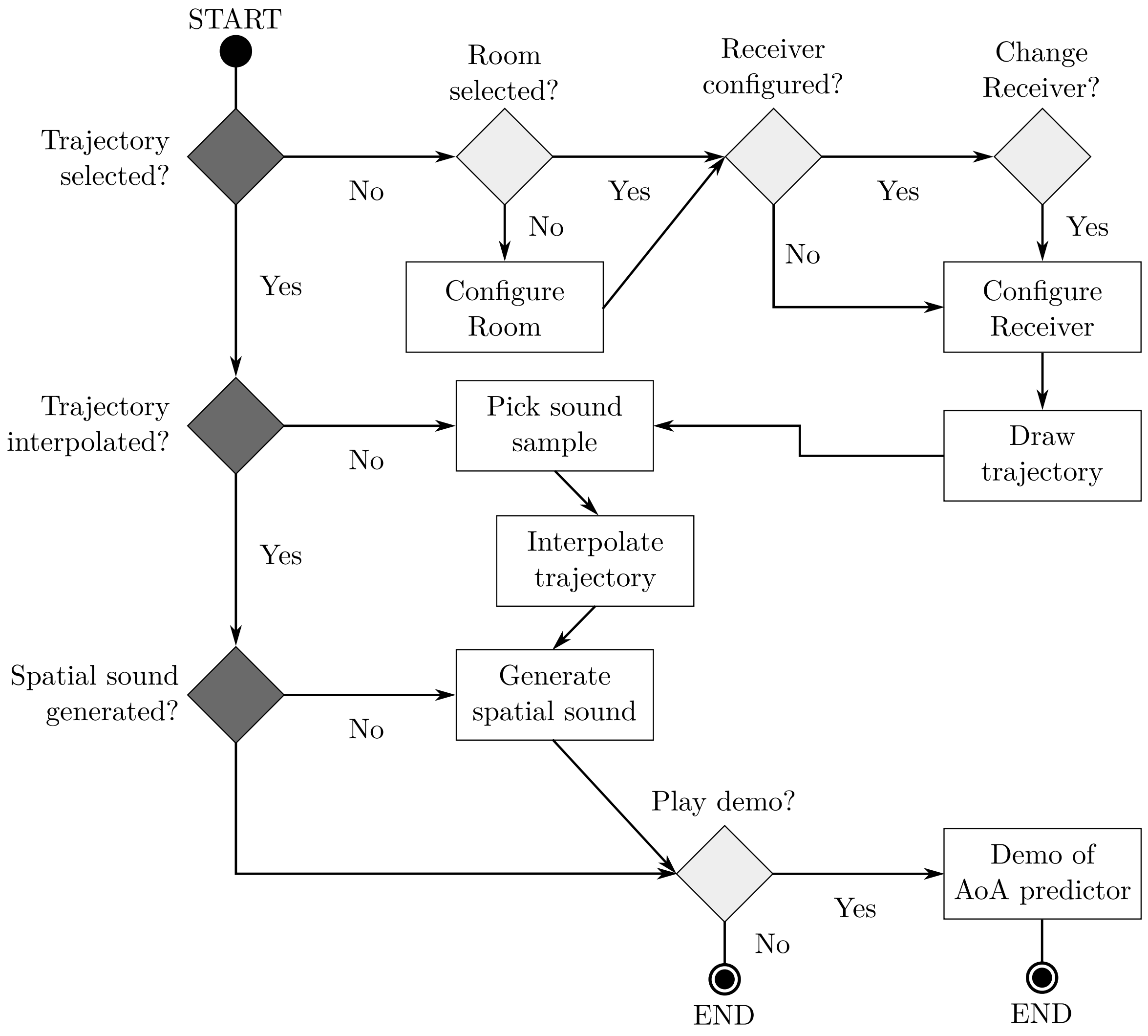


Figure 3.14: Flow chart of the *2D STS* application behaviour (where the darker decision markers represent the main stages).

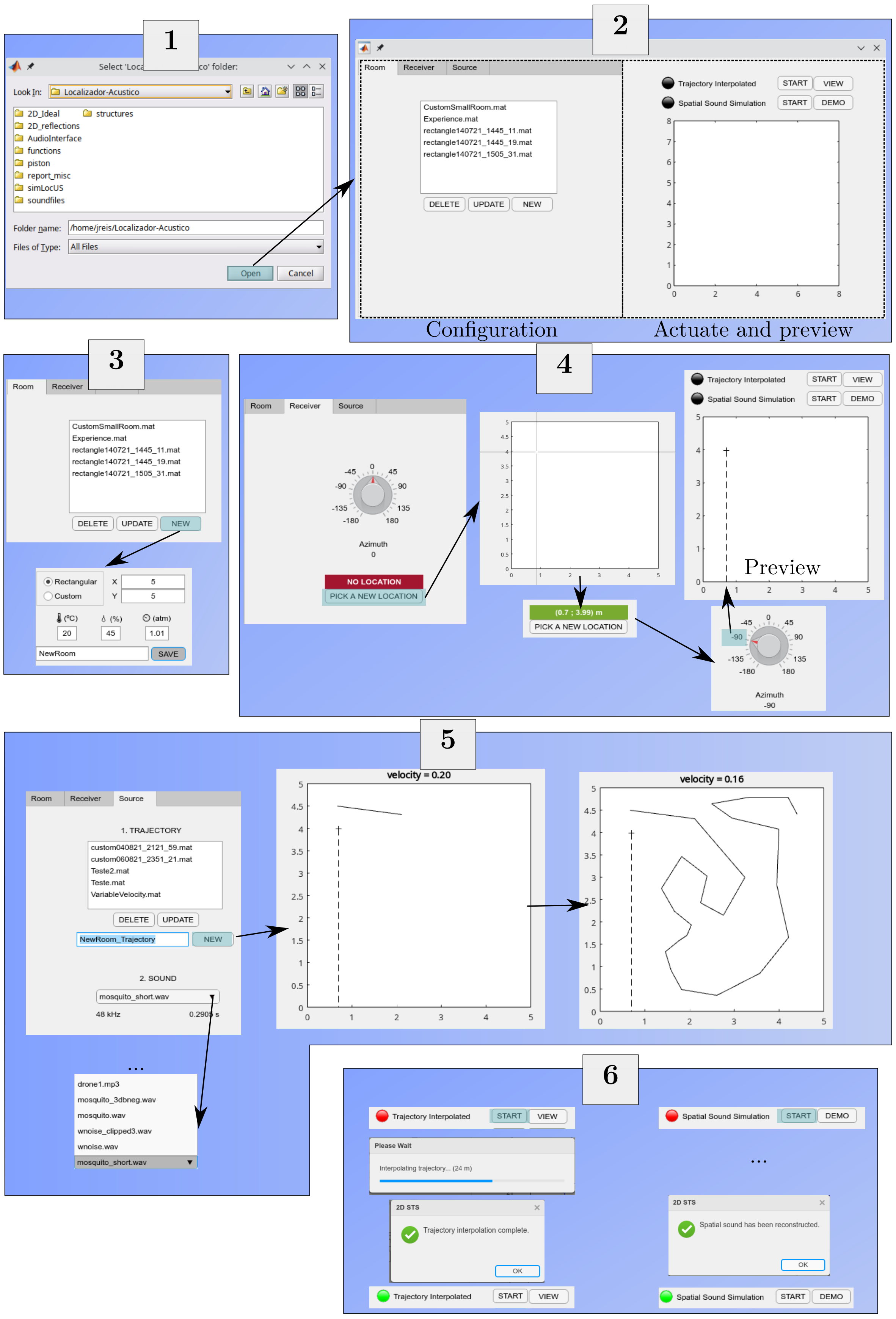


Figure 3.15: Sequence of images (screenshots) that aid the user navigate through the *2D STS* application.

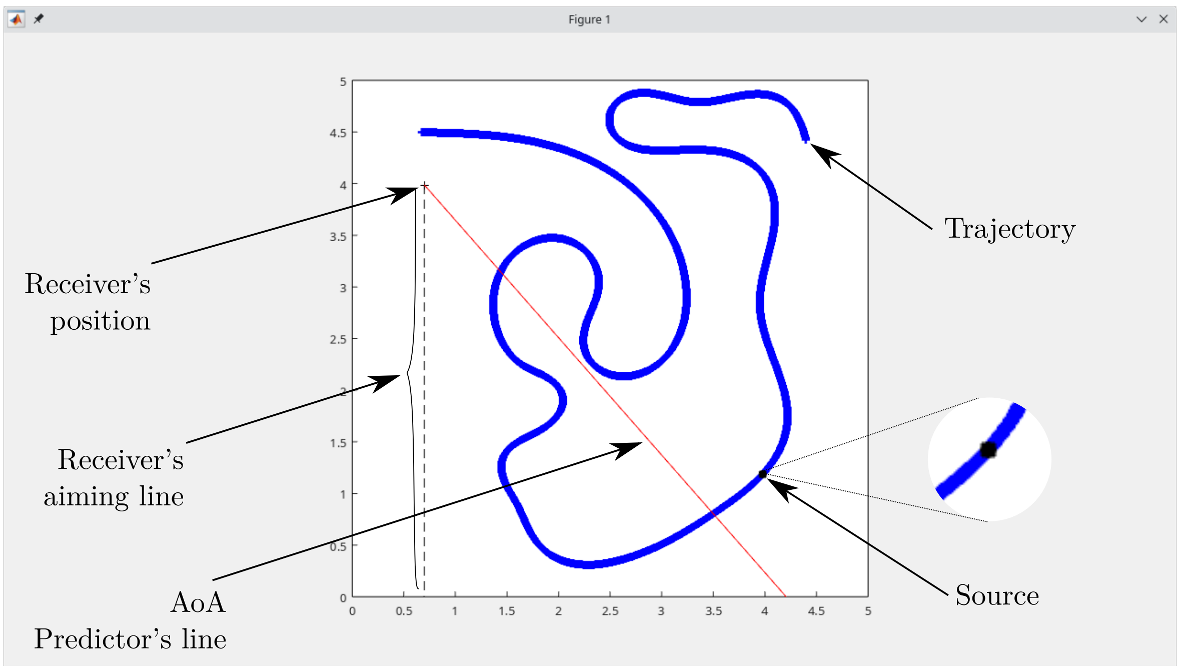


Figure 3.16: Demo playback in the *2D STS* application (with every marker identified).

CHAPTER 4

Experimental validation

This chapter describes the practical implementation of an acoustic source azimuth detector based on the far-field approximation discussed in the previous chapters, as well as the set of tests designed to assess its performance. These were carried out under controlled conditions, designed to recreate the simulation settings (anechoic, noise-free room) as closely as possible.

4.1 Testing setup

In order to avoid reflections and external noise interfering with the sound capture, the receiver was placed inside a sound-absorbing chamber, or "shed", built using blankets and duvets on a wooden structure (see figure 4.1). This does not ensure fully anechoic conditions, but proved enough for the intended purpose. creating a similar effect.

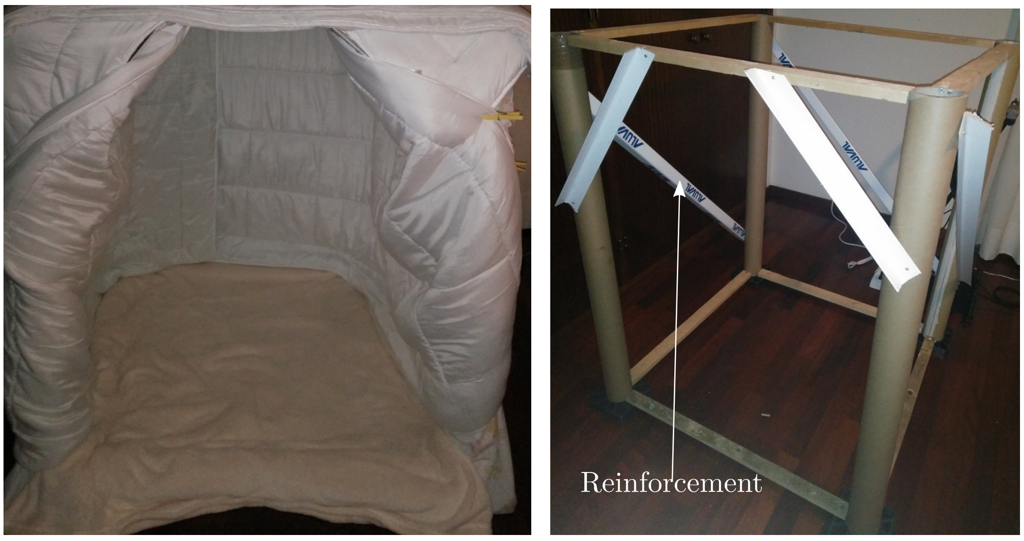


Figure 4.1: Shed (left) and its wooden structure (right).

The chamber had a base of 1×1 m^2 and a height of 1.5 m. For stability, each side was diagonally reinforced by a cardboard stripe, except for the entrance side, where the reinforcement was made at the top corners. This imposed a limitation on the height of the receiver because it could block sound waves from specific elevation angles – not a problem, since height was easily adjustable (see figure 4.2).

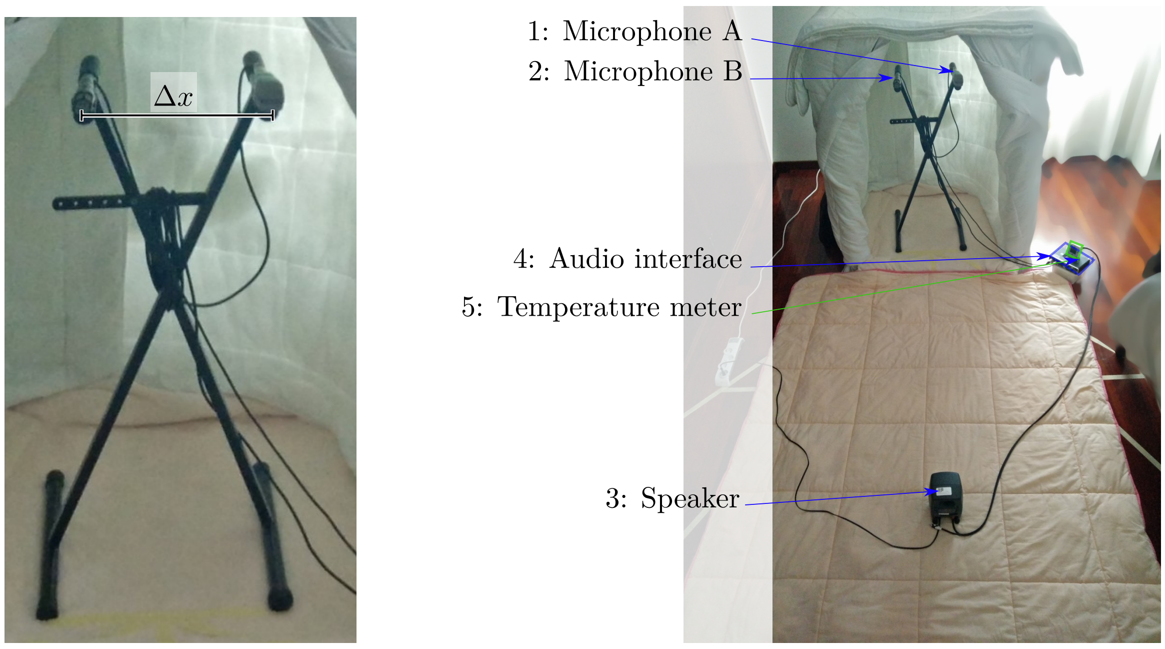


Figure 4.2: Receiver structure (left); Home-made quasi-anechoic test setting (right).

The receiver structure, placed inside the shed, consisted of two microphones mounted on an adjustable piano stand. In the chosen stand configuration, the separation between microphones, Δx, was 29.2 cm. As shown in the right-hand side image of figure 4.2, an additional duvet was employed to attenuate reflections off the floor, since the sound is originated by a speaker in front of the shed entrance. This duvet can be lifted to check the intended AoA or distance of the speaker in relation to the receiver (see figure 4.3).

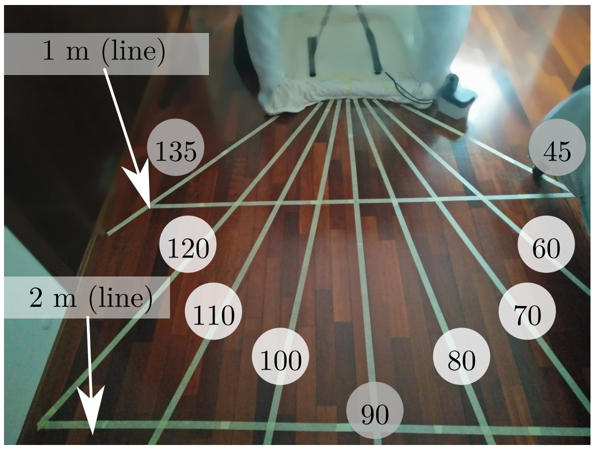


Figure 4.3: Duct-taped lines to guide the placement of the sound source.

Considering that the shed had only one entrance and the receiver must be inside it to avoid reflections, the effective unimpeded azimuth range was lower than 180°. To overcome this difficulty, to test the [0 45]° and [135 180]° ranges the receiver was rotated 45 degrees on each side, as shown in figure 4.4; the standard receiver position, with both microphones oriented along the 90° line, was used only for the [45 135]° range.

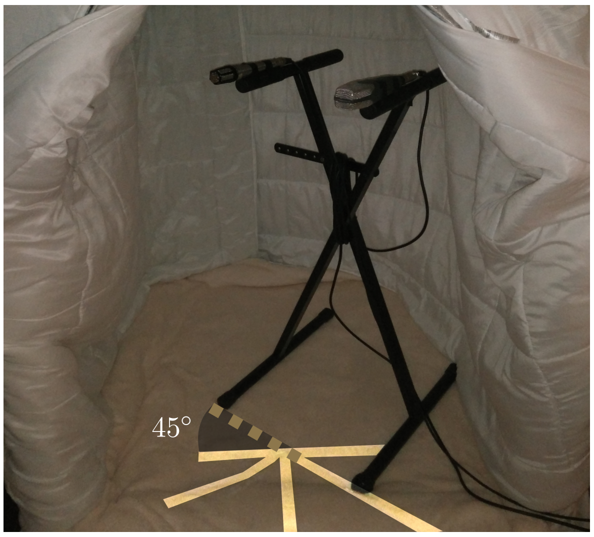


Figure 4.4: Receiver rotated by 45 degrees.

The audio equipment setup is described in figure 4.5 and table 4.1. The audio interface is capable of transmitting 24-bit audio with sampling frequency up to 192 kHz. The choice for these tests was 48 kHz.

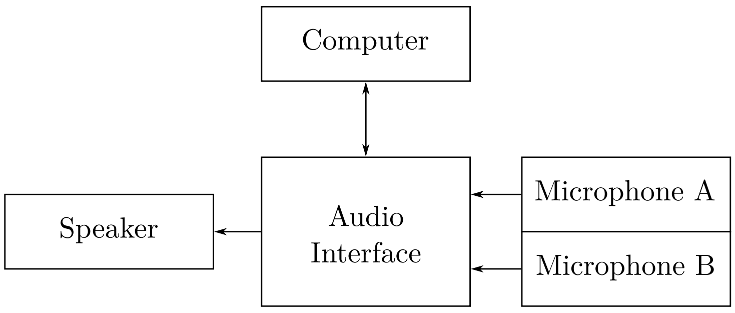


Figure 4.5: Block diagram of the home-made quasi-anechoic test setting.

ID	Block	Name
1	Microphone A	Sennheiser MD 419
2	Microphone B	Shure Prologue 10L
3	Speaker	Genelec 6010A
4	Audio interface	Roland QUAD-CAPTURE UA-55
5	Temperature meter	

Table 4.1: Home-made quasi-anechoic test setting component list.

The temperature meter is not represented on the block diagram, because it does not send data directly to the computer.

The tests required a computer with a USB interface, and an operating system compatible with MATLAB and the audio interface's driver. The website of the audio interface manufacturer)[43] does not provide drivers for Unix systems, but virtualisation software can be used as a workaround for this limitation[44]. All the sound files were captured with a custom MATLAB script in Microsoft Windows Enterprise LTSC 2019; the results were processed using another custom script in Arch Linux.

4.2 Testing procedure

In the tests, the position of the speaker was successively adjusted by the user along the 1 m line shown in figure 4.3 to cover 17 azimuth angles: 0°, 15°, 25°, 35°, 45°, 60°, 70°, 80°, 90°, 100°, 110°, 120°, 135°, 145°, 155°, 165° and 180°. Elevation was always kept constant, roughly at the same level as the microphones. A 10-second mosquito sound sample was recorded at each position. A small delay was included to allow for preparation, as shown in figure 4.6.

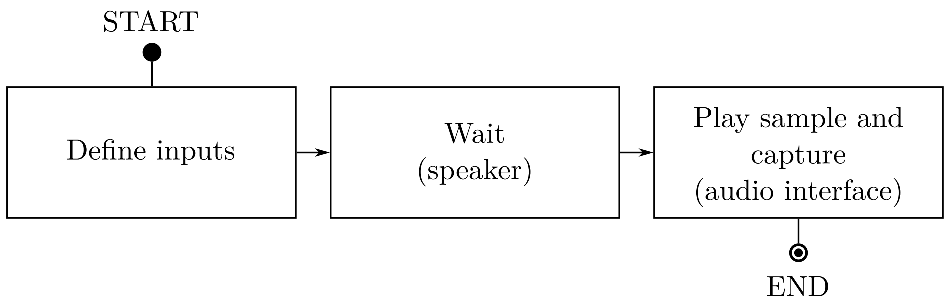


Figure 4.6: Flowchart for the audio capture script.

Before executing the script, the user defines the desired delay and the sample (mosquito sound) sent to the speaker (to be recorded by both microphones). In the waiting stage, the speaker emits a "beep" sound every second. Once the delay has elapsed, the system sends the sound sample to the speaker and starts recording at the same time. The captured sound is stored as a 16-bit `.wav` file.

The AoA can then be determined for each position and the associated error can be calculated considering the environment conditions (room temperature, microphone spacing and sampling frequency). The files were named `degX.wav`, where X is the azimuth angle of the recording position. Considering that there are M sound files, an array of size M (`AoA.ref(1:M)`) is created to capture AoA reference values (i.e. the azimuth of the speaker).

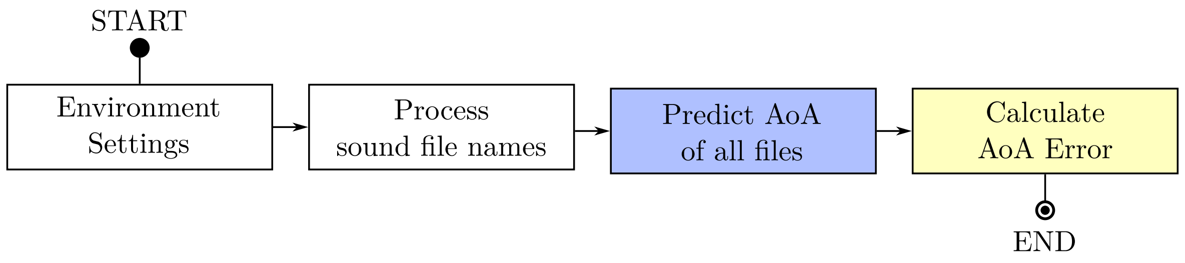


Figure 4.7: Block diagram (level 1) of the AoA prediction stage.

The captured sound file, which has the same duration as the sample sent to the speaker, is split into N temporal blocks. The AoA estimates obtained for the N blocks are stored in the `block(1:N)` array and then averaged into a single value. The `AoA.exp(1:M)` array contains the average value of the N blocks for each position. The overview of the system is shown in figure 4.7 and the detailed workflow is shown in figure 4.8.

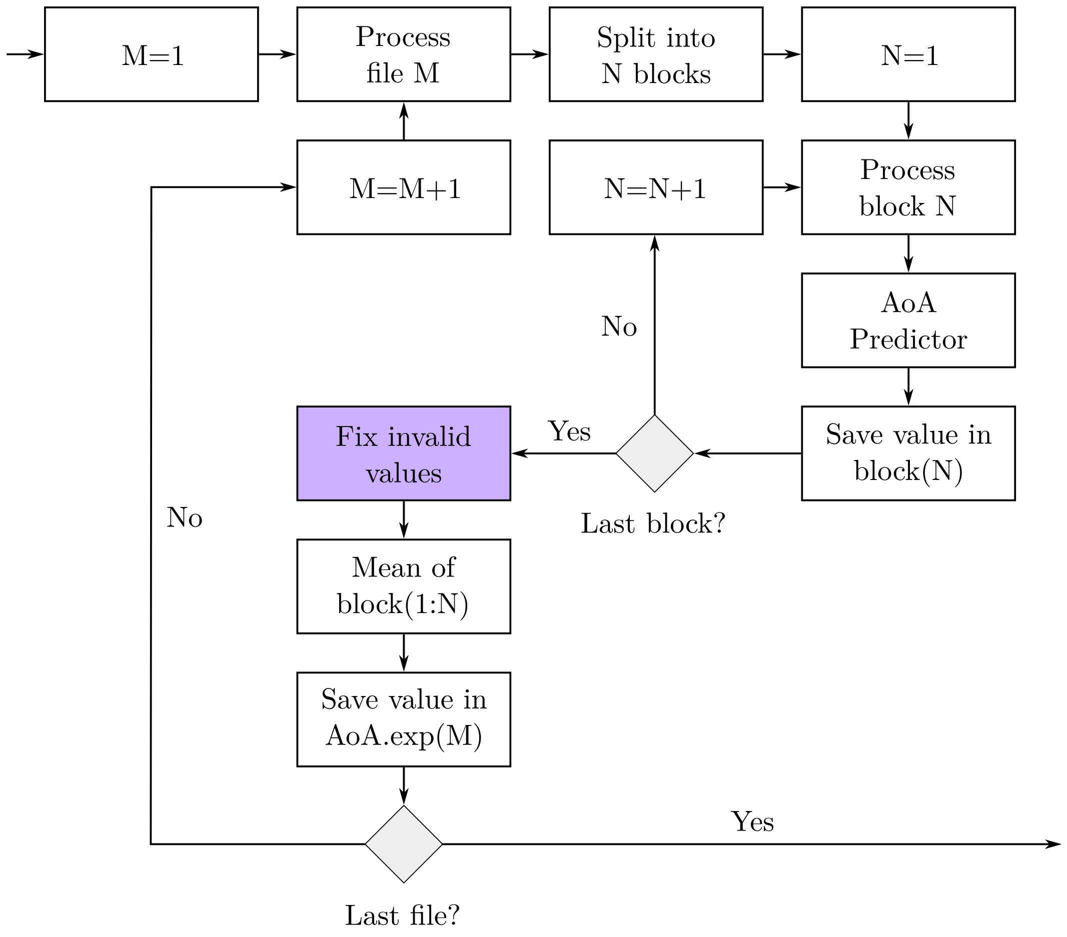


Figure 4.8: Detailed Flowchart of the block “Predict AoA of all files”.

A rule was established to fix outliers (i.e. invalid values caused by noise or sound wave reflections): these are replaced by the median value $\tilde{b}$ of `block(1:N)` (see figure 4.9). The threshold defined is of ten degrees, meaning that any sample of the `block(1:N)` array outside the interval ($[\tilde{b} - 10° \quad \tilde{b} + 10°]$) is replaced with $\tilde{b}$, the median value of block(1:N). Finally, the block “Calculate AoA Error” is very simple, as it consists of a simple operation between two arrays: `abs(AoA.exp(1:M)-AoA.ref(1:M))`.

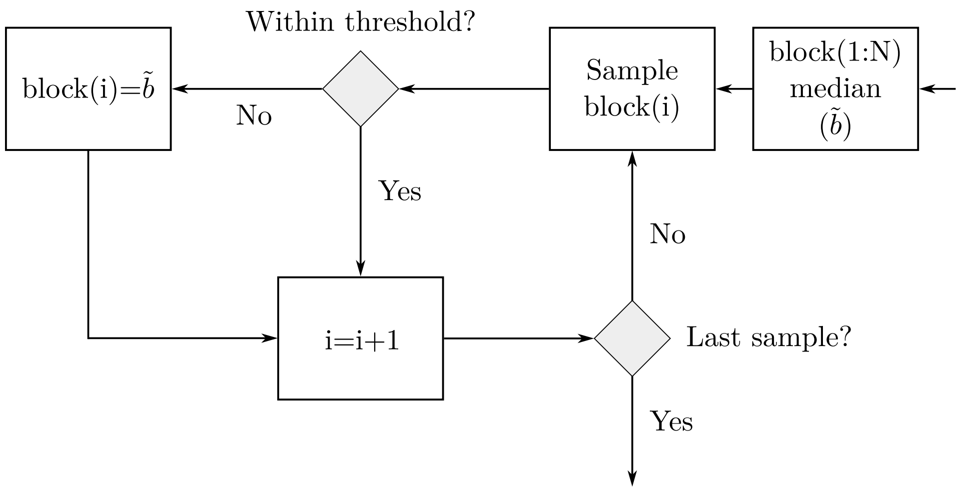


Figure 4.9: Detailed Flowchart of the block “Fix invalid values” of figure 4.8.

Simulations serve as another reference of values and can add more information about the predictor. They are implemented on a controlled environment where all parameters are equal or pretty close to the real ones in the home-made quasi-anechoic test setting. The environment variables that are introduced by the user are: the room dimensions, temperature and humidity; the receiver position, rotation and the distance between both microphones; the input audio sample (mono), the frequency range and the number of temporal blocks; and finally, trajectory Cartesian points that represent the position of the speaker in all the sound files originated from the experiment (the user must also define the trajectory velocity; this value can be determined in function of the total duration of the prediction). After defining all the variables, the script generates a stereo sound file of the speaker’s trajectory captured by the receiver’s microphones. Similar to the experiment, a custom script can be made to detect the AoA. However, while the previous script worked on multiple files, each one representing one AoA, in this simulation there is only one sound file but it contains all the AoA values in it (the sound file needs to be split into N blocks) – see figure 4.10.

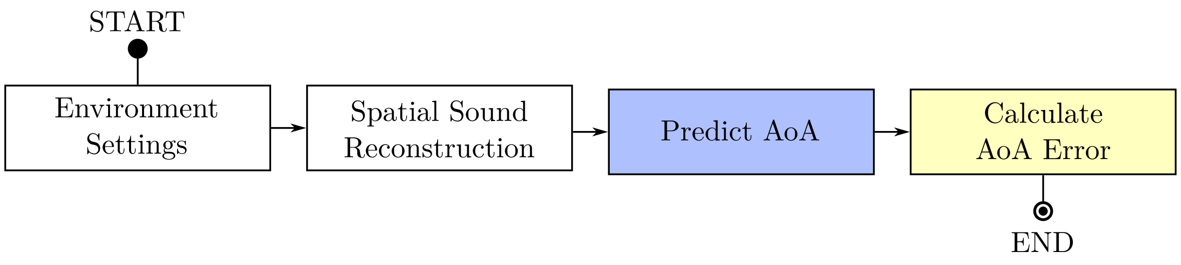


Figure 4.10: Block diagram (level 1) of the AoA prediction in a simulated environment.

The block that stands out is “Sound Spatial Reconstruction”, which is simply the generation of a stereo sound file. The block “Predict AoA” is similar to the previous one in figure 4.8 without the need to be executed on multiple files and the without “Fix invalid values” module, since this is a simulation.

The block diagram in figure 4.10 is incomplete due to the missing noise block, because the experimental results have some kind of noise associated with them (the shed is not perfect and does not block every interference or noise). To make the results stronger and increase their reliability, the simulation is run several times (it also reduces the probability of having pattern errors), averaging the results in the end.

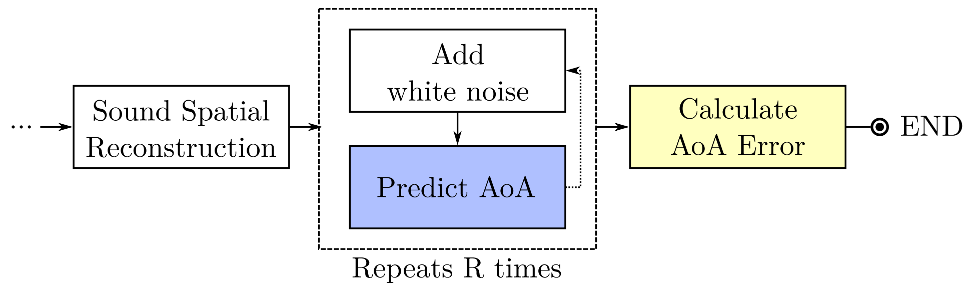


Figure 4.11: Block diagram presented in figure4.10 with the addition of noise (with repeatability).

This white noise block actuates on a temporal block, since each block represents an AoA value. It is different for each channel and for each repetition it goes through K number of Signal-to-noise ratio (SNR) measures – to also fully understand at which noise level the predictor becomes inaccurate.

Similar to the previous experiment, an array is created to store the values of the simulation, differing on the column size: `AoA.sim(1:N,1:K)`. This is due to the SNR values that are attributed to the white Gaussian noise being added for each channel (the user specifies an interval of SNR measures). K symbolizes the number of SNR measures and N the number of blocks that the sound file is split into. As each simulation is done, the K goes back to its initial value (to re-apply the SNR measures) and the resulting AoA is added to the previous value. In the end, when the simulation has been repeated R times, the array is averaged by the number of repetitions. Analysing figure 4.12 gives a better overview of what has been said.

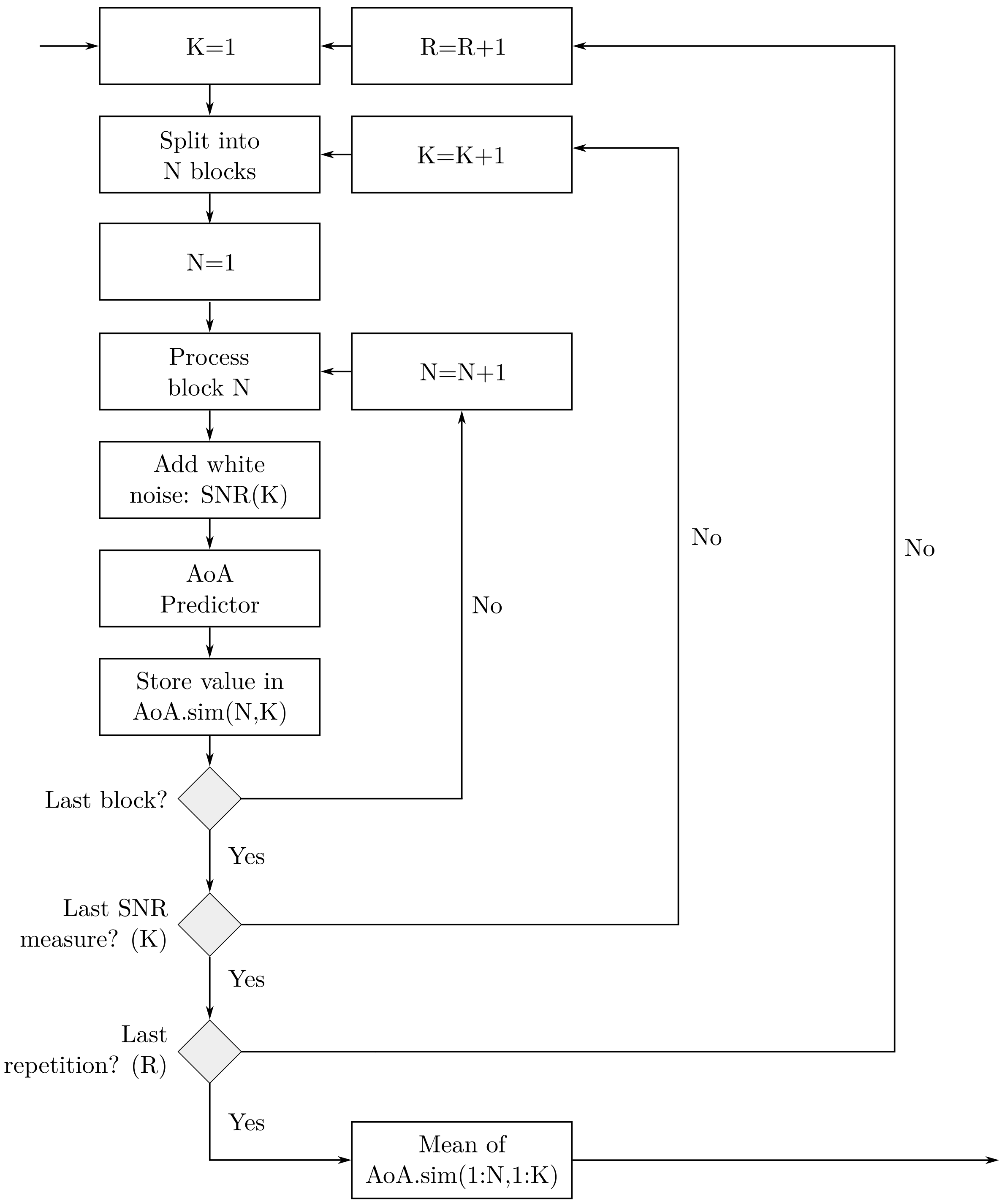


Figure 4.12: Detailed Flowchart of the simulation block combination “Add white noise” + “Predict AoA”.

The last experiment consisted of real-time azimuth estimation. To execute the script the user must configure the devices in MATLAB as well measure the experiment parameters needed for the predictor, to reduce the associated errors. The block diagram can be seen on figure 4.13.

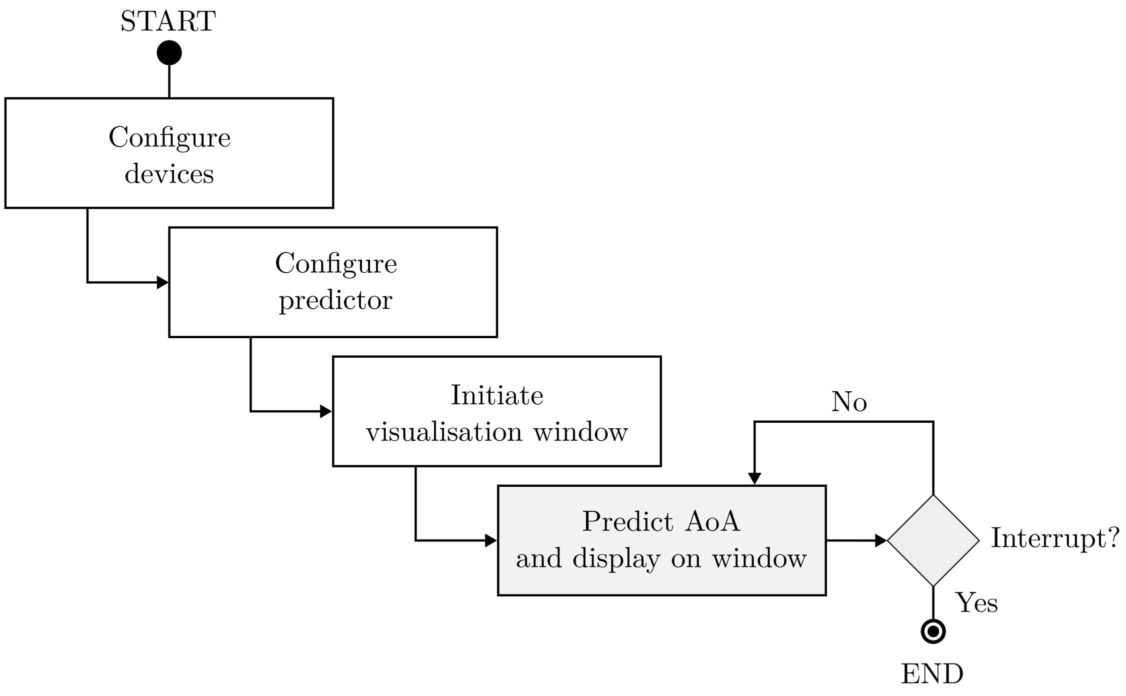

START
Configure devices
Configure predictor
Initiate visualisation window
No
Predict AoA and display on window
Interrupt?
Yes
END

Figure 4.13: Block diagram of the real-time script made in MATLAB.

The object *audioDeviceReader*[45] is responsible for the real-time processing of sound, whose function is to stream the data from the buffers made available by the audio interface's driver. Its buffer sample size can be defined in the setup process thus, attributing a time for each buffer. The speaker is configured using the *audioplayer* object and it plays a mono audio file chosen by the user. After measuring the sound velocity and the microphones spacing (Δx), the predictor configuration is updated with the same parameters.

When executing the script a figure window pops up, containing a plot of the receiver location and a pointer coming out of it. The infinite loop starts and the predictor is applied on each buffer data, being only interrupted if the user sends the respective signal, Control+C. The flowchart that describes the whole process is shown in figure 4.14.

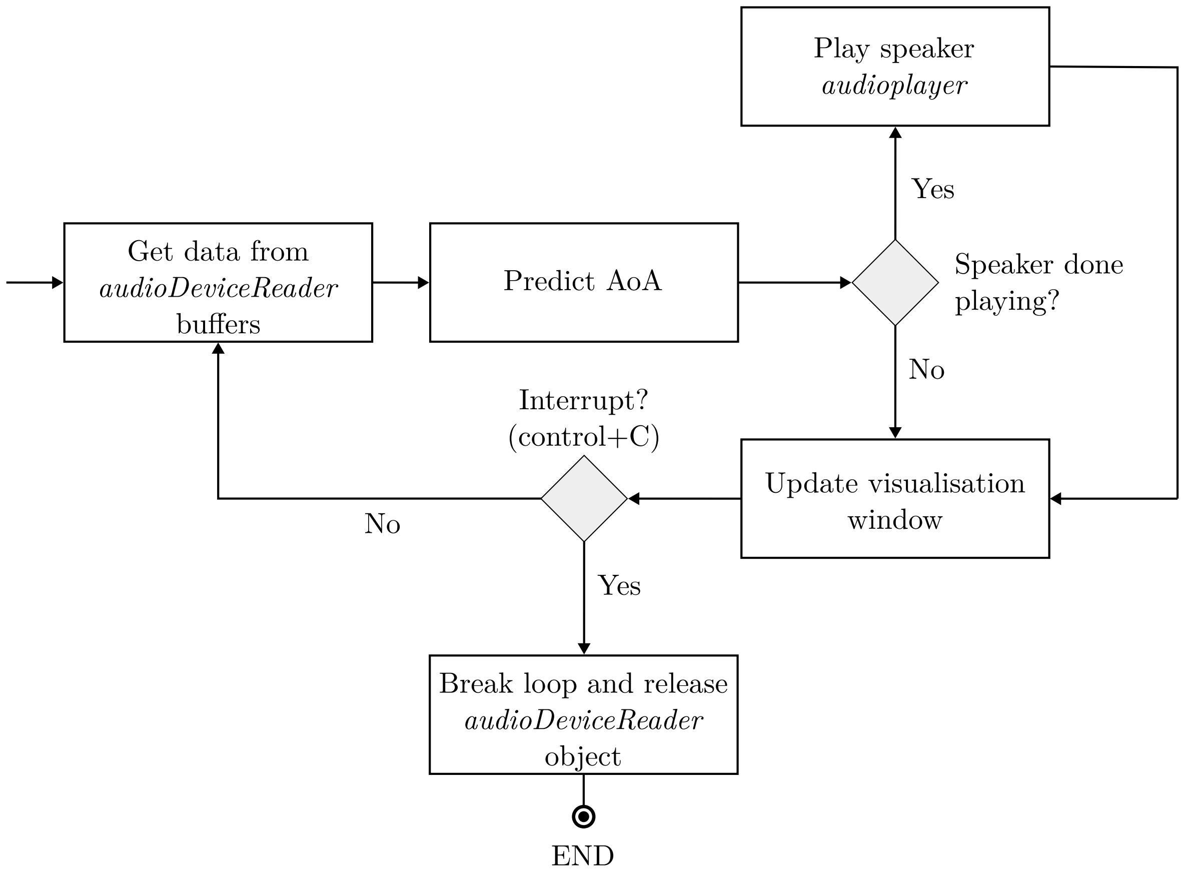


Figure 4.14: Flowchart of the real-time script made in MATLAB – precisely the block "Predict AoA and display on window".

4.3 Testing results

The simulation uses the same trajectory used in the home-studio experiment, a two-meter line one meter away from the receiver. Trying to emulate an anechoic room, a floor of three by three meters of size is used, with a null reflection coefficient. Similar to the experiment, the receiver is rotated two times (see figure 4.15). Needless to say, the shed does not need to be taken into account by the simulator. It is only present in the drawing to have an idea of its location.

The rest of the environment conditions are the same of the experiment made in the shed (home-made quasi-anechoic test setting). The only addition is the trajectory velocity, `SRC.traj.v`, which generates a stereo sound file with a duration of ten seconds ($\Delta t = (d/\Delta v) = (2/0.2) = 10s$) – see table 4.2.

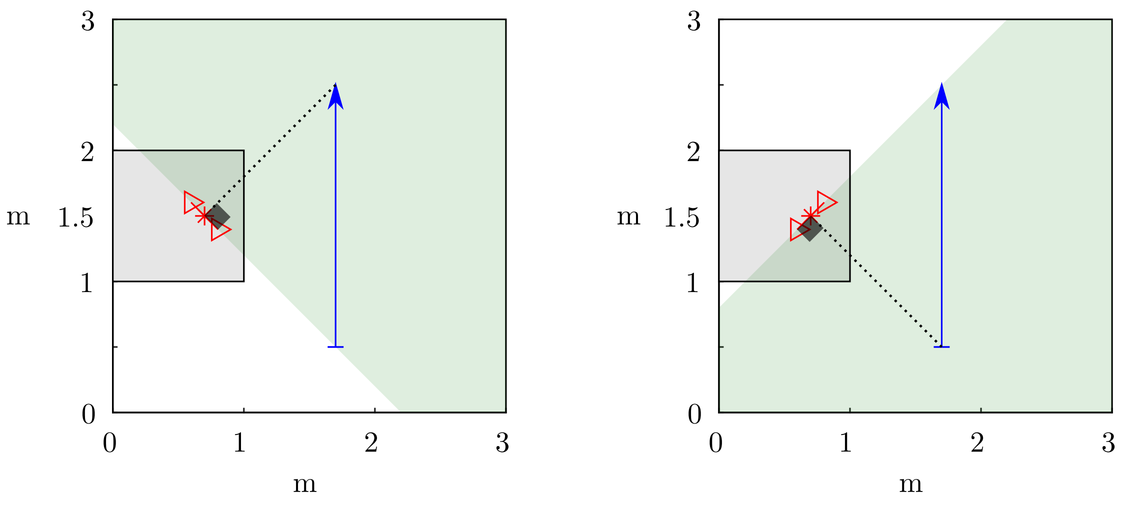


Figure 4.15: Simulation room mimicking to the studio's environment: receiver rotated by $-45°$ (left) and rotated by $45°$ (right).

Parameters	Value
`ROOM.AMB.T`	24 °C
`ROOM.AMB.H`	75 %
`ROOM.rec.dx`	292 mm
`SRC.traj.v`	0.2 m/s
`SRC.snd.fs`	48 kHz
Block length	250 ms

Table 4.2: Simulation parameters.

The procedure demonstrated in figure 4.10 with the previous input values originates the black line pattern, as show in the figure 4.16. There are two other lines, the green one which represents the predictor expected results where the estimation of the delay – TDoA – occurs without any errors (the best scenario possible for the system) and the blue line, which represents the predictor expected results with an error of 1% of a sample. It is important to compare these last two patterns, because in a real environment there will be an error associated to the sampling frequency used in the sound capture.

The predictor expected (theoretical) values start to form an "U" pattern when there is an error present on the TDoA estimation. This means that any source located in the extremes (0° and 90°) will inevitably have a higher prediction error that any other AoA value (remember, ±1% of a sample error[1] has been used and the extremes already have higher error values). In the AoA range of [45 135]° the simulation's error values are identical to the predictor expected values, oscillating around them. From outside of that, instead of converging to an error of 0°, it diverges to a 3° error.

[1]Negative on $-45°$ and positive on $45°$, to form a symmetric pattern.

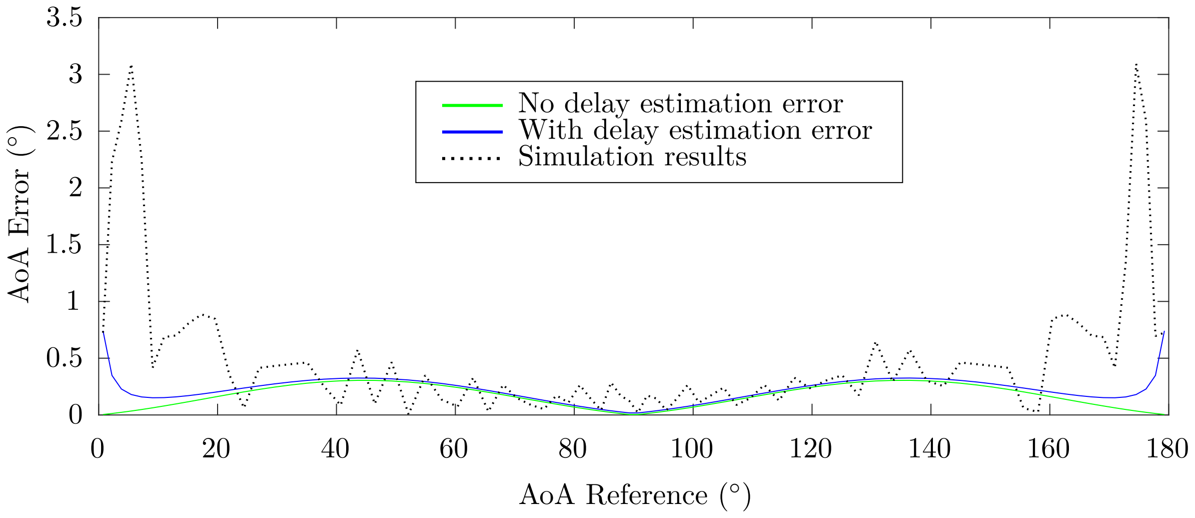


Figure 4.16: Simulation AoA error patterns.

The results produced until now came from a sound sample that did not contain any noise. Applying the procedure demonstrated in figure 4.11, with SNR measure values ranging from $[-19 \;\; 20]$ dB and with a total of 100000 repetitions, the simulation outputs an AoA error pattern of:

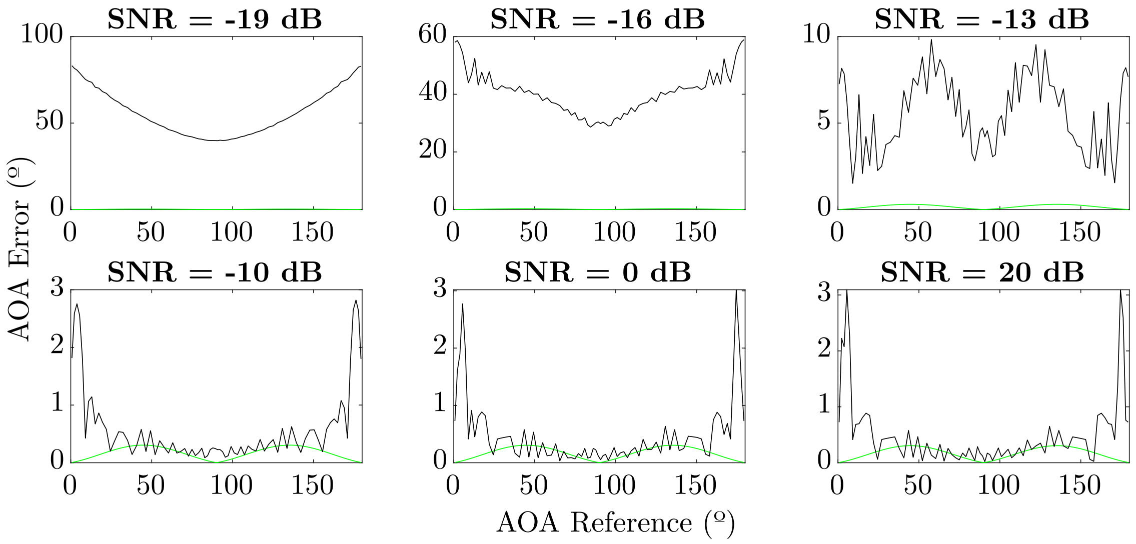


Figure 4.17: Simulation's AoA estimation error with white noise applied to the sound file.

The pattern is very similar in the SNR range of $[-10 \;\; 20]$ dB, changing drastically bellow 10 dB, which is the minimal SNR where the predictor outputs good results. This is why from here the values decreases in 3 dB steps – to demonstrate how quickly the system prediction error increases.

Now that all the simulated results have been shown, it remains to discuss about the experiment results. By applying the procedure discussed in figure 4.7, all the sound files give the following AoA error pattern:

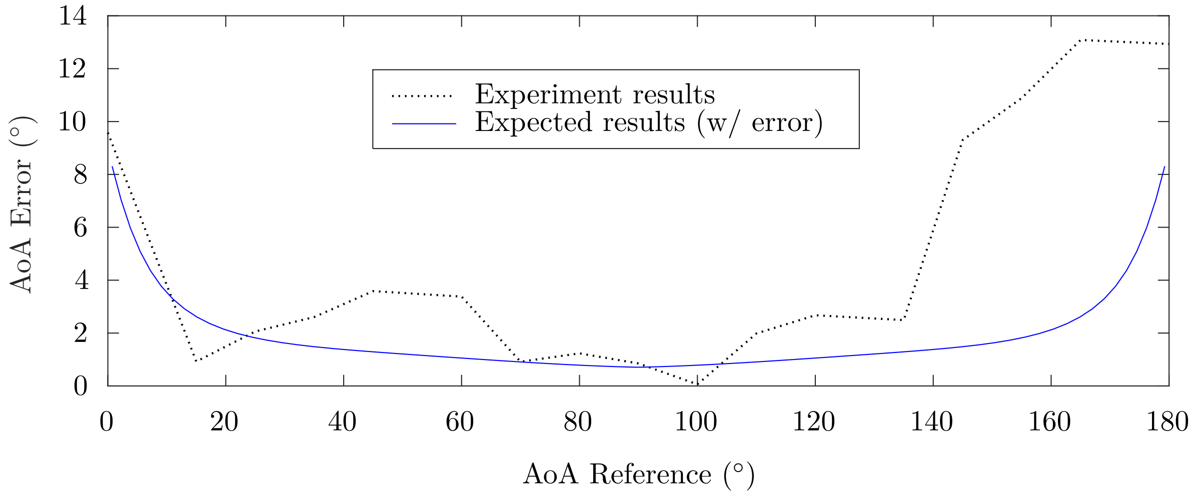


Figure 4.18: Experimental AoA error patterns.

The first thing to notice is that the pattern is asymmetrical, with an absolute global maximum error of ≈ 14° close to 180° reference and with a absolute local maximum error of ≈ 10° close to 0° reference. The second difference is that, unlike the expected curve where it has its global minimum value at 90°, it has its minimal value in 100°. Finally, all the errors are much higher than the ones presented in figure 4.16. Noticing its pattern, its also clear that there is noticeable error associated in the TDoA estimation.

The blue line pattern is the predictor expected AoA values with an error of half a sample, 50 times the error presented in the simulation figure. While not being the exact TDoA estimation error, it gives an idea of the scale of the error.

CHAPTER 5

Conclusion

5.1 Discussion

In the beginning, mosquito species were studied along with the human auditory system to inspire the possible prototype of the receiver. Nonetheless, by analysing simpler structures such as two microphones that are spaced of each other can provide useful data. The studies presented in the beginning of this thesis are valuable and should not be discarded, as they orient the user on how they should design their prototype – from displaying the most common frequencies in mosquitoes to recommending algorithms that accurately predict the azimuth and elevation of acoustic sources.

The first experiment consisted of estimating the AoA in an quasi-anechoic chamber. As stated, it consists of four main components: the computer, the audio interface, the receiver (set of microphones) and the speaker. The audio interface allows the computer to send data to the speaker and to gather the data from the microphones – all possible in real-time. In the end, MATLAB does all the signal processing and gives the estimates for the AoA values.

To establish a comparison with the experimental results, simulations were made using the same environment settings as those in the recordings – mainly because simulated values are close to real values. The theoretical values from equation (2.7) represent the perfect values – even when analysed with the associated errors from measurements (TDoA, distance and temperature). Simulated results serve as a reference for practice as they were modelled to imitate the processes of the real world.

In an ideal environment the simulation outputs an AoA error pattern that resembles an "U" symmetric shape around 90°. It oscillates $\approx \pm 0.25°$ around the theoretical curve at the AoA range of $[45\ \ 135]°$ and diverges up to an error of $\approx \pm 3°$ outside that range. However, when the sound samples used in the signal processing have an SNR of -10 dB or less, the AoA estimation errors increase greatly.

Furthermore, the experimental AoA results also display an "U" pattern but it is asymmetric with the minimum error on 100° instead of 90°. Since the algorithm is based mainly on cross-correlation of both channels, the microphones intensity (gain) is not the culprit. The error can be attributed due to the fact that: the microphone manufacturers are different and so the characteristics affect each one's output; radiation pattern from both microphones do not match and finally, due to the gain not being omnidirectional, which will distort the signal at specific angles. Not only that, but measurement errors in the microphone spacing (Δx) (as seen in 2.3), TDoA estimation (see 2.4) and sound velocity (temperature dependent) can also impact greatly the system's accuracy.

Finally, the last experiment consisted of the detection in real-time of a source in a 2D environment and it proves that the predictor is relatively fast (when compared to the speaker movements) and can make good estimates, as long as the noise conditions were favourable (see appendix C.2).

5.2 Future work

The predictor developed for this experiment is its early state, where it gives a reasonable estimate of the azimuth of a sound source in an ideal environment. However, it is not capable of estimating the elevation yet – something that can be added to the predictor with ease (lightweight and modular predictor).

Another feature that is not enabled yet is the frequency filtering of the sound captured by the receiver. Throughout this experiment almost all predictions did not use frequency filtering due to using other sound samples that are not wing-beats produced by mosquitoes. Still, this can be easily done using an algorithm based on Fast Fourier transform (FFT) that considers the range of the mosquitoes frequencies, as indicated in table 1.5. If the user desires to capture its harmonics, it can also add more band-pass filters around them to add even more precision.

Furthermore, some improvements to the existing system can be made. The structure is one of them, where the spatial diversity can be increased by adding more microphones, increasing the distance that separates each one of them and by adding obstacles between them. It was proven the AoA estimation error is decreased by increasing their spacing (see section 2.3) and that while $N+1$ microphones are needed for the detection in the Nth dimension, for 2D it can be used only two microphones if they have an obstacle around them (see section 1.4).

When taking into consideration the accuracy of the predictor, it becomes necessary to take measures to prevent any error propagation than can affect the results. The most important ones are the distance measurements between both microphones (Δx), the TDoA estimation and their radiation pattern matching – to avoid the distortion the signals. A solution would be the active measuring of Δx with a precise system and the increase of sampling frequency to avoid fake TDoA spikes in cross-correlation. For the microphones, the easiest solution would be to use omnidirectional microphones, but the problem is that they're more expensive. The alternative would be to balance each microphone radiation pattern to reduce the signal distortion.

In the interactive simulator and predictor application – *2D STS* – some features are not present and it is only executable inside the MATLAB software bundle. One feature that could make the application more flexible is the customization of the receiver being used i.e. changing the number of microphones, adding geometric shapes that serve as obstacles and changing the gain function for each microphone – something that is already possible on the *Ultrasonic Room Simulator*. Now for the portability of the application, it could be developed in cross-platform language (such as *python* or even *C++*) and embedded into an cross-platform interface (such as *Qt*[46]).

Lastly, the possibility of porting the implemented software onto other projects and researches makes an important point for future work. Since the user has access to all the source code of what has been developed (see appendix C.1), it can port whatever feature he may finds suitable. One useful port would be the custom trajectory drawing module from the *2D STS* app.

This work is being continued by my colleague from the same department – *DETI*, Universidade de Aveiro. What has been done allowed for the improvement of the predictor and development of the remaining modules.

ANNEX A

Mathematical expressions

A.1 Hyperbola

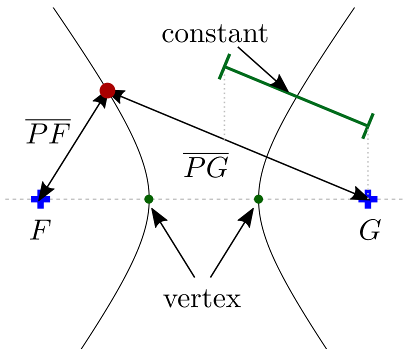


Figure A.1: Hyperbola demonstration.

This figure contains an hyperbola and it can be described by having a branch surrounding each focal point, F and G, where each branch has its sharpest turn on its vertex. The point P describes an arbitrary position in each branch.

By definition, an hyperbola is defined by the absolute value of the distance difference from the same point P (from the same branch) to both focal points:

$$\left|\overline{PF} - \overline{PG}\right| = constant. \tag{A.1}$$

This distance difference remains the same value as the point moves along the branch.

If the point P is placed on the x-axis (see figure A.2), it can be observed that this distance difference is equivalent to the distance between both vertices.

By applying coordinates to each point and moving the point $P = (x, y)$ to the right branch (positive coordinates and easier to read), the hyperbola is the following:

The distance difference measures $2a$ and the distance between both focal points is of $2c$. By applying the Euclidean distance formula[47] to both distances:

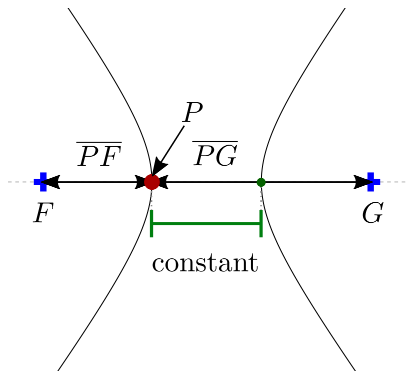


Figure A.2: Hyperbola with point P on top of a vertex.

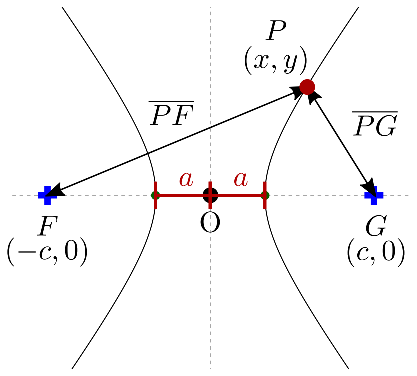


Figure A.3: Hyperbola with coordinates.

$$\overline{PF} = \sqrt{(x+c)^2 + y^2}, \tag{A.2}$$

$$\overline{PG} = \sqrt{(x-c)^2 + y^2}, \tag{A.3}$$

and completing equation (A.1) with both distances and constant values:

$$\left| \sqrt{(x+c)^2 + y^2} - \sqrt{(x-c)^2 + y^2} \right| = 2a, \tag{A.4}$$

which is equivalent to

$$\sqrt{(x+c)^2 + y^2} - \sqrt{(x-c)^2 + y^2} = 2a, \tag{A.5}$$

since $\overline{PF}$ will always be greater than $\overline{PG}$ when the point P moves along the right branch.

ANNEX B

Physics demonstrations

B.1 Inverse-square law

B.1.1 Definition

This law implies that a physical quantity is inversely proportional to the square of the distance from the source of that physical quantity:

$$\text{intensity} \propto \frac{1}{\text{distance}^2}. \tag{B.1}$$

B.1.2 Explanation

Inverse-square law is typically applied when a physical quantity radiates evenly from a source that is located in a 3D space. As expected, this forms a sphere around the source, where its surface area is defined by:

$$4\pi r^2. \tag{B.2}$$

As emitted radiation travels away from the source, it spreads out in proportion to the square distance from the source, since its surface are is defined by equation (B.2). This results in radiation intensity per unit area being inversely proportional to the square of the distance to the source.

One good example can be observed in figure B.1, where s represents a light source and r the sphere radius (distance from the source). The total number of flux lines (red lines) is constant for every distance and the surface area at each radius is increased with the square of its value (one square, four squares, nine squares – each one has the same area). It can also be observed that the flux lines on the darker square decrease with the inverse square of the value of the sphere radius.

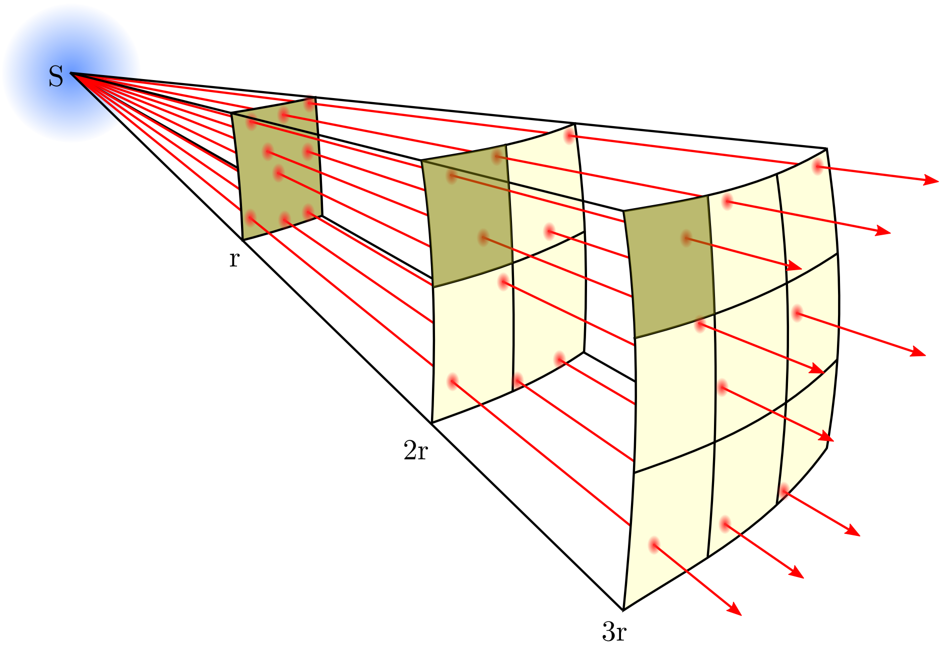


Figure B.1: Inverse-square law applied on a light source (CC license by 4.0)

APPENDIX C

Source files and media

C.1 Code repository

All the source files made on MATLAB are located on the GitHub repository: `https://github.com/joserei5/Localizador-Acustico/`.

The directories are explained in table C.1.

	Folder	Description
📁	2D_Ideal	Azimuth detection in ideal conditions
📁	2D_Reflections	Azimuth detection with reflections
📁	AudioInterface	Real-time and offline detection using the studio setup
📁	functions	Self-titled: scripts that help user productivity; 2D_STS program location (`app/simsound.mlapp`)
📁	piston	Interactive radiation plot of a circular plane piston
📁	report_misc	Miscellaneous files related to the thesis (figures, scripts, etc.)
📁	simLocUS	Ultrasonic Location System (simulator) adapted for this experiment
📁	soundfiles	Mono sound samples (generic), recordings from the studio (capture), mosquito species sound samples and sounds generated by scripts
📁	structures	Structures necessary for the 2D_STS interactive program

Table C.1: GitHub repository – preview of the folders.

C.2 Video demonstrations

All the video files related to the predictor working in real-time can be found at the Google Drive link: `https://drive.google.com/drive/folders/1n3OlIe_u8yPeCGVW0FHMW2-Jnh3XIu9t`

C.3 Contacts

If there are difficulties while accessing the URLs presented in the appendices, or even finding the necessary files, please do contact me at jreis.ua@gmail.com.

References

[1] G. F. Killeen, J. Kihonda, E. Lyimo, *et al.*, “Quantifying behavioural interactions between humans and mosquitoes: Evaluating the protective efficacy of insecticidal nets against malaria transmission in rural Tanzania,” in BioMed Central, Nov. 2006, vol. 6, pp. 4–6. DOI: 10.1186/1471-2334-6-161.

[2] K. Dye-Braumuller, C. Fredregill, and M. Debboun, “Mosquito Control,” in *Mosquitoes, Communities, and Public Health in Texas*, Cambridge, MA, USA: Academic Press, Jan. 2020, pp. 249–278, ISBN: 978-0-12-814545-6. DOI: 10.1016/B978-0-12-814545-6.00008-0.

[3] *Sistema Anti-Melgas I: Identificação de localização de fontes sonoras*, University of Aveiro, 2020-2021.

[4] “Using Insect Sounds to Estimate and Monitor Their Populations,” *The Florida Entomologist*, vol. 71, no. 4, pp. 416–425, Dec. 1988, ISSN: 00154040. DOI: 10.2307/3495001.

[5] D. Saunders, X. Vafopoulou, C. Steel, and R. Lewis, *Insect Clocks, Third Edition*, 3rd ed. 2002, pp. 1–100, ISBN: 0080534716.

[6] T. D. Schowalter, *Insect Ecology. An Ecosystem Approach*, 4th ed. Academic Press, 2016, pp. 30–138, ISBN: 0128030372.

[7] M. J. and S. M. Catherine A. Tauber, *Seasonal Adaptations of Insects*, 1st ed. Oxford University Press, 1985, ISBN: 9780195036350.

[8] *Family Culicidae Meigen, 1818 | Mosquito Taxonomic Inventory*, [Online; accessed 24. Jun. 2022], Jun. 2022. [Online]. Available: https://mosquito-taxonomic-inventory.myspecies.info/family-culicidae-meigen-1818.

[9] D. C. P. Câmara, C. d. S. Pinel, G. P. Rocha, C. T. Codeço, and N. A. Honório, “Diversity of mosquito (Diptera: Culicidae) vectors in a heterogeneous landscape endemic for arboviruses,” *Acta Trop.*, vol. 212, p. 105 715, Dec. 2020, ISSN: 0001-706X. DOI: 10.1016/j.actatropica.2020.105715.

[10] A. R. Medeiros-Sousa, W. Ceretti-Júnior, G. C. de Carvalho, *et al.*, “Diversity and abundance of mosquitoes (Diptera:Culicidae) in an urban park: Larval habitats and temporal variation,” *Acta Trop.*, vol. 150, pp. 200–209, Oct. 2015, ISSN: 0001-706X. DOI: 10.1016/j.actatropica.2015.08.002.

[11] S. Boukraa, M. A. de La Grandiere, T. Bawin, *et al.*, “Diversity and ecology survey of mosquitoes potential vectors in Belgian equestrian farms: A threat prevention of mosquito-borne equine arboviruses,” *Prev. Vet. Med.*, vol. 124, pp. 58–68, Feb. 2016, ISSN: 0167-5877. DOI: 10.1016/j.prevetmed.2015.12.013.

[12] Petrić, Dušan, Bellini, *et al.*, “Monitoring population and environmental parameters of invasive mosquito species in Europe,” *Parasites Vectors*, vol. 7, no. 1, pp. 1–14, Dec. 2014, ISSN: 1756-3305. DOI: 10.1186/1756-3305-7-187.

[13] P. F. Mattingly, “The urban mosquito hazard today,” *Bull. World Health Organ.*, vol. 29, no. Suppl, pp. 135–139, 1963, ISSN: 0042-9686. [Online]. Available: https://apps.who.int/iris/handle/10665/266878.

[14] M. C. Kahn, W. Celestin, and W. Offenhauser, “Recording of Sounds Produced by Certain Disease-Carrying Mosquitoes,” *Science*, vol. 101, no. 2622, pp. 335–336, Mar. 1945, ISSN: 0036-8075. DOI: 10.1126/science.101.2622.335.

[15] A. Moore, J. R. Miller, B. E. Tabashnik, and S. H. Gage, “Automated Identification of Flying Insects by Analysis of Wingbeat Frequencies,” *J. Econ. Entomol.*, vol. 79, no. 6, pp. 1703–1706, Dec. 1986, ISSN: 0022-0493. DOI: 10.1093/jee/79.6.1703.

[16] D. T. Huck, M. S. Klein, and M. E. Meuti, “Determining the effects of nutrition on the reproductive physiology of male mosquitoes,” *J. Insect Physiol.*, vol. 129, p. 104 191, Feb. 2021, ISSN: 0022-1910. DOI: 10.1016/j.jinsphys.2021.104191.

[17] A. N. Clements, *The Physiology of Mosquitoes.* Oxford, England, UK: Pergamon, Jan. 1963, pp. 210–298, ISBN: 978-1-48322276-9.

[18] H. Mukundarajan, F. J. H. Hol, E. A. Castillo, C. Newby, and M. Prakash, “Using mobile phones as acoustic sensors for high-throughput mosquito surveillance,” *eLife*, Oct. 2017. DOI: 10.7554/eLife.27854.

[19] A. N. Clements, *Biology of Mosquitoes. Sensory Reception and Behaviour*, 1st ed. CABI, Jun. 1999, vol. 2, pp. 287–332, ISBN: 9780851993133.

[20] W. G. Brogdon, “Measurement of Flight Tone Differences Between Female Aedes aegypti and A. albopictus (Diptera: Culicidae),” *J. Med. Entomol.*, vol. 31, no. 5, pp. 700–703, Sep. 1994, ISSN: 0022-2585. DOI: 10.1093/jmedent/31.5.700.

[21] W. G. Brogdon, “Measurement of Flight Tone Differentiates Among Members of the Anopheles gambiae Species Complex (Diptera: Culicidae),” *J. Med. Entomol.*, vol. 35, no. 5, pp. 681–684, Sep. 1998, ISSN: 0022-2585. DOI: 10.1093/jmedent/35.5.681.

[22] L. J. Cator, B. J. Arthur, A. Ponlawat, and L. C. Harrington, “Behavioral Observations and Sound Recordings of Free-Flight Mating Swarms of Ae. aegypti (Diptera: Culicidae) in Thailand,” *J. Med. Entomol.*, vol. 48, no. 4, pp. 941–946, Jul. 2011, ISSN: 0022-2585. DOI: 10.1603/ME11019.

[23] J. J. Feher, *Quantitative Human Physiology. An Introduction.* Academic Press, 2012, pp. 371–374, ISBN: 9780123821638; 0123821630.

[24] Y. Park, A. Choi, and K. Kim, “Monaural Sound Localization Based on Reflective Structure and Homomorphic Deconvolution,” *Sensors (Basel, Switzerland)*, vol. 17, no. 10, Oct. 2017. DOI: 10.3390/s17102189.

[25] E. A. Macpherson and A. T. Sabin, “Vertical-plane sound localization with distorted spectral cues,” *Hear. Res.*, vol. 306, pp. 76–92, Dec. 2013, ISSN: 0378-5955. DOI: 10.1016/j.heares.2013.09.007.

[26] R. A. Butler and R. A. Humanski, “Localization of sound in the vertical plane with and without high-frequency spectral cues,” *Percept. Psychophys.*, vol. 51, no. 2, pp. 182–186, Mar. 1992, ISSN: 1532-5962. DOI: 10.3758/BF03212242.

[27] P. Schleich, P. Nopp, and P. D’Haese, “Head Shadow, Squelch, and Summation Effects in Bilateral Users of the MED-EL COMBI 40/40+ Cochlear Implant,” *Ear Hear.*, vol. 25, no. 3, pp. 197–204, Jun. 2004, ISSN: 1538-4667. DOI: 10.1097/01.AUD.0000130792.43315.97.

[28] M. C. Flynn, C. A. Sammeth, A. Sadeghi, G. Cire, and G. Halvarsson, “Baha® for single-sided sensorineural deafness: Review and recent technology innovations,” *Seminars in Hearing*, vol. 31, no. 4, pp. 326–349, Nov. 2010, ISSN: 1098-8955. DOI: 10.1055/s-0030-1268033.

[29] M. M. Van Wanrooij and A. J. Van Opstal, “Contribution of Head Shadow and Pinna Cues to Chronic Monaural Sound Localization,” *J. Neurosci.*, vol. 24, no. 17, pp. 4163–4171, Apr. 2004. DOI: 10.1523/JNEUROSCI.0048-04.2004.

[30] D. W. Grantham, B. C. J. Moore, and M. P. Friedman, *Spatial Hearing and Related Phenomena.* Dec. 1995, ISBN: 978-0-12505626-7. DOI: 10.1016/B978-012505626-7/50011-X.

[31] E. Tardif, M. M. Murray, R. Meylan, L. Spierer, and S. Clarke, “The spatio-temporal brain dynamics of processing and integrating sound localization cues in humans,” *Brain Res.*, vol. 1092, no. 1, pp. 161–176, May 2006, ISSN: 0006-8993. DOI: 10.1016/j.brainres.2006.03.095.

[32] A. P. G. Richard M. Warren and L. Krasner, *Auditory Perception. A New Synthesis.* Pergamon Press, 1982, ISBN: 9780080259574.

[33] C. Baumann, C. Rogers, and F. Massen, “Dynamic binaural sound localization based on variations of interaural time delays and system rotations,” *J. Acoust. Soc. Am.*, vol. 138, no. 2, pp. 635–650, Aug. 2015, ISSN: 0001-4966. DOI: 10.1121/1.4923448.

[34] G. Papadopoulos, K. Efstathiou, Y. Li, and A. Delis, “Development of a passive acoustic system for real time source detection, tracking and recognition,” in *[1992] Proceedings of the IEEE International Symposium on Industrial Electronics*, IEEE, May 1992, 460–463vol.1, ISBN: 978-0-7803-0042. DOI: 10.1109/ISIE.1992.279641.

[35] J. Capon, “High-resolution frequency-wavenumber spectrum analysis,” *Proc. IEEE*, vol. 57, no. 8, pp. 1408–1418, Aug. 1969, ISSN: 1558-2256. DOI: 10.1109/PROC.1969.7278.

[36] R. Roy and T. Kailath, “ESPRIT-estimation of signal parameters via rotational invariance techniques,” *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, no. 7, pp. 984–995, Jul. 1989, ISSN: 0096-3518. DOI: 10.1109/29.32276.

[37] T. Damarla, “Azimuth & elevation estimation using acoustic array,” in *2010 13th International Conference on Information Fusion*, IEEE, Jul. 2010, pp. 1–7. DOI: 10.1109/ICIF.2010.5711874.

[38] A. Pourmohammad and S. M. Ahadi, “N-dimensional N-microphone sound source localization,” *J. AUDIO SPEECH MUSIC PROC.*, vol. 2013, no. 1, pp. 1–19, Dec. 2013, ISSN: 1687-4722. DOI: 10.1186/1687-4722-2013-27.

[39] D. Bies and C. Hansen, “Engineering noise control - theory and practice,” in 4th ed. New York: CRC Press., 2009, pp. 18–19.

[40] “Indoor ultrasonic simulator (simlocus),” in [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/61364-indoor-ultrasonic-simulator (visited on 06/24/2022).

[41] D. F. Albuquerque, J. M. Vieira, S. I. Lopes, C. A. Bastos, and P. J. Ferreira, “Indoor acoustic simulator for ultrasonic broadband signals with doppler effect,” pp. 21–58, 2015, ISSN: 0003-682X. DOI: 10.1016/J.APACOUST.2015.04.010.

[42] “Matlab app designer,” in [Online]. Available: https://www.mathworks.com/products/matlab/app-designer.html (visited on 06/24/2022).

[43] “Roland quad-capture drivers,” in [Online]. Available: https://www.roland.com/global/support/by_product/quad-capture/updates_drivers (visited on 06/24/2022).

[44] “Audio hardware support for linux,” in [Online]. Available: https://wiki.linuxaudio.org/wiki/hardware_support#roland_-_quad-capture_ua-55_usb-2 (visited on 06/24/2022).

[45] *ReadFromMicrophoneAndWriteToAudioFileExample*, [Online; accessed 24. Jun. 2022]. [Online]. Available: https://www.mathworks.com/help/audio/ref/audiodevicereader-system-object.html.

[46] “Qt | Cross-platform software development for embedded & desktop,” in [Online; accessed 24. Jun. 2022]. [Online]. Available: https://www.qt.io.

[47] D. Cohen, “Precalculus: A problems-oriented approach,” in 6th ed. Brooks Cole, 2004.

[48] A. Moore and R. H. Miller, “Automated Identification of Optically Sensed Aphid (Homoptera: Aphidae) Wingbeat Waveforms,” *Ann. Entomol. Soc. Am.*, vol. 95, no. 1, pp. 1–8, Jan. 2002, ISSN: 0013-8746. DOI: 10.1603/0013-8746(2002)095[0001:AIOOSA]2.0.CO;2.

[49] T. S. Rappaport, “Wireless communications principles and practice,” in 2nd ed. Hoboken, New Jersey, USA: Prentice-Hall, 2010, p. 108.