TDOA-Based Multilateration Using Networked Devices for Real-Time Positioning and Visualisation

OLIVER BERGENDORFF & MARTIN BERGMAN
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

TDOA-Based Multilateration Using Networked Devices for Real-Time Positioning and Visualisation

Oliver Bergendorff
ol6704be-s@student.lu.se

Martin Bergman
dat14mb1@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Ove Edfors, ove.edfors@eit.lth.se

Examiner: Michael Lentmaier, michael.lentmaier@eit.lth.se

Carried out at Axis Communications AB

Company supervisors:
Astrid Rost, astrid.rost@axis.com
Fredrik Wällstedt, fredrik.wallstedt@axis.com

May 30, 2024

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Every day more devices all over the world are connected to networks. Some of these devices have microphones that could potentially be used in a networked array. This opens up the possibility to determine where sounds in the environment are produced, given the positions of the microphones are known. This thesis investigates the feasibility of using a number of network connected speakers with built-in microphones to track where a sound came from. The proposed system tries to position sounds by synchronising the clocks between devices and sending the audio streams from their microphones to a server to be processed. The server estimates the difference in arrival time of the signal between the devices. This results in a series of differences that can be used in multilateration to estimate the location of origin. The results are then visualised in a graph. The investigated system displays decent results when positioning sounds in a controlled environment, with differences in appropriate configurations for different types of sounds. The results show that more reliable synchronisation methods would allow for greater precision with the suggested approach.

Acknowledgements

We would like to thank Axis Communications for supporting us in doing this thesis work by supplying equipment and supervisors.

We would also like to send a big thank you to our supervisors Astrid and Fredrik for helping us find equipment, people who might be able to help us, and space to carry out testing, while also helping us stay on track to finish this thesis.

Finally we would like to thank our supervisor Ove Edfors for providing useful insights and ideas to improve our study in a multitude of ways.

Table of Contents

List of Figures

List of Tables

Acronyms

CI Confidence Interval. 38–41, 43, 55–58

DHCP Dynamic Host Configuration Protocol. 17

GCC-PHAT Generalized Cross-Correlation with Phase Transform. 8, 22–24

MAE Mean Absolute Error. 34, 38–41, 55–58

MAEBC Mean Absolute Error with Bias Correction. 37–41, 55–58

MSE Mean Square Error. 34, 39–41, 55–58

NTP Network Time Protocol. 2, 12, 13, 17

PoE Power over Ethernet. 17

PTP Precision Time Protocol. 13, 17, 46

RTP Real-time Transport Protocol. 21, 47

SNR Signal-to-Noise Ratio. 6, 11, 21, 23, 25

TDOA Time Difference of Arrival. v, vii, 7–9, 13–15, 18, 22–25, 37, 45, 50

UDP User Datagram Protocol. 21

Chapter 1
Introduction

This introduction chapter aims to give the reader an understanding of why this thesis was conducted, what previous work that has been done on the subject as well as determine limitations and scope for the thesis. Lastly it also introduces the research questions to be answered.

1.1 Background

The world is becoming more and more connected. Everything from fridges to speakers can be connected to a network. This opens up possibilities to aggregate data from multiple units, something which might not previously have been possible. This data collection comes with its own challenges, and a real-time system has high demands on latency, scaling and correct concurrency handling.

This thesis explores the possibilities of using an array of network connected microphones to determine the location of a sound source. Furthermore it aims to achieve an implementation that can be run in real-time within a reasonable processing budget, while still being able to display the position on e.g. a floor plan with enough accuracy to be useful. This could then be used in a surveillance system to detect disturbances and direct attention to a possibly problematic situation. For example detecting the sound of a window shattering, even if no camera is aimed directly at that window at the moment. An operator could then possibly investigate further with other surveillance equipment or get the position to check what happened in person by moving to the estimated location.

Axis Communications

Axis Communications develops and manufactures network devices such as cameras and speakers and surrounding technologies including management software. Many of their products contain different types of sensors, relevant to this thesis project is that many of them contain microphones. In a distributed surveillance system with microphones at several different known locations interesting data can be extracted about the environment and the sound sources within it.

This thesis project aims to estimate the position of a sound source based on data received by the microphones in a distributed system of Axis devices.

1.2 Research questions

The report aims to answer the following questions:

R1. How can position tracking of a noise source be implemented using networked devices with built-in microphones and known positions?

R2. What is such a system's performance and accuracy limits when dealing with different types of sound sources (e.g. short and sharp sounds compared to continuous sounds, or a specific known sound compared to a more vaguely classified sound)?

R3. How can the positional data be visualised?

R4. Is real-time analysis and visualisation of this data viable on standard hardware?

R5. What are the real world applications for this data?

1.3 Scope

To achieve a useful result and to make sure the project was achievable in the time given for a master's thesis some thought was given to limiting the scope of the project.

- The project will not develop or train any neural networks or machine learning algorithms.
- The project will not take extra steps to handle situations with more than one applicable noise source, although they may still be tested to ascertain the current system's limitations.
- The system will only attempt to position signals in two dimensions, and will therefore assume that the signal came from a predetermined plane.
- The system will be primarily tested in an office setting due to availability.
- The clocks on the sensor devices can be synchronised by a variety of means, but no novel synchronisation tools will be implemented for the thesis, and as such the live synchronisation is limited to the options that are available (NTP with a common server on a local network, or a basic implementation of ptpd2[19]).
- The project will attempt to be real time applicable, meaning low-cost-of-computation approaches will be generally favoured.

1.4 Related work

There are many previous studies conducted on the matter of locating a sound source with multiple sensors. A few have even been done at Axis Communications. In a thesis project by Chan & Karlsson[3] the goal was to detect a gun shot, classify it as such and then determine where it originated from. The results were promising

and showed that good enough synchronisation could be achieved to give satisfying location estimates. The proposed system did, however, have some drawbacks such as needing to calibrate it with a number of sounds first to compensate for latency in the solution. It also propose an edge detection algorithm to determine when an event occurs. This has the drawback of not being able to position continuous sounds, other than when they initially start.

Farzone and Smidje conducted a similar study[6] where a sound would be positioned and the estimated position would be used to aim a camera at that location. A setup with external microphones were used and the data was sent over the network to a server to be processed in real time. The resulting system could, to a satisfactory degree, estimate the location and aim the camera. This is very similar to what this thesis aims to achieve but instead of aiming a camera the intention is to show the location on a map in real-time.

Summary

In the above chapter a brief overview of previous work has been presented, the research questions established, and the scope and limitations clarified. With this as background, the next chapter will discuss the underlying theory.

Chapter 2
Theory

This chapter aims to give a brief explanation of the theoretical concepts it touches upon, and of the mathematics involved, as well as what factors will impact a positioning system's precision.

2.1 Introduction to acoustic localisation

Localisation using acoustics is a very common way of determining the position of a sound source. For example, it is used by animals such as bats which use echolocation as a navigational tool. It is also used in a different way in military operations to, for example, locate artillery weapons. Humans are able to use these techniques instinctively to some extent with some people, particularly people with poor eye sight, using it as a navigational tool in everyday life[13].

Passive and active localisation

The example of bats' echolocation is a form of *active* localisation, where the object being positioned emits a signal and the response to that signal (i.e. the echo) is used to gain information about the surroundings or the object's own position. This is the same technique used in sonar devices in submarines where a loud ping is emitted. The technique differs from *passive* localisation. In passive localisation the signal is instead assumed to be present in the environment, having been emitted by an external source, rather than by the positioning system, such as in the case with military positioning of artillery weapons. Certain techniques, discussed later, are used to separate what is perceived to be the real signal from the background signal, i.e. the noise. This means that a system using passive localisation can theoretically position any signal (within the system's frequency range). In practice this requires a certain amount of filtering to extract signal from the background noise of the area as the calculated position of said noise would end up being essentially random in many real use cases. This filtering can be done in several different ways including, but not limited to, standard signal processing filters, volume thresholds, correlation thresholds and sound classifiers[11].

2.2 Comparing signals

For a signal to be useful to a positioning system it needs to be identified at several different sensors. Identifying which signals are the same is not a trivial matter. A signal captured at two different sensors will rarely, if ever, be exactly the same in a real life scenario. Electromagnetic radiation travelling through a sparse medium or a vacuum, such as radio waves sent into space, lose very little of their information over distance compared to sound waves. Signals degrade in intensity over distance according to the inverse square law[17]. Electromagnetic radiation carries its own energy and can propagate even without the need of a medium. Sound waves are elastic waves and therefore propagate by actually moving physical particles, necessitating a medium to travel through. Propagating through a medium causes the waves to weaken over distance as they transfer part of their energy to the surrounding medium[4]. This also attenuates the signal, changing its frequency content.

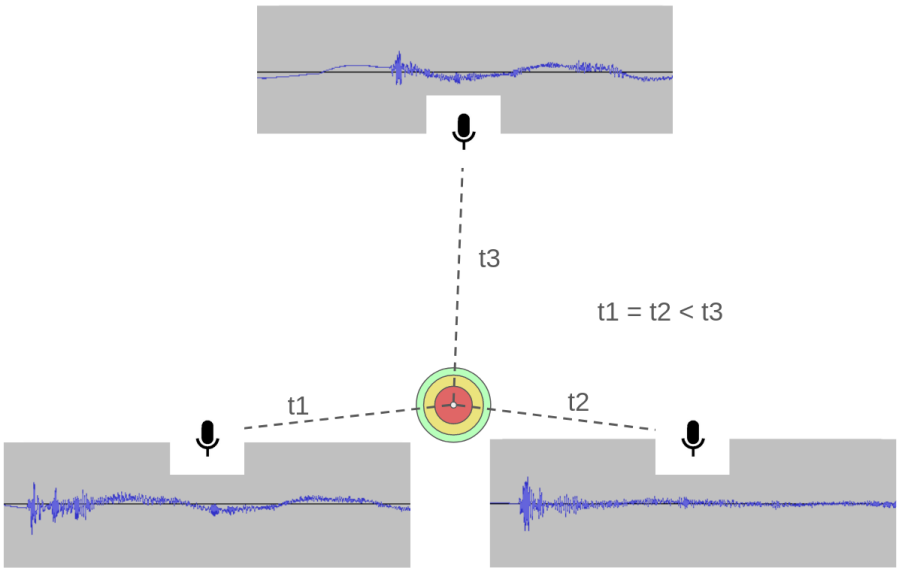


Figure 2.1: Illustration of the differences in a snap signal captured at different sensors

Signals also move at different speeds through different mediums. The speed at which a sound propagates changes when it moves through even very small variations in temperature and pressure within the same environment. The rate of change in these properties is also reliant on the frequency content of the specific signal. This means that audio signals travelling through air will quickly change and get a different representation.

The weaker a signal is the lower the Signal-to-Noise Ratio (SNR) becomes, and the harder it is to extract useful data. Therefore the most that can be done with an audio signal in a real scenario is to establish a quantifiable measure of similarity between two signals, and use that information to determine when two

signals are similar enough to be considered the same.

In Figure 2.1 the effects of an acoustic signal degrading can be clearly seen in data extracted from a real test where data was collected from three of the microphones used in the system. A single signal, in this case a person snapping, gets represented slightly differently and arrives at slightly different times in the different sensors. There is a clear similarity between the signals visually, but they are not the same. Thus the challenge is quantifying this similarity and extracting useful positioning data from it. For example a simple assumption might be that a signal with a low amplitude has come from further away than a signal with a higher amplitude, but this might not always be the case. Different frequencies are attenuated at varying rates in the atmosphere based on factors that are difficult to account for [8] without already having information on the sound's source position or frequency content. This makes the relative distance of a sound with previously unknown characteristics impossible to calculate accurately based purely on the amplitude. Instead it is clear that another measure of similarity between signals is needed, one that can hopefully also extract some other useful information about the signal, such as the delay between the times when the signals arrive at the sensors. There exists a simple such measure, and it is the cross-correlation of the two signals. Cross-correlation effectively measures the similarity between two signals as a function of the relative delay[5], something that is very useful for an audio positioning system.

The cross-correlation of a continuous real valued signal is defined as:

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} f(t)g(t+\tau)dt \tag{2.1}$$

or more specifically, for discrete values

$$(f \star g)[n] \triangleq \sum_{m=-\infty}^{\infty} f[m]g[m+n]. \tag{2.2}$$

When these are applied to two signals that are identical, or identical except for scaling of the amplitude, the output will peak at zero lag $n = 0$. If the signals have that same relation but are also shifted by a certain amount the peak of the cross-correlation will in turn be shifted by that same amount. In the case of digital audio extracted through microphones, this means that the amount of samples that one signal is shifted compared to another can be calculated through the discrete case of the above equations, by simply returning the index of the maximum value in the output.

The cross-correlation is a powerful enough tool that it can return a quantifiable measurement of how similar they are, and with which offset they would be most similar, even though they are not the same. Theoretically this should be enough for a solution to the Time Difference of Arrival (TDOA) calculation, but testing showed that it is not very resilient to the types of acoustic interference that will appear in a real life scenario. The algorithm would theoretically give the correct estimate of the TDOA in an environment with no reverberance, no other sound sources, and a completely even medium for signal propagation. A real acoustic environment very rarely exhibits these traits, and as such there is a need for a method that is more robust to this kind of signal degradation.

2.2.1 Generalized Cross-Correlation with Phase Transform (GCC-PHAT)

With multiple similar devices placed in the same environment, as required for the system to work, the noise contributed by the environment itself will inevitable be similar between devices. Noise from objects in the environment and from electrical interference in the devices will therefore show a high correlation. Thankfully, most of the information that is important specifically to calculating the offset between the two signals is preserved in the phase information.

GCC-PHAT works by implementing a weighting component to the cross-correlation which makes it less prone to uncorrelated noise. The weight is based on normalisation and multiplies the samples with the inverse of the absolute value, ideally resulting in samples of unit amplitude in the frequency domain. This results in ignoring amplitude information while preserving the phase. This can have benefits in reverberant environments and other less optimal conditions.[12]

2.3 Multilateration

Given just a single sensor and the arrival time of a signal all that can be deduced is that the signal was sent before it arrived to the sensor. This information is not very useful for positioning the source.

However, with two sensors there is a lot more information to be gained. Using the TDOA information about the source position can be extracted. First, the simple two dimensional case is considered, with two sensors and two arrival times of a signal at those sensors, t_1 and t_2.

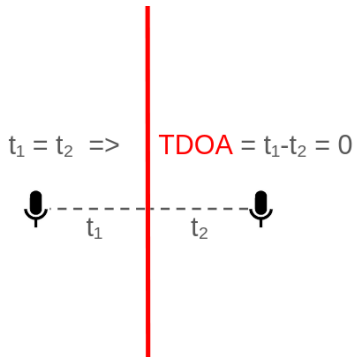


Figure 2.2: Possible sources of a signal arriving at both sensors simultaneously

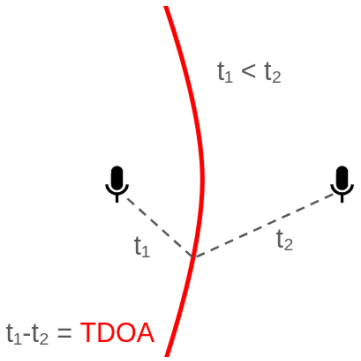


Figure 2.3: Possible sources of a signal with a specific earlier arrival time at the leftmost sensor

With two sensors, if the signal arrives at the same time (given that the signal has travelled through a relatively evenly distributed medium) it can be deduced that the signal was sent from a point with roughly equal distance from the two sensors. This set of possible points forms a straight line as seen in Figure 2.2.

If the signal is instead shifted in time in the two sensors the difference in distance can be calculated by calculating the extra distance that the signal would travel during that time offset. Then all possible points that would have that distance offset form one half of a hyperbola as demonstrated in Figure 2.3.

These two cases both present a simplified version where only the intersection between the $z = 0$ plane is visualised. More generally the TDOA problem can be framed in three dimensions as

$$\begin{aligned} \delta_1 &= \sqrt{(x_t - x_1)^2 + (y_t - y_1)^2 + (z_t - z_1)^2} \\ \delta_2 &= \sqrt{(x_t - x_2)^2 + (y_t - y_2)^2 + (z_t - z_2)^2} \\ \delta_1 - \delta_2 &= c(t_1 - t_2) \end{aligned} \tag{2.3}$$

where c represents the speed of the signal. Solving for (x, y, z) provides a hyperboloid which is generally formulated as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = -1 \tag{2.4}$$

More specifically, since the sign of the TDOA is known, the system solves to one half of a hyperboloid. Within the scope of this thesis it is assumed that the signal originates from a point within a plane spanned by the sensors, similarly to the case in figures 2.2 and 2.3. As such the points of interest are on the line of intersection between the hyperboloid and the plane spanned by the sensors, which can be assumed to be $z = 0$. This simplifies the equation to the two dimensional case for the TDOA calculation, as

$$\begin{aligned} \delta_1 &= \sqrt{(x_t - x_1)^2 + (y_t - y_1)^2} \\ \delta_2 &= \sqrt{(x_t - x_2)^2 + (y_t - y_2)^2} \\ \delta_1 - \delta_2 &= c(t_1 - t_2) \end{aligned} \tag{2.5}$$

This system solves to a hyperbola as

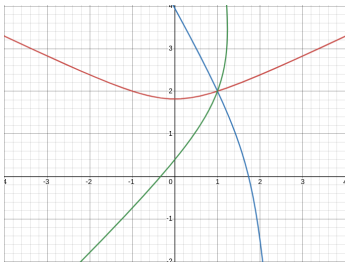
$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1 \tag{2.6}$$

Since the sign of the TDOA is still known the final solution is only one of the arcs of the above hyperbola.

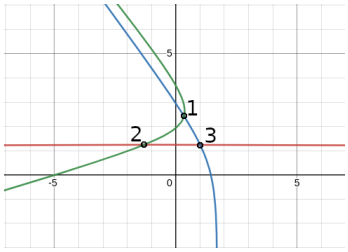
Now, assuming that the signal did originate from the given plane, the TDOA can be used to calculate a parabolic function of possible origin points.

To actually position a signal to more accuracy than any point along this infinite parabola a third sensor is needed. Each new sensor provides TDOA measurements from each other sensor, so with three sensors three TDOA measurements can be calculated.

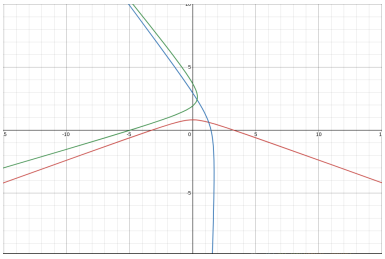
In an ideal environment with perfect signal preservation and no synchronisation errors these three measurements can actually be perfectly represented by just two values. With the time when the signal first arrives at a sensor as T_0 and the arrival times at the other two sensors as T_1 and T_2, the two TDOA values $(T_0 - T_1)$ and $(T_0 - T_2)$ also contain all the information necessary to construct



(a) Hyperbolas that all intersect in a single point.



(b) Hyperbolas that intersect in 3 distinct points.



(c) Hyperbolas where some of the parabolas never intersect

Figure 2.4: Different cases for how the hyperbolas might intersect

the third value $(T_1 - T_2)$. This holds true for any number of sensors, as all events $(T_0 - T_N)$ can be placed on the same timeline with N comparisons to T_0. In this idealised world the solution to the positioning problem would be found quite easily at the intersection of the two parabolas in a closed-form solution[14]. If the third parabola is calculated anyway, just for the sake of completeness, it will intersect the same point, and the solution is found in that point as seen in Figure 2.4a.

Unfortunately, once again a real acoustic environment is a limiting factor, and while calculating two parabolas does (usually) lead to *a* solution, it is not necessarily an exact nor a theoretically correct solution.

Environmental noise and inter-device synchronisation will inevitably lead to shifts in the hyperbolas by some amount, and even a small shift will lead to a different case, where calculating all three hyperbolas will yield not a single intersection point, but three, as seen in Figure 2.4b

These three points are therefore all presented as possible solutions to the positioning problem. It is possible at this point that there is a singular solution to the problem but that it lies above or below the plane that the calculations assume, needing a three dimensional solver to be perfectly positioned. Within the scope of the thesis, however, it can be assumed that the sound did originate in the plane, so any solution outside of the plane would be inaccurate by default, and would only be possible through some sort of measurement error. The thesis also assumes that a sound has a singular point as a source even if this may not always be the case, introducing another source of errors that will offset the hyperbolas. With this in mind the centroid of the triangle, the mean of all intersection points, can

be used as an acceptable approximation of the true solution in this simple case.

Another problem arises when the hyperbolas do not form three intersections that lie in the plane. Depending on the magnitude of noise and the actual differences between the arrival times of the signals between 0 and 6 intersections could actually form. When this happens the positioning problem has much less obvious solutions. As can be seen in Figure 2.4c the previous method does not yield a satisfactory answer in this case. One could also easily imagine a case where a small shift in one of these hyperbola that would otherwise form an intersection near the correct point causes an intersection very far away.

Given an input signal with enough noise there will inevitably be occurrences of this nature. There are different ways to handle these cases, but it is not easy to know which way will be more accurate for any specific case. Without knowing the nature of the noise well enough to accurately compensate for it the divergent hyperbolas cannot be avoided, but if that knowledge was possible to extract there would be no divergent hyperbola after applying said compensation. There is also the issue of sensor placement, which heavily impacts both the likelihood of a divergent hyperbola and the potential impact of it. In general signals that come from outside the area spanned by the devices have a smaller tolerance for divergence and experience a larger shift in the position of intersections, while signals that come from an approximately equidistant point from all the sensors (i.e. the centre of the area spanned by the sensors) experience the smallest shift.

With all this in mind choosing which estimation to return from a case where parabolas diverge comes down to how the system values false negatives relative to false positives.

The simplest case is returning a negative whenever an intersection between two of the parabolas can not be found, and treating it as not having gotten a signal, which would constitute a false negative. Other estimation techniques, such as choosing the centroid of the intersection points that were actually found, will lead to errors, and as such risk constituting a false positive, even if the answer might be within error tolerance.

With more than three sensors a more precise location estimate can be achieved. It can be assumed that the inaccuracies introduced by noise and synchronisation errors don't lead to an offset in any specific direction as long as the sensors are well distributed and are frequently re-synchronised. If that is the case then any offsets introduced should follow a distribution centred on the correct point. This means that as more sensors are used the average position of each intersection point should converge closer to the correct solution. However, the further a sensor is from the signal the worse the SNR becomes, meaning that introducing more sensors will likely not improve the accuracy if they are not sufficiently close to the sound source, and could instead reduce it. This accuracy loss can be mitigated by weighting the values based on some factor that correlates with signal strength, such as overall measures amplitude, or the value of the correlation peak with other sensors.

With at least four sensors it is also possible to produce a solution in three dimensions, as long as the sensors themselves span three axes, i.e. they are not positioned in the same plane. This has the same limitations as the previous case, meaning that in most real cases the hyperboloids will likely not intersect in a

single point, but in several different points in space. Theoretically this holds for N sensors that span $N-1$ dimensions, but the practical use cases for more than three dimensions are limited.

2.4 Limiting factors for positional accuracy

There are many factors that affect the maximal accuracy a positioning system is able to achieve. No system will be able to position every signal perfectly, instead it is more reasonable to measure the system's accuracy by looking at the average error when it positions a signal well, as well as the amount of false positives, false negatives and true negatives. Some specific factors are important enough for the accuracy that they deserve separate consideration.

2.4.1 Device synchronisation

Once the similarity of two signals has been established, and the estimation of the time offset has been calculated, one issue remains. Within a distributed system such as the one in this thesis each device has its own clock, and sending data across a network does not take a consistent amount of time, so the arrival time of the packets will not be a reliable measurement of when they were captured. To solve this the data is tagged with a timestamp of when the signal was captured. The data point that is most important for positioning, the time offset between when the signals arrived at each device, is completely dependent on this synchronisation for the comparison to be valid. As such the accuracy in the time difference estimation is limited by the synchronisation between devices.

There are many ways to synchronise device clocks, the most common of which uses NTP servers [15]. This provides a baseline level of synchronisation between the units that is acceptable for most users, as it tends to keep accuracy to the tens of milliseconds, increasing to about a millisecond in optimal scenarios such as a local network with a dedicated NTP server.

Network Time Protocol (NTP)

Network Time Protocol or NTP[15] for short is a very commonly used synchronisation method used by computers today. A common use case is to automatically synchronise the clock in a computer to an NTP-server that acts like a master clock and tells the client what time it is. This allows for computers to set the correct time without the user having to input it manually. It also comes with the added benefit of compensating for clock drift automatically. All clocks drift by some amount, meaning they deviate from the correct time. Each second will not be treated as exactly one second and over time these small errors compound and result in an increasingly incorrect clock until synchronised again. The solution to this is to just let the computer re-synchronise its clock with the NTP-server on regular intervals to ensure that the clock does not drift too far. NTP is not ideal in time sensitive scenarios since the synchronisation can vary greatly. With an average internet connection it can reach synchronisation of around 10 milliseconds which would result in an accuracy loss of about 3 meters for each measured value.

Under ideal circumstances NTP could reach an accuracy of less than 1 millisecond but in worst case could result in over hundreds of milliseconds, and it is difficult to confirm how well synchronisation is working.

Precision Time Protocol (PTP)

Precision Time Protocol or PTP[18] is another synchronisation method that can achieve much better synchronisation between units than NTP. It is more commonly used in scenarios where accurate synchronisation is important such as in digital sound equipment for live audio applications, and situations such as the one covered in this thesis. PTP has an array of features to improve synchronisation over NTP such as better latency compensation and selection of a so-called grandmaster clock in the network. The grandmaster clock is the reference time all the other devices tries to synchronise against. The PTP network automatically detects the clock source that is deemed the best one. If a new device is added which is deemed a better clock by the network, the grandmaster clock is changed to the new device. To ensure the devices are synchronised, the grandmaster clock sends out a *Sync event* as well as the time that it was sent. The receiver notes the time it thinks the messages arrived and responds with a Delay request to the grandmaster clock. The grandmaster then responds with when it received the *Delay request*. Now the receiver knows when the *Sync* and *Delay request* where sent and received according to the two parties. This is enough for the receiver to accurately calculate how much to adjust the clock to be in sync with the grandmaster.

2.4.2 Speed of sound

The thesis project system uses microphones, which are sensors that specifically measure acoustic signals, i.e. sound. Sound travels at a speed of approximately 343 metres per second in normal pressure and room temperature air [17]. This speed varies depending on multiple factors, such as temperature, air pressure and humidity. However, these factors can be assumed to be close enough to equal across all devices across the system as they are all within the same network and therefore very likely to be facing the same atmospheric conditions. Due to this relative homogeneity the speed of sound is simply assumed to be a static 343 metres per second within the system.

When measuring the TDOA between multiple distributed sensors the accuracy that would be lost with a certain amount of desynchronisation can therefore be approximated as

$$\begin{aligned} |\Delta\text{TDOA}| &= |\text{TDOA}_{\text{estimated}} - \text{TDOA}_{\text{expected}}| \\ \text{Hyperbola}_{\text{error}} &= 343 * |\Delta\text{TDOA}|. \end{aligned} \tag{2.7}$$

Thus, a synchronisation error of a single second would lead to an accuracy loss of up to 343 meters in the estimated difference in arrival time, something that would render the system completely useless at estimating the location of an acoustic source with any sort of accuracy in most environments. The sound would need to be extremely loud to be measurable in microphones at large enough distances that an intersection point being shifted by 343 meters would still provide an acceptable

estimate, and a hyperbola being shifted by 343 meters might shift the intersection points by much more than that. As such it is clear that a much higher degree of synchronisation than a second is needed for the calculations to be correct.

A similar positioning system to the one presented in this thesis could be constructed to position any type of signal, and the importance of synchronisation would depend heavily on the speed of that signal. If a similar system was set up at room scale intended to measure the TDOA of light emitted by some source the synchronisation demands would be extremely tight, while a system working at an intergalactic scale might not care about losing a few thousand kilometres in accuracy. Luckily, sound is a relatively slow moving signal, and there are several synchronisation methods that can operate within the necessary accuracy to give a decent result, as explained in Section 2.4.1.

2.4.3 Signal characteristics

The precision one can achieve in a positioning system is also dependent on the characteristics of the signal itself. When it comes to sound a sharp, clearly defined sound can be positioned with a higher resolution than a long repetitive sound. Consider the case of a basic sinusoidal tone, such as a standard A note at 440 Hz. Each second the signal oscillates 440 times, and as such a second long signal of this tone would have 440 different time offsets where the signals would be considered the same, and it would become impossible to extract which specific offset is most correct based on just the phase information. In contrast a single instance of very loud sound preceded by and followed by complete silence would only have one overlap where the peaks would align, leading to a much more obvious correct answer. All signals in a real life scenario will probably end up on a scale somewhere in between these examples, with overall sharper sounds, such as a snap or clap, being easier to correctly position, and more periodic sounds, such as speech or string instruments, being harder.

2.4.4 Reverberance and reflection

When using a system based on this theory in a real acoustic scenario it is unlikely that the area is acoustically perfect, and that all signals have taken the shortest route to the sensor. The ideal choice of locale for accurate positioning would be an anechoic chamber where as much sound as possible is absorbed by the walls, and almost nothing is reflected. Normally sound reflects off of walls and objects in the room and these reflections can cause false correlation spikes and lead to inaccurate positioning.

An example of this can be seen in Figure 2.5, where a wall leads to a correlation being found with a much larger t_2 than the line-of-sight case, which will lead to an inaccurate position. This problem would be further exacerbated if there was a wall positioned in between the source and one of the sensors. This would need to be included in the TDOA calculations, which is out of scope for this thesis project. As such it is assumed at the basic level that there are no reflections, and that all signals have taken a line-of-sight path to the sensor, which will sometimes lead to false positives. However, since these reflected signals have grown weaker due to

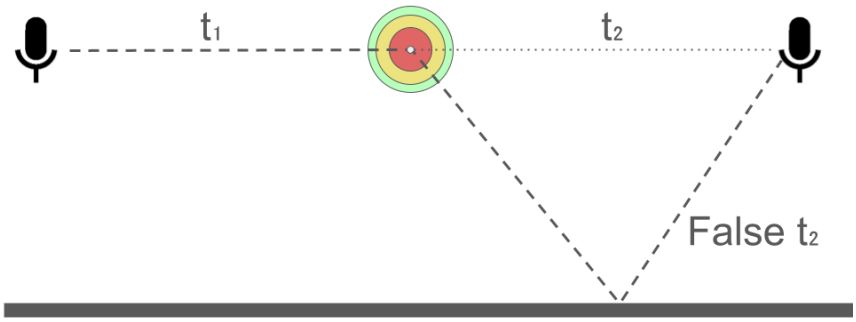


Figure 2.5: A basic example of reflection off of a wall leading to a false TDOA

having taken a longer path the correlations can often fall under the correlation threshold due to the lack of signal strength, so the system is partially resistant to this case.

Summary

This chapter has provided the reader with some insight in the fundamental concepts of this thesis. It has presented information on the topics of acoustic localisation with techniques such as multilateration. It also introduces signal comparison with cross correlation and some factors that might impact the positional accuracy. This should be a sufficient foundation to grasp the upcoming chapter that describes the method used as well as some details of the implementation.

Chapter 3
Method

This chapters purpose is to describe how the research in the thesis was conducted. Further it describes how the proof-of-concept system was implemented as well as how the testing was carried out.

3.1 Equipment

3.1.1 Axis Devices

The system was implemented to be used with a set of identical network speakers. The devices are powered using Power over Ethernet, have a 2.5-inch speaker and a built in microphone. They run a version of Linux OS and have built in functionality to synchronise its clock using NTP, either automatically or by setting a specific NTP-server as a source. Software is also publicly available to synchronise the devices with an open source PTP[18] implementation called ptpd2[19]. It is also possible to adjust the gain level of the microphone from the interface provided by the device. The microphones have a frequency response of 50-12000 Hz, and as such any signal should be in that range to be represented as accurately as possible. It is still possible that signal outside this range is captured, but the amplitude might not be correct.

3.1.2 Network

The network used for testing, as illustrated in Figure 3.1, consisted mainly of a router, a Power over Ethernet (PoE) capable switch and regular twisted pair cables. The switch is responsible for power delivery and connections to the network speakers. The router is mainly used for DHCP capabilities as well as allowing a computer to connect to the network using WiFi. The network has no access to the internet and is thus solely a local network.

3.2 Implementation overview

Previous work at Axis [3] [6] used C code to retrieve data and MATLAB to process it, often taking several seconds to classify and analyse data. This is an approach

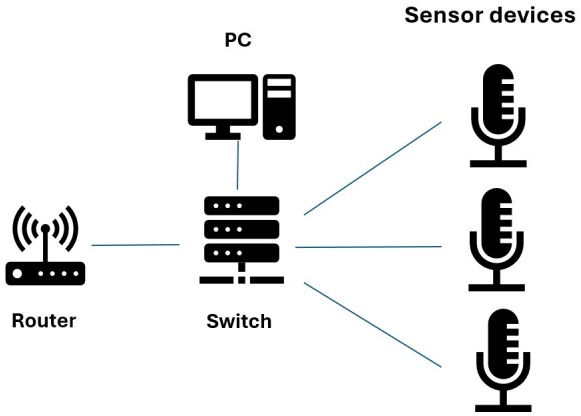


Figure 3.1: Network setup used in testing

that will make it difficult to get results in real time. Instead a synchronous pipeline approach was chosen.

The system was constructed as a series of Python modules[23] to which a configuration file could be supplied to control the parameters of the system. Python is flexible enough to be compatible with the streaming pipelines that Axis devices already use, while also being able to perform the calculations necessary to determine the signal source location. It also simplified the process of setting up a replicable environment for potential further development of the system.

The system presented in this thesis consists of a set of physical devices to collect and send audio data as well as a Python program that uses threads to synchronously perform tasks from all the following modules:

- Data retrieval
- Signal filtering
- TDOA calculation
- Positioning
- Presentation

A visualisation of the system's pipeline can be found in Figure 3.2. The implementation to send data from the devices to a specific target, in this case an IP address, was already done, barring some configuration changes. All the work listed above was implemented in Python code as part of this thesis.

3.3 Producing sound

To test the system two main methods of producing sound was used, snaps and speech. This was done to cover both ends of the spectrum of sharp sounds and non-sharp sounds as described in Section 2.4.3.

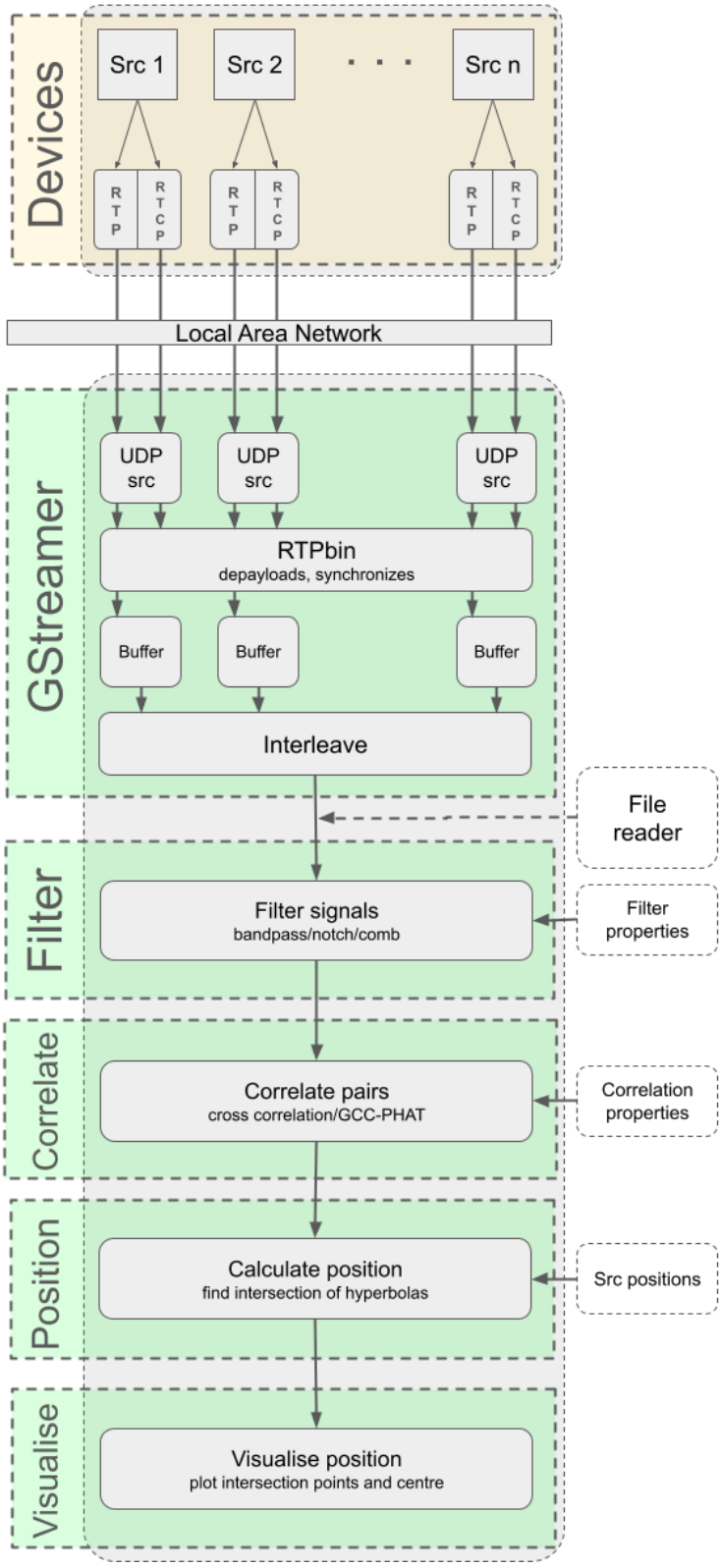


Figure 3.2: Graphical representation of pipeline

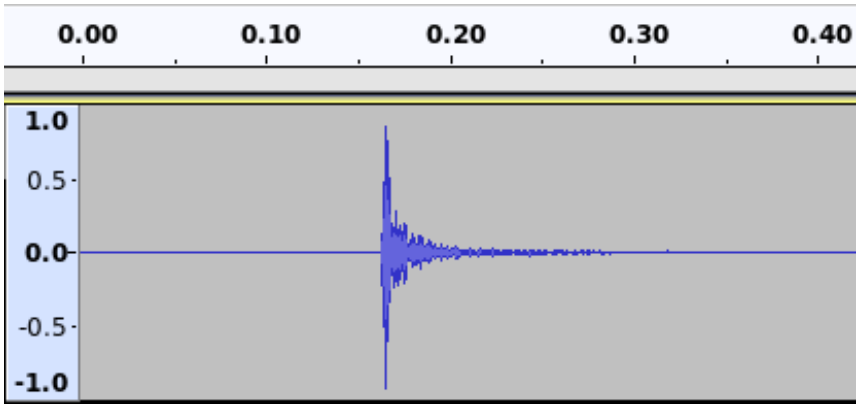


Figure 3.3: Example waveform of a finger snap

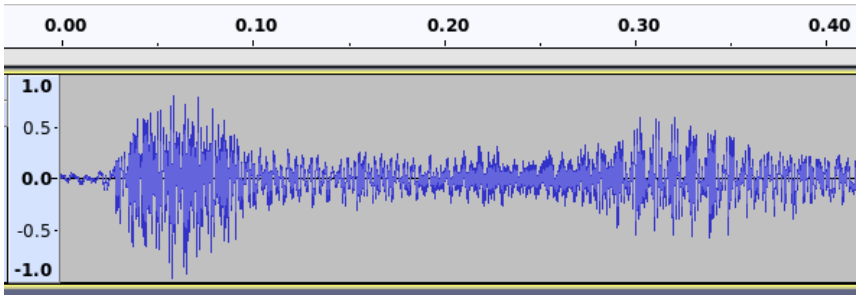


Figure 3.4: Example waveform of speech

3.3.1 Snaps

The snaps were simply produced by snapping with the fingers in the same plane as the sensors. This results in a sharp short sound that does not reverberate a lot. A visualisation of this can be found in Figure 3.3

3.3.2 Speech

The speech was produced by pronouncing a series of so called Harvard Sentences. Harvard Sentences are phonetically balanced and are intended to be used in testing speech quality[1]. An example of such a sentence is "Smoky fires lack flame and heat.", which is one of the sentences used for testing in this thesis. This ensures a more fair comparison between measurements since the same sentences are repeated in every instance of testing. The sounds produced this way is more continuous and not as sharp as a snap. An example of this can be seen in Figure 3.4.

Another difference is that while the sound of a speaking person will spread outwards in every direction, it will be stronger in a cone in front of the person. This is not the case for something like a snap, whose sound spreads out more evenly. Given that the theory requires a signal to arrive at multiple sensors to be able to position a sound, the angle the speaker is facing can also become important.

3.4 Retrieving audio data

The units developed by Axis can send out audio data in several different ways. However, many of these ways introduce latency which for normal use is well within reasonable limits, but which is difficult to compensate for within the limitations of this thesis. Using the API to extract the data results in latency differences far above acceptable limits, so the system is limited to using lower latency methods to meet synchronisation demands.

Instead a GStreamer-based application on the devices is used to capture audio, timestamp it and send it in RTP packets over a network. This allows for a configuration file to be supplied to the device with some options for latency profiles and clock synchronisation. This module is based on the GStreamer library, a multimedia framework. GStreamer has features that allow it to manipulate and synchronise multiple data streams, and has a Python binding scheme, making it a good choice for the capturing end of the pipeline as well.[7]

The GStreamer pipeline that was implemented captures the RTP packets sent over UDP by the GStreamer application on the devices, aligns the data using the timestamps, interleaves it so that adjacent samples were captured at the same time, and sends it to the next step of the program, which filters the signal.

3.5 Signal Filtering

As the thesis had high synchronisation requirements the audio data was fetched as early in the audio pipeline as possible to avoid scheduler inaccuracies, meaning minimal audio processing had been performed on the device. This lead to the audio data having a weak Signal-to-Noise Ratio, necessitating the use of audio filters. Depending on what frequency band the signal is expected to be in the system has very different filtering needs. One relatively predictable noise source is electrical noise. With the gain needed to get a good signal there was a large amount of electrical noise at 50 Hz, which could be reduced using a notch filter or a comb filter to also get rid of higher resonant frequencies. A graph of the noise profile during the benchmarking process can be seen in Figure 3.5, where a clear spike is seen at 50 Hz and many of its resonant frequencies. There is also a large amount of generally low frequency background noise probably mainly caused by ventilation systems which necessitated the use of a high pass filter or bandpass filter.

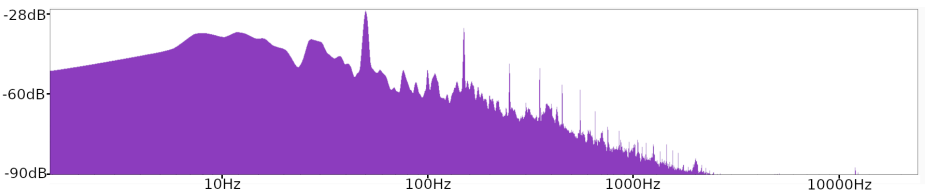


Figure 3.5: Frequency content of unfiltered background noise in the benchmarking environment

For this purpose the Python package SciPy[20] has several useful functions. A bandpass filter was chosen due to the possibility of filtering out the low frequency hum while maintaining most of the signal. It was implemented as a Butterworth filter of the 4th order with cascaded second-order functions and forward-backward filtering to preserve delay information (which also essentially makes it an 8th order filter). The bandpass filter reduced the background noise to the frequency content seen in Figure 3.6.

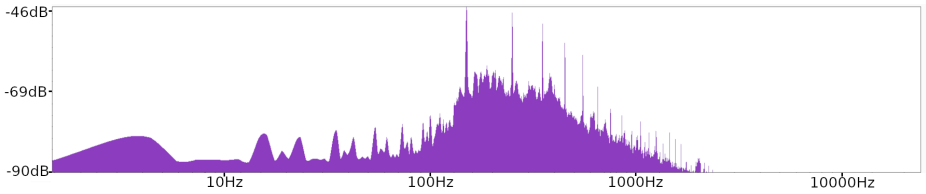


Figure 3.6: Frequency content of bandpass filtered background noise

When benchmarking the system both speech and sharper sounds such as snaps were used as described in Section 3.3. In the case of snaps, bandpass filtering lead to the the frequency content seen in Figure 3.7.

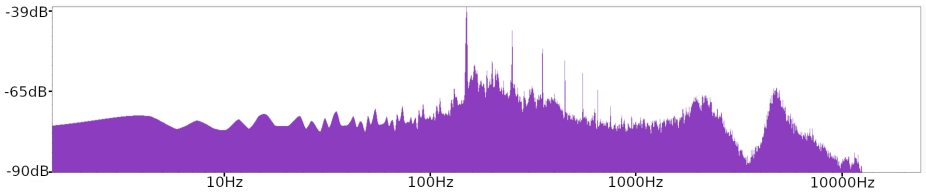


Figure 3.7: Frequency content of bandpass filtered snap signal and surrounding noise

A notch filter (as well as a series of intermittent notch filters, i.e. a comb filter) was also tested to filter out the electrical noise but deemed to provide too little benefit for the cost of phase shifting near the affected frequencies, which decreases the accuracy of phase-based TDOA estimation techniques for signals in those bands.

3.5.1 Noise injection

It was found that while the bandpass filter worked decently for some methods for TDOA estimation (cross-correlation) it worked less well for the more robust GCC-PHAT. The electrical noise in the background was weak compared to the signal, but it was strongly correlated between units as it had a very consistent frequency of 50 Hz (and its harmonics). This lead to the system attempting to position the background noise at an equidistant point from all sensors, as the electrical noise was synchronised between the units, since they were all plugged into the same local grid.

This was solved by dithering the signal by adding some amount of randomness onto a periodic signal to reduce its periodicity. In implementation this was done by adding random Gaussian noise with a median of 0 as to not influence the mean amplitude of the signal. The standard distribution was chosen based on the typical standard distribution of the noise that the system encountered, scaled with a configurable factor from 0 (no added noise) to any float number. Any number close to or above 1 would lead to massive degradation of the overall Signal-to-Noise Ratio as most of the signal would be hidden by the random noise. In practice, numbers between 0.02-0.15 seemed to be best at preserving signal while reducing the chances of either correlating based on the electrical noise or random TDOA estimates due to too much injected noise.

3.6 Calculating the TDOA

The TDOA is calculated by taking the data buffers received from the devices, collating them, and pairwise calculating the cross correlations on the collected audio data. The collating is performed by appending the latest buffer to the previous buffer. This was done to prevent the system from missing a signal that ends up in different buffers at two devices because of the time difference. Essentially each set of data is checked for correlation twice, once grouped with the set of data that came before it and once with the set that came after it. The minimum buffer size was chosen to ensure that any signal emitted within the zone that the system operates in will arrive with at most one buffer of time difference according to

$$
\begin{aligned}
&n_{\text{sensors}} = \text{Number of sensors} \\
&b_{\text{sample}} = \frac{\text{Bytes}}{\text{sample}} = 2 \\
&\Delta_{max} = \frac{D_{max}}{v} = \frac{\text{Maximum distance between sensors}}{\text{Speed of sound}} \\
&S = 48\text{kHz} = 48000 \text{ samples/s} \\
&B_{size} = \text{Buffer size} \geq n_{\text{sensors}} * b_{\text{sample}} * \Delta_{max} * S.
\end{aligned}
\tag{3.1}
$$

Once the buffers are collated the cross-correlation is calculated, yielding a set of TDOAs expressed in number of samples. Given a known sample rate, these differences in samples can be converted into time shifts in seconds. By extension one can convert the difference in seconds to a difference in distance the signal has travelled by incorporating an estimate of the speed of the signal as explained in Section 2.3.

In the code implemented for the system these calculations are done using either the SciPy[20] or NumPy[16] packages, depending on if the regular cross-correlation or the GCC-PHAT solution is selected, respectively. The regular cross-correlation uses SciPy's built in function signal.correlate() from the signal package[22].

The GCC-PHAT implementation is slightly more complex. First the Fast Fourier Transforms (FFT) for the two signals are calculated. Then one of the signals is multiplied with the conjugate of the other signal which gives the frequency

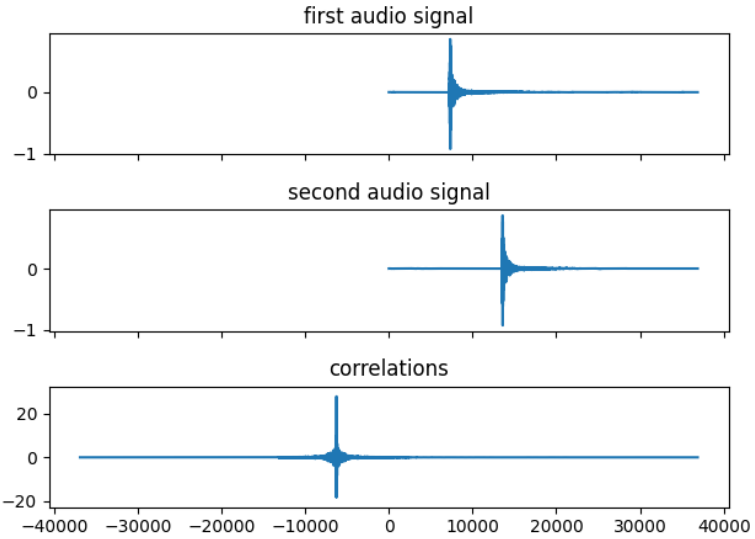


Figure 3.8: Example of cross correlation

function of the cross-correlation. The next step is dividing the elements of the multiplied signals with their respective absolute values. This is what differentiates the regular cross-correlation from the GCC-PHAT. Once this is done the Inverse FFT is calculated to return the signal to the time domain.

The next step is finding where the maximum value is located in the signal by finding its index. Once the index and the length of the signal is know, it is possible to determine the time offset in samples. If the middle sample of the signal is assigned to be index 0 and index x is the sample where the maximum was located, then the offset is the signed distance between index 0 and x. An example of this can be found in Figure 3.8 where the peak can be seen at around -7000 samples. Meaning the peak of the first audio signal occurs roughly 7000 samples before the peak of the second audio signal. This can in turn be converted in to seconds since the sample rate is known.

3.7 Estimating the position of the sound source

To estimate the position of the sound source the TDOAs and the locations of the sensors are used to construct a set of parabolas as explained in Section 2.3. To find the intersections of two hyperbolas the SciPy package was once again used in the implementation. This time using fsolve from the optimize package[21]. fsolve solves the equation by finding a root to the constructed equations using methods that are optimised for efficiency. This resulted in a set of points representing the intersections of the hyperbolas as explained in section 2.3. This set of points can be quite large, as with 6 sensors over a hundred intersections can be found. Therefore some methods of removing the less likely estimates are needed.

Any calculated intersection that is not deemed to be a "reasonable estimate"

is thrown out. A reasonable estimate can be defined in several ways, but it should be obvious that an intersection several thousand kilometres away is not something that could be picked up by this system, and allowing those values to be part of the overall calculation made the estimated positions completely useless when the SNR was low. As such a reasonable estimate for the benchmarking case was chosen to be any position within the area spanned by the sensors, or outside it by less than a configurable buffer zone of a few metres. The physical room where the benchmarking took place was itself smaller than this buffer zone, and as such the removal of any values outside this zone should not have removed any correct estimates from the calculations. This subset of possibly valid intersections then go through a further selection process.

It is unlikely that an intersection calculated from sensors that received little signal is more accurate than an intersection from two sensors with a better SNR. Because of this the intersections were sorted based on the average correlation strength of the TDOA estimates that they were calculated from, and a subset of the first n intersections (configurable in the configuration file) was used to calculate the actual position. As correlation strength is simply a measurement of the highest absolute value of the correlation it does not automatically imply that any intersections calculated from that TDOA value are better predictors of the position. The measurement is, however, highly correlated with signal strength, which should in turn be highly correlated with TDOA estimation accuracy.

From the points in this subset an estimate of the true position is taken in one of two ways. The simplest method was to simply take the average of all points, which is a low complexity calculation, but not very resilient to outliers. Therefore a second method was implemented which incorporates a further weighting based on the correlation score raised to a configurable exponent. This allows a system operator to adjust the system to situations with higher or lower SNR. Then the SciPy optimize package [21] is utilised again, this time using the minimize function with the Nelder-Mead solver type and the aforementioned weighting. This overall minimises the impact of outliers and places more importance on correlations that had a higher signal strength, allowing the system to take into consideration more samples with a lower risk of introducing poor data.

3.8 Visualisation

The visualisation module of the pipeline was implemented using the Python library Matplotlib[9] to display the estimated position. This was done in real time by plotting the intersection points and centroid calculated in the previous step.

Early in the project only three sensors were used, and therefore only three intersection points were plotted, and connected by lines to form a triangle similar to Figure 2.4b. The centroid of the triangle was displayed as a plausible estimate for the most likely point of origin. To signal an estimate as more recent the older plots changed their line style to progressively more dashed lines, providing an effect similar to fading out. The colour of the lines also changed to signify the strength of the correlation that caused the localisation event. The effect is displayed in Figure 3.9, where a sound source is seen moving left to right at around $y = 4$ and

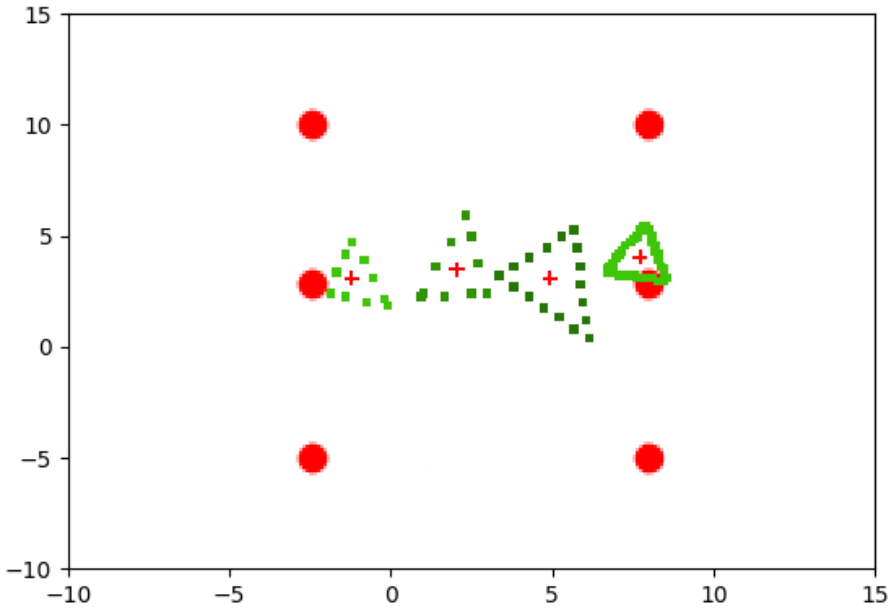


Figure 3.9: An illustration of the early visualisation method for a signal moving from left to right (not to scale)

producing enough signal to reach the positioning threshold 4 times.

When more sensors were introduced new visualisation methods were needed. Plotting the outline of all intersections was no longer as interesting, as many of the predictions are very close together. Instead each intersection used to estimate the signal's position was displayed as a circle with a low opacity, making it easier to see when many estimates were stacked on top of each other. Showing all the estimates used for the calculation makes it easier to see which devices might be poorly synchronised and causing an offset, as well as improved the understanding of how accurate the estimate was likely to be.

In Figure 3.10 the true position is marked with a blue star. The spread of the potential values becomes obvious, implying a relatively poor accuracy. However, the actual predicted position, marked with the red cross, is still less than a metre off.

In Figure 3.11 it is visually clear that the estimated position is quite likely to be the correct one as a very large majority of the samples fall within a very small area, with only a few outliers, and the predicted position is off by less than 5 centimetres. The blue star is not shown as it would almost completely overlap the red cross.

As mentioned in Section 3.7 most position estimates did not benefit from using the full set of intersections, and as such the examples shown in Figure 3.10 and 3.11 are mostly for demonstrational purposes, and in actuality the system usually displayed (and calculated) fewer intersection points.

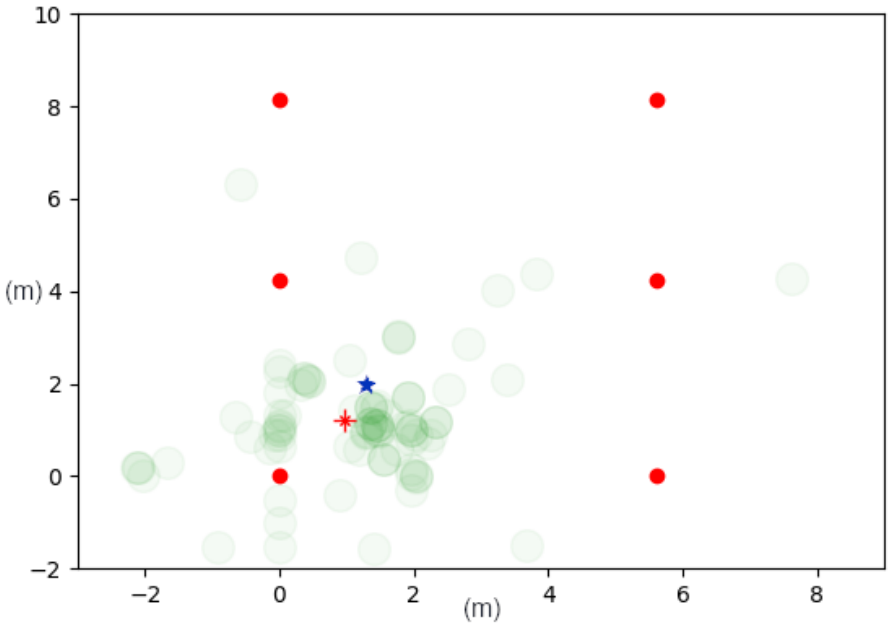


Figure 3.10: An example screenshot of the visualisation of a snap sound emitted at $(1.3, 2.0)$ with relatively poor sensor synchronisation

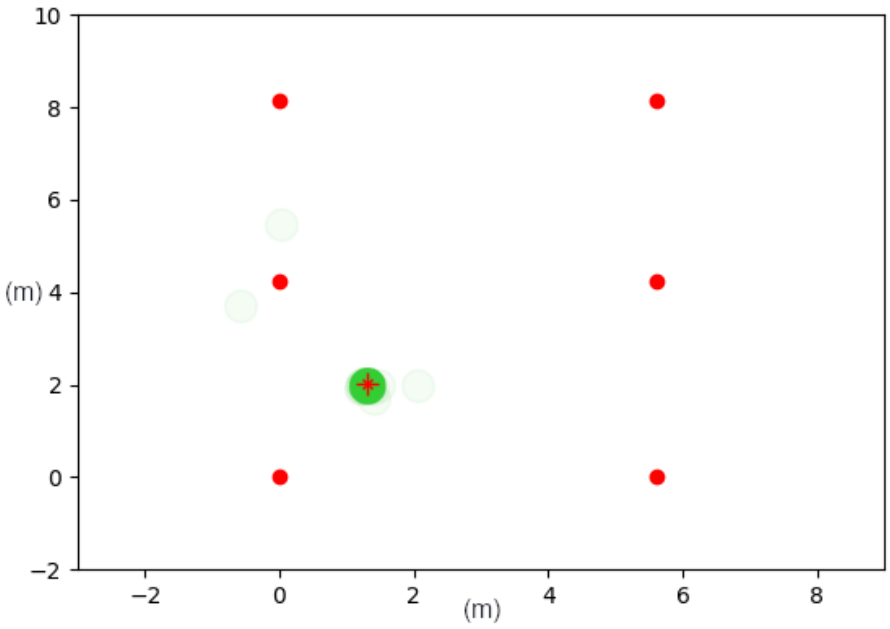


Figure 3.11: An example screenshot of the visualisation of a snap sound emitted at $(1.3, 2.0)$ with manual sensor synchronisation

3.9 Configuration

There are many variables and parameters that affect the performance of the system and these were collected and stored in a configuration file which the system then loaded the data from. This enabled more rapid testing of different configurations, as there was no need to search through the code to change the values. It also facilitated the automation of comparing the effectiveness of different values. The system used the following variables and parameters:

- Buffer size, how much data was included in each buffer
- Filter type(s) and cutoffs, to control which filters, if any, are applied
- Correlation method, to control which correlation method to choose
- Correlation threshold, to control the sensitivity of the system
- Sensor layout, to give the system information about which sensors it has access to and their positions
- Visualisation data, e.g. how many estimates to display at once
- Benchmarking data, to control options relevant for gathering quantifiable results

3.10 Benchmarking

To achieve quantifiable results of the system's performance a visual indication of the position is not enough. Instead a more objective measurement of the accuracy of a position estimate needed to be established. The first step in this process involved implementing a way to run through the same test data multiple times, saving the data that entered the system in a format that could be processed in the same way at a later time. This was accomplished by splitting the pipeline right before the data was sent to the correlating module, and saving the data to a WAVE-file, as well as by implementing a way to parse said files rather than read live data from the GStreamer pipeline.

Once the system could parse input files a module for the actual benchmarking process was implemented, where an audio file could be sent through the pipeline with a set of configuration files. Benchmarking files of different acoustic scenarios could then be run through the program once for each configuration file provided, to produce a file with the configuration settings used for a certain test run and its calculated positions. This data could later be analysed, compared to the ground truth, and presented to provide quantifiable results for the thesis. This process was also automated by a script used to parse the data and calculate some useful measurements such as the mean error.

Summary

This chapter has described the implementation methods used in this study. It described the hardware setup as well as the design of the software. Finally it

described how sound was produced to test the system and how the benchmarking was carried out. The upcoming chapter will describe what results this setup was able to achieve during testing.

Chapter 4
Results

In this part of the thesis the results of the implementation described in Chapter 3 are presented. First through an overview of the different ways in which results were gathered along with some visual examples, then with actual measurements of the system's accuracy and performance presented through tables.

4.1 Initial testing

The first experiments were conducted in an uncontrolled environment in an office area during the development of the solution, as seen in Figure 4.1. This provided an early indication of whether the implementation could position any sounds what so ever. The program produced some promising results in this setting by sometimes being able to very accurately determine where a sound came from. However, since the office area is uncontrolled, other sound sources could interfere with the testing and the layout of microphones was probably not optimal, with walls (marked grey in the figure) potentially causing reflections and resulting in faulty estimates. This resulted in a lot of false positives and false negatives in initial testing, meaning a sound was either discovered and positioned incorrectly or the intended sound was not positioned at all respectively. The testing did however give a decent approximation of whether the system worked at all or if some recent change had made an impact on its accuracy. When it could somewhat reliably position sounds a more controlled test session was performed.

4.2 Benchmarking session

A first attempt was made at gathering more structured and measurable data, but very varied results were achieved. As such a repeatable way to test the same files was necessitated as described in Section 3.10. Once this was implemented a new benchmark session was initiated.

Six devices were set up in a large approximately rectangular room and connected to the same router, as shown in Figure 4.2. The device clocks were synchronised with an implementation of ptpd2 to the leader device positioned at $(0, 0)$. The devices were laid down flat on tables of the same height and rotated so that the microphones all faced the centre of the area. The distances between the microphones of all devices were measured with a measuring tape, providing a reasonable

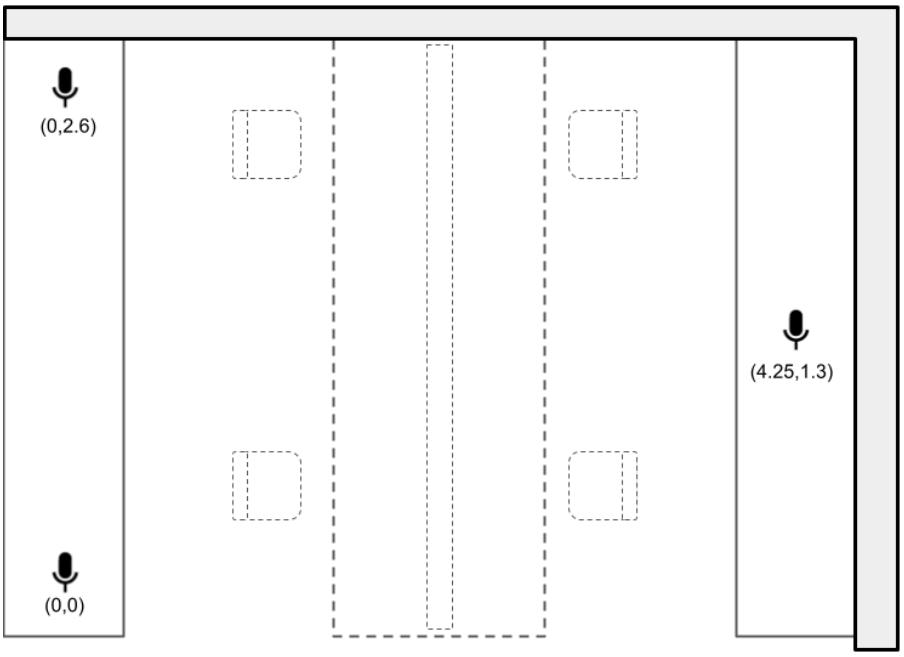


Figure 4.1: An overview of the initial testing setup

amount of accuracy where the positions should not be displaced by more than a couple centimetres.

The system was then started and configured to produce an output file of the audio that was recorded by all six devices. A test sequence of sounds was produced during each system run. Then the system was restarted and a new sequence of sounds was produced. All the data was recorded and saved, along with the approximate position(s) where sound was created.

The following test sequences were recorded:

- A noise profile where no signal was produced
- Repeated snaps at a location, two different locations
- Snaps along a straight path inside the sensor area(one vertical, one horizontal)
- Speech at consistent location, three different locations (one outside the sensor area)
- Speech along a straight path inside the sensor area (one vertical, one horizontal)
- A synchronisation control, with snaps very close to each device's microphone

With these files the system could be fine tuned to the different scenarios to provide an estimate of the best case accuracy, as well as tested with a setup that gives the best general performance. It was also at this point that the positioning

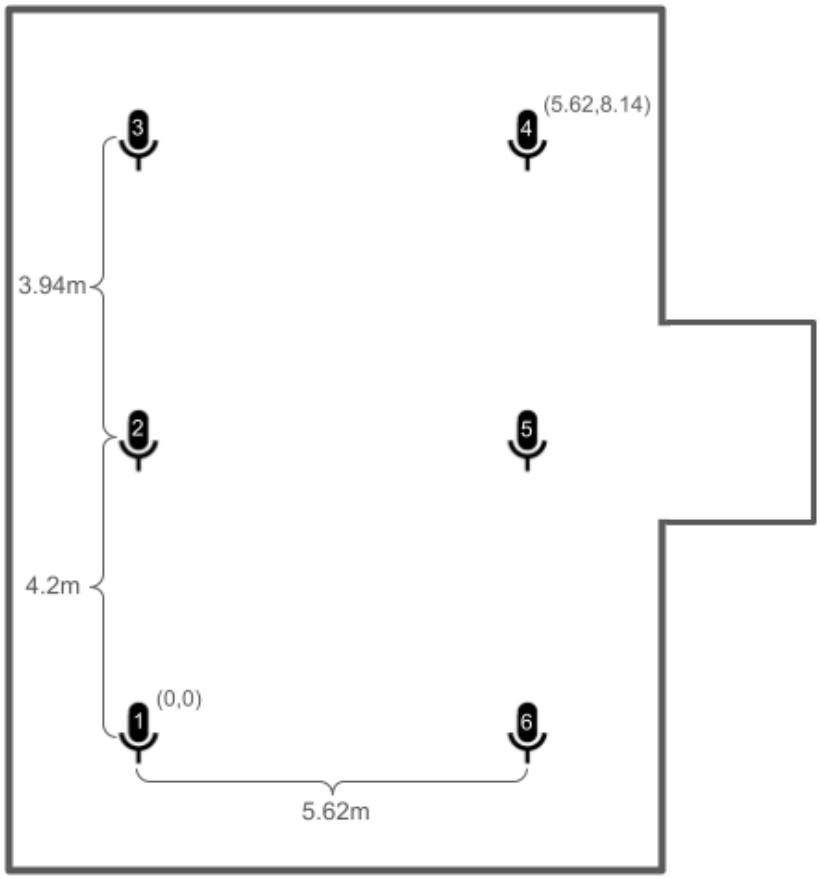


Figure 4.2: Benchmarking setup

was updated to reflect the usage of more than three sensors at a time, as previously the implementation only supported selecting exactly three of the sensors to use for the positioning.

4.3 Accuracy

Only some of the files produced were used to get quantifiable results. These were the files where the signal was repeatedly coming from the same location, as calculating the error of those positions was a lot simpler, rather than having to compare the estimated position at a certain time. The results for these files can be seen in Section 4.5.

The files with moving signals were also tested, but as the ground truth available for those positions was unreliably measured (by estimating the position while walking) any numerical error estimates would also be unreliable. Instead, they were used to produce Figure 4.3 which give a visual indication of the system's performance when estimating the position of a moving signal. As can be seen in Figure 4.3a and Figure 4.3b the tracking is decent for snap sounds. It is possible to determine which path the different sounds were produced along, even if the exact positions are slightly shifted. In contrast the tests with speech in Figure 4.3c and Figure 4.3d, the path is harder to distinguish. It is however possible to make a guess as to where the sounds came from due to their grouping.

For the data tables each file was run 5 times to mitigate the randomness factor of the noise injection, and the results of all 5 runs were collated and processed to produce 3 metrics.

First the Mean Absolute Error (MAE), formulated as the sum of the euclidean distances between each estimation and the ground truth divided by the number of estimates, represented mathematically as

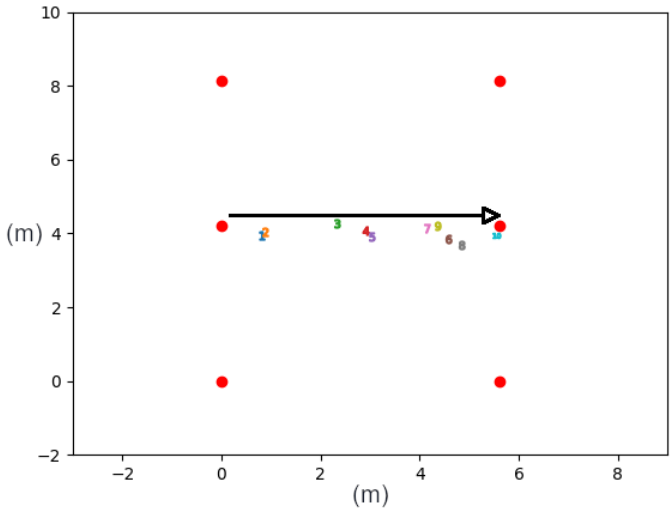
$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x-\hat{x}_i)^2+(y-\hat{y}_i)^2} \tag{4.1}$$

where

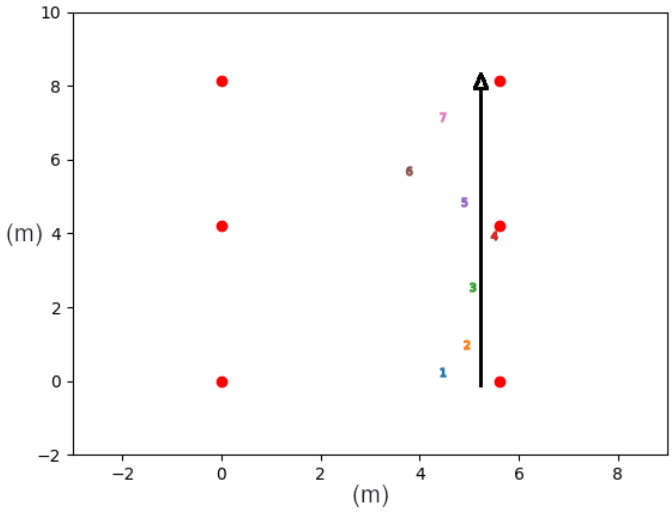
- MAE is the mean error of distance.
- n is the number of detected sound instances
- (x, y) are the actual coordinates where the sound was emitted.
- $(\hat{x}_i, \hat{y}_i)$ are the predicted coordinates for the i-th detected sound instance.

Similarly to the MAE the Mean Square Error (MSE) was also calculated, formulated as the sum of the squares of the euclidean distances between each estimation and the ground truth, divided by the number of estimates. It is represented with the same input data as the MAE with MSE as the mean squared error of distance as

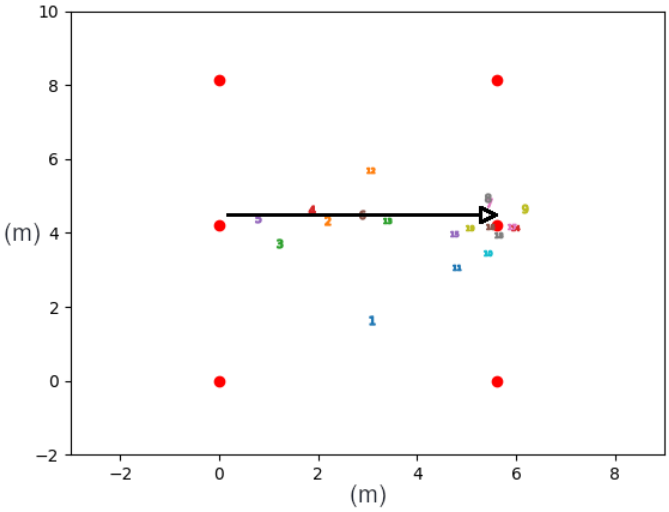
$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x-\hat{x}_i)^2+(y-\hat{y}_i)^2 \tag{4.2}$$



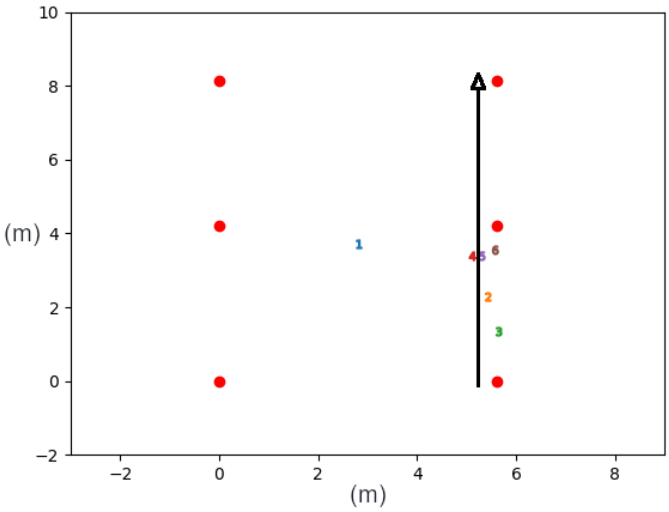
(a) Positioning snaps moving from left to right



(b) Positioning snaps moving from bottom to top



(c) Positioning speech moving from left to right



(d) Positioning speech moving from bottom to top

Figure 4.3: Example figures of the moving test signals

Because the synchronisation of the system is not perfect, many errors in positioning are actually not errors in the position estimation, but errors in the synchronisation, which affects the expected position. As such another interesting measure is the density or spread of the values, which gives an idea of how the system would perform without synchronisation errors. This will be referred to as the Mean Absolute Error with Bias Correction (MAEBC). The selected method of evaluating this value was to take the Mean Error as compared to the geometric average of the selected points rather than to the ground truth, represented mathematically as

$$\text{MAEBC} = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(\overline{x}-\hat{x}_i)^2+(\overline{y}-\hat{y}_i)^2} \tag{4.3}$$

where

- MAEBC is the mean error of distance from the estimates to the average point.
- n is the number of detected sound instances
- $(\overline{x}, \overline{y})$ are the average estimated coordinates.
- $(\hat{x}_i, \hat{y}_i)$ are the predicted coordinates for the i-th detected sound instance.

The MAEBC essentially becomes a measurement of the spread of the positions. A low value indicates that the predicted positions are clustered together, while a high value indicates that they are further apart. This in turn relates to the synchronisation error as the system's synchronisation was relatively stable during each run. This means the expected error in TDOA was also relatively stable.

It is important to note that while the MAEBC does give some indication of how much the error could be reduced by better synchronisation, it is not a direct measurement of that metric. The relationship between errors in signal arrival time and errors in predicted position is not linear, as one can easily deduce from e.g. Figure 2.4c and the related hyperbolic equations. Different magnitudes of desynchronisation will lead to different magnitudes of positioning error depending on the sign of the TDOA inaccuracy and the location the sound was emitted from. As such the MAEBC is mostly intended to give a vague estimate of how much better the system could perform under good synchronisation conditions.

4.3.1 Configuration

As explained in Section 3.3 the system was tested on two different classes of sound. These classes, roughly represented as sharp sounds and non-sharp sounds, made different demands of the positioning system, and as such performed better under different configurations. As such each input file was tested with three configurations, one optimised for detecting snaps, one optimised for detecting speech, and one intended to provide good results for both classes of sound. The differences between the configurations are shown in Table 4.1.

These configurations were arrived at by experimentation with the benchmarking files, and through seeing which features impacted which type of sound positively or negatively. For example snaps have a frequency content with a lot of high

Configuration	Buffer Size	Filter(Hz)	Min Corr.	Corr. Used
Snap	Small (8000)	180-14000	7	8
Generic	Medium (12000)	80-14000	5	12
Speech	Medium (12000)	80-14000	3	8

Table 4.1: Variation between the different configurations

frequencies, and could thus filter out more of the noise below 180 Hz without losing accuracy, while speech has a wider frequency content and sometimes performed poorly if all frequencies below 180 Hz were filtered out.

4.4 Tables of Results

The data for the benchmarking session is presented on the following pages in Tables 4.2-4.10. They contain the average error for each of the metrics described above, as well as 95% Confidence Interval (CI) margins and the associated upper bound (labelled as U.B. in the tables) for that Confidence Interval. These were calculated using the normal distribution calculations in the stats library of scipy[20]. The tests of snaps had an expected amount of events as each snap is a single sound and the system is expected to position each one. This also shows that some configurations for snaps found more than the expected amount of snaps, which means the system is vulnerable to false positives if an inappropriate configuration is used. It is more difficult to estimate an expected amount of positions for speech as it is a more continuous signal and not easy to separate into a discrete amount of events. As such the speech tests were simply labelled with the amount of events detected for each test, as well as a lower and upper bound for the amount of events detected in each run to show the variance introduced by the noise injection.

4.5 Synchronisation

The synchronisation of the devices varied between test sequences, and as such some performed much better than others. As can be seen in the difference in results in Table 4.2 and 4.5 the synchronisation and measurement error is responsible for $\geq 90\%$ of the error, and the Mean Absolute Error with Bias Correction of the snaps that were not manually synchronised is still rather large (even though it is considerably smaller than the corresponding Mean Absolute Error).

The manual synchronisation of the files whose errors are represented in Tables 4.2 -4.4 was achieved by calculating the sample offset that a snap emitted at $(1.3, 2.0)$ should have between sensors, and then manually shifting those audio streams so that each snap peak had the correct sample offset. Therefore the manual synchronisation actually corrects for both synchronisation errors and measurement errors in the positions of both the devices and the sound source, as the correct offsets are calculated assuming each device was positioned exactly where it was measured to be positioned.

Snap results, manually synchronised

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	0.09008	±0.0151	0.10518
MSE (m^2)	0.01133	±0.00434	0.01567
MAEBC (m)	0.08568	±0.0156	0.10128
Expected Events	Detected events	Discrepancy	Detection rate
55	55	0	1.0

Table 4.2: Snaps at $(1.3, 2.0)$, manually re-synchronised, with snap configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	0.1428	±0.0268	0.1696
MSE (m^2)	0.0305	±0.0099	0.0404
MAEBC (m)	0.1362	±0.0262	0.1624
Expected Events	Detected events	Discrepancy	Detection rate
55	55	0	1.0

Table 4.3: Snaps at $(1.3, 2.0)$, manually re-synchronised, with generic configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	1.1508	±0.2328	1.3836
MSE (m^2)	2.7071	±0.6546	3.3617
MAEBC (m)	1.1964	±0.0652	1.2616
Expected Events	Detected events	Discrepancy	Detection rate
55	99	+44	1.8

Table 4.4: Snaps at $(1.3, 2.0)$, manually re-synchronised, with speech configuration

Snap results, no manual synchronisation

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	1.0198	±0.103	1.1228
MSE (m^2)	1.1700	±0.221	1.391
MAEBC (m)	0.4301	±0.0584	0.4885
Expected Events	Detected Events	Discrepancy	Detection Rate
55	48	-7	0.8727

Table 4.5: Snaps at $(1.3, 2.0)$, no re-synchronisation, with snap configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	1.0522	±0.0925	1.1448
MSE (m^2)	1.2254	±0.1604	1.3857
MAEBC (m)	0.3826	±0.0714	0.454
Expected Events	Detected Events	Discrepancy	Detection Rate
55	54	-1	0.9818

Table 4.6: Snaps at $(1.3, 2.0)$, no re-synchronisation, with generic configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	1.0749	±0.1103	1.1851
MSE (m^2)	1.3484	±0.2734	1.6219
MAEBC (m)	0.7713	±0.1607	0.9321
Expected Events	Detected Events	Discrepancy	Detection Rate
55	62	+7	1.1273

Table 4.7: Snaps at $(1.3, 2.0)$, no re-synchronisation, with speech configuration

Speech results

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	0.8168	±0.0655	0.8823
MSE (m^2)	0.7431	±0.1285	0.8716
MAEBC (m)	0.3467	±0.0634	0.4101
Test Duration	Tests Run	Detected Events	Events per Run
22.08s	5	69	$(12 \leq n \leq 17)$

Table 4.8: Speech at $(4.32, 6.30)$, no re-synchronisation, with speech configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	0.9357	±0.0743	1.01
MSE (m^2)	0.9216	±0.1502	1.0718
MAEBC (m)	0.2549	±0.0486	0.3035
Test Duration	Tests Run	Detected Events	Events Per Run
22.08s	5	33	$(6 \leq n \leq 7)$

Table 4.9: Speech at $(4.32, 6.30)$, no re-synchronisation, with generic configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	0.91	±0.1468	1.0568
MSE (m^2)	0.9851	±0.3529	1.338
MAEBC (m)	0.445	±0.1171	0.5622
Test Duration	Tests Run	Detected Events	Events Per Run
22.08s	5	29	$(5 \leq n \leq 6)$

Table 4.10: Speech at $(4.32, 6.30)$, no re-synchronisation, with snap configuration

4.6 Performance

The system as a whole was created with the goal of achieving real-time positioning. That is, a live system in an acoustic environment should be able to position a sound and display it visually in a short enough time that the sound and the positioning of it seems coupled to an observer. Since the system is built to run indefinitely while the devices are online this also effectively formulates the requirement that "processing a given amount of data should take less time than producing that data". If this was not the case the system would lag further and further behind the current time, or would need to start dropping buffers in order to catch up.

The system implemented for the thesis passed this test on the hardware available with one important caveat, namely that there was not a constant signal. The system was constructed such that every buffer gets sent to the correlation module, but only buffers with a correlation above a certain threshold are treated as signal and sent to the positioning module, which was the most computationally expensive task. Because of this a buffer that appears to have a signal takes more time to process than a buffer without one. The sizes of the buffers can be varied, but must be at least a certain amount as shown in Equation 3.1.

Buffer size (samples)	Time Allotted (s/buffer)	Empty B. (s/buffer)	Signal B. (s/buffer)	Visualised B. (s/buffer)
4000	0.0418	0.0328	0.0377	0.2595
20000	0.2123	0.1771	0.2033	0.4059

Table 4.11: Performance results on a mid-range laptop

Some data was collected for various cases, as can be seen in Table 4.11. At the maximum resolution of 4000 samples per buffer per channel each buffer contains 42 ms of data and buffers without a signal took 33 ms to process, while each buffer with a signal took 260 ms. This means the system was real time applicable as long as only 4% of the buffers contained signal, and if more signal was present the system would not be able to keep up. At 20000 samples per channel the percentage of buffers containing signals that could be handled increased to 15.3%. The slow performance for buffers containing signals was mostly due to the visualisation module. All the positional data could be calculated and saved at any buffer size from 4000-20000 without exceeding the performance budget, but displaying it using the implemented methods (matplotlib) proved expensive.

This is currently an issue that could be easily mitigated with more powerful hardware, as the system was being run from a work station laptop. For scalability purposes, however, future implementations of similar systems could focus on achieving some form of distributed solution. One very simple solution would be to offload the actual visualisation to another computing unit by e.g. sending it to a web socket that then captures and displays the data. There is also the possibility of making part of the entire system, including the calculations, distributed across the actual devices used. This might be the most suitable solution as the system already uses a set of distributed devices with some amount of calculation capabilities, but it would also require some large changes in the current system.

Summary

The system seems to display a satisfactory ability to determine the location of origin of a sound, depending on the configuration. The data shows that the system is heavily impacted by errors in measurement and synchronisation. It also shows that the configuration impacts which sounds it is able to position well, as the speech configuration performed the worst when positioning snap sounds, mainly due to false positives, but performed the best when positioning speech. In the best case scenario of perfect synchronisation and minimal measurement errors the precision has a 95% Confidence Interval upper bound of less than 11cm.

With the hardware used visualisation was a potential bottleneck for the performance, and as such only a certain percentage of buffers could be visualised without the system falling behind. The calculations themselves were within the performance budget, and with more powerful hardware, or a more decoupled visualisation solution, the percentage of buffers that could be visualised would increase.

Chapter 5
Discussion

This chapter summarises the results of the thesis and discusses threats to the validity of those results. It also touches on real life use cases and some ethical and environmental considerations for this type of system.

5.1 Validity

The system as a whole performs the tasks it was designed for, but has relatively low general applicability. With six sensors set up in relatively close proximity the position of sounds could be somewhat reliably estimated given a decent synchronisation was achieved, and the results for the optimal case of positioning sharp sounds with perfect synchronisation were very promising at under 11 cm.

5.1.1 Development choices

Since the results with sharp sounds (snaps and clinking in glass/porcelain) proved much more reliable than other sounds (mainly speech and music) early on in the development process those sounds were more heavily used to estimate the accuracy during development as parameters were adjusted and feature sets changed. As such the system has always been more focused on sharp sounds, and alternative solutions that might have been better at estimating the TDOA of other types of signals may have been ruled out as the performance for sharp sounds was worse.

5.1.2 On the subject of noise

The system relied on noise injection as described in Section 3.5.1 in order to prevent most buffers from correlating based on the background noise of the sensors. This has both positive and negative effects regarding the validity of the results. Firstly, it is always possible that the quality of the results, both good and bad, are in some part based on the random seed used for the noise. This had to be accounted for by running each test several times to get an estimate of the variance. The random seed could be fixed to a certain value to reproduce the exact same values every run, but that would remove the one advantage that actually came from introducing noise, namely a more generalised result. As random noise is added the signal is slightly changed, meaning running the system multiple times produces slightly

different results, and thus the problem of overfitting [10] to the benchmarking data is partially mitigated. Because the input signal becomes slightly different the testing should also cover the case where an input signal starts out as slightly different, e.g. a snap that produces a slightly different sound. This should mean a better generic performance that won't risk leaning too heavily on a fine tuning of the parameters to the specific input files that were produced.

5.2 Improving system accuracy

The accuracy of the system was very dependent on the signal and surroundings. A more robust system could theoretically position more types of sounds to a higher amount of precision.

5.2.1 Synchronisation

While the devices within the system were synchronised with PTP[18] there is still a question of how well that synchronisation of the device clocks translates to the synchronisation of the timestamps that are sent with the captured audio signals. Since the devices are running an operating system stack which is scheduling many different tasks it is unclear how well the device clock synchronisation transfers to the actual synchronisation of the audio streams on the receiving end. Tests done on the system suggested that the synchronisation was consistent once the system started, but could have a static offset of a not insignificant amount of samples depending on when the devices first connected.

Once the system is up and running, it somewhat ensures stable continued synchronisation. Any packet loss from one of the sources is detectable, and if packets are not lost, the devices will each supply a constant signal stream at their specified sample rate. Since the system was tested on small local networks with physical connections there was minimal packet loss, which should mean that the relative synchronisation stayed consistent during each run of the system.

5.2.2 Adaptive filtering

The system only implemented a simple bandpass filter in the end, but much more advanced filtering techniques exist. An adaptive filter could extract a noise profile of the environment to more specifically target certain frequencies, and improve the signal-to-noise ratio further.

Adaptive filtering could be implemented to modify the filter parameters automatically over time based on the current properties of the signal or to base the filter parameters on data gathered from the environment in a calibration sequence. Both of these alternatives could lead to better system performance in the average case, but excessive filtering could also cause signal degradation.

5.2.3 Calculating reflections

If the system is given more information about its surroundings it's possible to calculate possible reflection vectors and thereby eliminate some errors. This would

require any implementation of the system to also include information about where any walls or similar obstacles are present, but should give the system fewer false positives where sounds are erroneously positioned partially based on reflected sound instead of sound that's taken the shortest path.

This could also give the system a more precise metric for determining when a position is unreasonable, as it currently only bases it on the area spanned by the sensors. If there are walls or similar obstacles within that area the current system can still suggest that a sound came from inside a wall or other unreasonable positions. This data could however be difficult to gather as it would require a lot of manual measurements.

5.3 Real world applications

With a working implementation of the suggested system there is an array of different use cases. One example is monitoring of a facility that already has a setup of network connected devices with built-in microphones. This means an operator or somebody working in the area can "examine" the area for any disturbances. It is also possible that this could be used in symbiosis with a setup of surveillance cameras to be able to aim a camera at a point of interest discovered based on the estimated location of the sound source. This would be similar to the suggested system used by Farzone and Smidje[6]. This could be used to discover events that might not be registered by for example a directional camera, since a microphone might have a less directional coverage area. This could reduce the amount of cameras needed to surveil an area.

Knowing the position of a person could also open possibilities to implement beamforming and similar techniques to play sounds aimed at a specific position for better sound accuracy, or to offset audio recordings from the environment to synchronise the signal at each receiver, thereby achieving a better signal-to-noise ratio in the recorded audio.

5.4 Portability

The system was constructed to be relatively easy to set up. As long as there are RTP packets with interleaved signals from sensors with known locations and synchronised clocks being sent to a device that can run Python it should be possible to set up the system for any type of signal through the configuration file. Setting up the Python environment itself is also a simple task as a virtual environment was utilised, limiting potential compatibility issues.

This means that the system can be implemented using practically any hardware capable of recording a signal and transmitting it as long as there is some means of synchronisation across the devices and a central computer powerful enough to run the calculations.

5.5 Future work

The system has many possible improvement areas, some of which are mentioned in Section 5.2. However, there are other areas than positioning accuracy that can be improved upon or modified with an application of further theory outside the scope of this thesis.

5.5.1 Sound classification

While there are advantages to the approach of positioning every sound there are also disadvantages. Positioning is a computationally intensive task and for scalability purposes it would be better to position only the sounds that are important. What is important will vary from system to system, but implementing a classification module to distinguish between important and non-important signals could be more system agnostic. A solution based on machine learning such as that seen in previous work at Axis by Chan & Karlsson[3] is a viable idea, although their implementation is also computationally expensive and has a narrow scope, only classifying whether something is or is not a gunshot.

A classifier could theoretically be trained to identify many more sounds, opening the door to many possibilities for data gathering. Similar techniques are being used today in monitoring animal populations and other ecological factors[2], reducing the manual workload and increasing the accuracy of population counts.

One could imagine similar systems in place in all sorts of disciplines. Tracking sneezes or coughing in hospitals to predict hot spots for infections. Monitoring traffic flow in busy intersections to gather congestion data and improve future road design. Quickly finding people who are hurt or yelling in large masses of people, such as during a concert or in a theme park.

All these methods rely on a robust classification module that needs to be trained with preexisting data sets, something which was not within the scope of this thesis, but which could easily be hooked into the system as an extension of the filtering module.

5.5.2 Better presentation of data

The system currently presents data in a human readable fashion, giving an idea of both when the event happened, where the most likely location is, and an approximation of the correlation strength i.e. the certainty of that location. However, there are still ways to communicate these factors, and perhaps other factors, more clearly. A future implementation could present more data about each sound. Coupled with a classification module the user could choose to see only specific groups of sounds, or to display sounds with an icon representative of what the sound was classified as.

The most important feature to add if the system is to be used in security or surveillance would be some sort of database and timeline system. Currently the system has no way to check old data other than manually interpreting text files or by rerunning an entire file of audio data and observing it. A timeline feature where the user could scroll through and replay old positioning data would greatly help a

human operator in analysing the data should the system ever be made available as part of a product or solution.

5.5.3 Scaling to more devices and multi-microphone devices

The tested implementation scales rather poorly to a larger quantity of device for a few reasons. Since all the calculations are done on a single computer in the setup used in the experiments one might end up with a much higher load of computations which might not be feasible. It might be possible to scale further than the current performance by simply having a CPU with more cores and an implementation that handles threads in a more performant way, but this could potentially become expensive. A more distributed system where the workload is shared among the device in the network would be preferable for scalability, since the increased load from more calculations is not placed on a single device. This does however pose some additional difficulties when it comes to distributing the workloads, especially if the devices are also performing other tasks.

Introducing more sensors also further exacerbates the problem of sensor selection. Deciding which sensors to use for calculations is not a trivial problem to solve in an optimal way, and calculating every single correlation is a costly solution. Sensors far from the sound source might result in worse measurements and can decrease accuracy. However more sensors could potentially result in better results. This is something that could be studied further and would probably be necessary for real world applications in larger systems.

5.6 Ethical considerations

When implementing any type of surveillance system it is important to consider the ethical impacts. As the system was developed at a company with a security and surveillance profile and has explicit compatibility with their products it can be assumed that any ethical considerations would be made by the company according to their normal procedure before the system was implemented in any product released to the public. Such considerations have been made for other products, and there are already privacy features in place such as dynamic masking to prevent personally identifying information in video footage.

That being said the system in this thesis could well be extended to allow for a surveillance system that could collect data that would be considered an invasion of privacy. Tracking of objects can be used in many different ways, and the same underlying techniques that allows the system to position sounds have long been used in military applications for both defensive and offensive purposes. The system has no inherent ties to any personal data, but it could possibly be connected to something like a voice recognition system to further the capabilities of a large scale surveillance system. However, that would represent quite a small part of such a system, and the position of objects to such a high degree would be quite a small part of the important data.

With enough positional data from enough sources it is possible to extract patterns that are otherwise impossible to see in how people or objects move. This data could be used to predict future movement patterns to improve overall mobility

or for more ethically questionable activities, like advertising more in areas with a lot of movement. But in general this type of use of the technology would be cumbersome, and other types of technology, such as video analysis, would probably be simpler.

In conclusion it is the opinion of the thesis writers that the technology has use cases within both ethical and non-ethical scenarios, but that the non-ethical scenarios would require a much more advanced implementation to the point where the system presented in this thesis becomes a very minor part of the overall system.

5.6.1 Environmental impact

Another ethical concern is the environmental impact of the system. It is a fact that producing data with the system means spending energy to perform the calculations. With tighter demands on sensor selection and a more distributed workload the impact could be reduced, but never removed. Any use of this system will simply have to take the processing power into account, and perhaps change variables such as the system's uptime to reduce the energy usage.

On the other hand a system such as this could potentially use less equipment than a video surveillance system performing similar tasks. A microphone array streaming data and only storing positional info could have a much lower power usage and lower storage demands, and might use much less materials to produce. However the specific scenarios where positioning data is useful on its own are probably quite rare. Instead enhancing a current system with positioning capabilities seems more likely, where there could be a need for fewer cameras if they are able to, for example, rotate to observe sound events as implemented by Mazdak & Smidje[6].

5.7 Summary

The research questions established in Section 1.2 can now be answered with the background and results of the thesis.

R1. How can position tracking of a noise source be implemented using networked devices with built-in microphones and known positions?

It can be implemented by estimating the Time Difference of Arrival (TDOA) of a signal arriving at different sensors through the use of the correlation peak of the signals, as explained in Section 3.6. These TDOA values can then be combined as described in Section 3.7 to estimate the position.

R2. What is such a system's performance and accuracy limits when dealing with different types of sound sources?

On a standard consumer grade laptop the system was able to run continuously and save positioning data no matter how much signal was present. It was, however, only able to display a certain amount of data to the screen without falling behind, as explained in Section 4.6. This problem could be solved by either investing in

stronger hardware, or implementing a more distributed approach which off-loads the visualisation to another processing unit, or at least a different thread.

The accuracy varied for different types of sound, as can be seen in the tables in Section 4.5. As a general rule, with decent synchronisation around a 1 m accuracy could be achieved, while sharper sounds and optimal synchronisation increased the accuracy in a best case scenario to around 10 cm as seen in Table 4.2.

R3. How can the positional data be visualised?

There are many possible ways of displaying the data, and matplotlib[9] is one viable option to display data in several different ways, as shown in Section 3.8. It did, however, use a lot of the performance budget, as can be seen in Table 4.11, due to not being optimised for showing updated data, and having to redraw certain data points each frame.

R4. Is real-time analysis and visualisation of this data viable on standard hardware?

Based on the results achieved it is a very viable approach with ≤ 6 units. The system itself should be able to be expanded to include more units with no major issues as long as some sort of distributed method is adopted, especially for the visualisation.

R5. What are the real world applications for this data?

There are several ways to use this data, e.g. to alert a security operator or to improve the quality of recorded audio as described in Section 5.3 on the real world applications. Many more use cases would open up if the system were to be expanded upon as described in Section 5.5 on future work, e.g. by implementing sound classification or by scaling the system to a much larger amount of devices.

References

[1] IEEE Recommended Practice for Speech Quality Measurements. *IEEE No 297-1969*, pages 1–24, 1969. doi:10.1109/IEEESTD.1969.7405210.

[2] Daniel T. Blumstein, Daniel J. Mennill, Patrick Clemins, Lewis Girod, Kung Yao, Gail Patricelli, Jill L. Deppe, Alan H. Krakauer, Christopher Clark, Kathryn A. Cortopassi, Sean F. Hanser, Brenda McCowan, Andreas M. Ali, and Alexander N. G. Kirschel. Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations and prospectus. *Journal of Applied Ecology*, 48(3):758–767, 2011. URL: https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2664.2011.01993.x, doi:10.1111/j.1365-2664.2011.01993.x.

[3] Martin Chan and Sofie Karlsson. Detection and localisation of gunshots using sound data. Master's thesis, Lund University, faculty of mathematics, 2018. https://www.lu.se/lup/publication/8953047.

[4] *Computational Acoustics.* CISM International Centre for Mechanical Sciences, Courses and Lectures: 579. Springer International Publishing, 2018.

[5] Introduction to cross correlation. [Accessed: 2024-02-12]. URL: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/cross-correlation.

[6] Mazdak Farzone and Kim Smidje. Embedded sound localization using multilateration in network camera system. Master's thesis, Lund University, faculty of engineering, 2013. https://www.eit.lth.se/index.php?gpuid=288&eauid=682&L=1.

[7] GStreamer documentation. [Accessed: 2024-04-23]. URL: https://gstreamer.freedesktop.org/documentation/?gi-language=c.

[8] Cyril M. Harris. Absorption of sound in air in the audio-frequency range. *Journal of the Acoustical Society of America*, 35(1):11 – 17, 1963.

[9] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi:10.1109/MCSE.2007.55.

[10] IBM. Overfitting, 2024. [Online; accessed April 18, 2024]. URL: https://www.ibm.com/topics/overfitting.

[11] Hong-Goo Kang, Michael Graczyk, and Jan Skoglund. On pre-filtering strategies for the gcc-phat algorithm. In *2016 IEEE International Workshop on Acoustic Signal Enhancement (IWAENC)*, pages 1–5, 2016. doi:10.1109/IWAENC.2016.7602964.

[12] C. Knapp and G. Carter. The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(4):320–327, 1976. doi:10.1109/TASSP.1976.1162830.

[13] Andrew J. Kolarik, Silvia Cirstea, Shahina Pardhan, and Brian C. J. Moore. A summary of research investigating echolocation abilities of blind and sighted humans. August 2017. doi:10.1016/j.heares.2014.01.010.

[14] II Mellen, G., M. Pachter, and J. Raquet. Closed-form solution for determining emitter location using time difference of arrival measurements. *IEEE Transactions on Aerospace and Electronic Systems, Aerospace and Electronic Systems*, 39(3):1056 – 1058, 2003.

[15] NTP v4 specification. [Accessed: 2023-11-13]. URL: https://www.ntp.org/reflib/rfc/rfc5905.txt.

[16] Numpy 1.26 documentation. [Accessed: 2024-03-22]. URL: https://numpy.org/doc/1.26/.

[17] Panos Photinos. *The Physics of Sound Waves (Second Edition)*. 2053-2563. IOP Publishing, 2021. URL: https://dx.doi.org/10.1088/978-0-7503-3539-3, doi:10.1088/978-0-7503-3539-3.

[18] Ieee standard for a precision clock synchronization protocol for networked measurement and control systems. [Accessed: 2024-03-22]. URL: https://standards.ieee.org/ieee/1588/4355/.

[19] ptpd2 - precision time protocol daemon (1588-2008). [Accessed: 2024-03-22]. URL: https://man.freebsd.org/cgi/man.cgi?query=ptpd2&sektion=8&manpath=freebsd-ports.

[20] scipy 1.11.3 documentation. [Accessed: 2024-03-22]. URL: https://docs.scipy.org/doc/scipy-1.11.3/.

[21] scipy 1.11.3 optimize package documentation. [Accessed: 2024-03-22]. URL: https://docs.scipy.org/doc/scipy-1.11.3/reference/optimize.html#module-scipy.optimize.

[22] scipy 1.11.3 signal package documentation. [Accessed: 2024-03-22]. URL: https://docs.scipy.org/doc/scipy-1.11.3/reference/signal.html.

[23] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

Appendix A

Additional tables

Further tables of results are presented below as the gathering of this data was described in Section 4.2. They were placed in the appendix due to the generally worse results for these files, stemming mainly from a lack of synchronisation, as none of these files have been manually synchronised. This made the results less representative of the system's potential performance under good synchronisation conditions, and made it more difficult to compare results between tests.

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	1.5403	± 0.2181	1.7584
MSE (m^2)	3.0537	± 0.7529	3.8066
MAEBC (m)	1.0677	± 0.1766	1.2443
Expected Events	Detected Events	Discrepancy	Detection Rate
55	56	+1 (+1)	1.0182

Table A.1: Snap at $(4.32, 6.30)$, with snap configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	1.9712	± 0.1316	2.1027
MSE (m^2)	4.1287	± 0.4966	4.6254
MAEBC (m)	0.617	± 0.0782	0.6951
Expected Events	Detected Events	Discrepancy	Detection Rate
55	55	0 (±0)	1.0

Table A.2: Snap at $(4.32, 6.30)$, with generic configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	2.6768	± 0.2147	2.8915
MSE (m^2)	7.8495	± 0.9814	8.8309
MAEBC (m)	1.3862	± 0.2571	1.6433
Expected Events	Detected Events	Discrepancy	Detection Rate
55	58	+3 (+1)	1.0545

Table A.3: Snap at $(4.32, 6.30)$, with speech configuration

Test Duration	Tests Run	Detected Events	Events Per Run
16.32s	5	0	0

Table A.4: Speech at $(1.3, 2.0)$, with snap configuration.
No events were detected with this configuration

Test Duration	Tests Run	Detected Events	Events Per Run
16.32s	5	0	0

Table A.5: Speech at $(1.30, 2.0)$, with generic configuration.
No events were detected with this configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	1.9492	± 0.344	2.2932
MSE (m^2)	4.8161	± 1.761	6.577
MAEBC (m)	1.6928	± 0.3148	2.0075
Test Duration	Tests Run	Detected Events	Events Per Run
16.32s	5	147	$(28 \leq n \leq 31)$

Table A.6: Speech at $(1.30, 2.0)$, with speech configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	2.6925	± 0.4171	3.1096
MSE (m^2)	8.5631	± 2.5989	11.1619
MAEBC (m)	1.4303	± 0.255	1.6853
Expected Events	Detected Events	Discrepancy	Detection Rate
35	30	-5 (−1)	0.8571

Table A.7: Snap at $(2.80, -1.0)$, outside of sensor area, with snap configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	2.9641	± 0.4364	3.4004
MSE (m^2)	9.5292	± 2.3699	11.8991
MAEBC (m)	0.9942	± 0.2093	1.2036
Expected Events	Detected Events	Discrepancy	Detection Rate
35	16	-18 (−5 : −3)	0.4571

Table A.8: Snap at $(2.80, -1.0)$, outside of sensor area, with generic configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	2.2446	± 0.3237	2.5683
MSE (m^2)	5.8566	± 1.5609	7.4175
MAEBC (m)	1.8821	± 0.363	2.2451
Expected Events	Detected Events	Discrepancy	Detection Rate
35	31	-4 (−1)	0.8857

Table A.9: Snap at $(2.80, -1.0)$, outside of sensor area, with speech configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	2.6552	± 0.2233	2.8785
MSE (m^2)	7.7771	± 1.3017	9.0788
MAEBC (m)	1.316	± 0.2489	1.5649
Test Duration	Tests Run	Detected Events	Events Per Run
20.0s	5	57	$(9 \leq n \leq 14)$

Table A.10: Speech at $(2.80, -1.0)$, outside of sensor area, with snap configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	3.1627	± 0.1355	3.2982
MSE (m^2)	10.3516	± 0.8818	11.2334
MAEBC (m)	0.8919	± 0.0986	0.9905
Test Duration	Tests Run	Detected Events	Events Per Run
20.0s	5	74	$(14 \leq n \leq 15)$

Table A.11: Speech at $(2.80, -1.0)$, outside of sensor area, with generic configuration

Error Metric	Estimate	95% CI Margin	95% CI U.B.
MAE (m)	2.991	± 0.1396	3.1307
MSE (m^2)	9.6874	± 0.8905	10.5779
MAEBC (m)	1.8687	± 0.1646	2.0333
Test Duration	Tests Run	Detected Events	Events Per Run
20.0s	5	147	$(28 \leq n \leq 31)$

Table A.12: Speech at $(2.80, -1.0)$, outside of sensor area, with speech configuration

Printed by Tryckeriet i E-huset, Lund 2024

LUND
UNIVERSITY

Series of Master´s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2024-976
http://www.eit.lth.se