# A self-calibrating system for finger tracking using sound waves

Master's thesis

by

Linus Hammarlund

September 7, 2017

# About the report

| | |
|---|---|
| **Title** | A self-calibrating system for finger tracking using sound waves |
| **Date** | September 7, 2017 |
| **Author** | Linus Hammarlund<br>`linus.hammarlund@gmail.com` |
| **Supervisor** | Kalle Åström<br>`kalle@maths.lth.se`<br>Centre for Mathematical Sciences,<br>Faulty of Engineering,<br>Lund University |
| **Co-supervisor** | Mikael Swartling<br>`mikael.swartling@eit.lth.se`<br>Department of Electrical and Information Technology,<br>Faulty of Engineering,<br>Lund University |

# Abstract

In this thesis a system for tracking the fingers of a user using sound waves is developed. The proposed solution is to attach a small speaker to each finger and then have a number of microphones placed ad hoc around a computer monitor listening to the speakers. The system should then be able to track the positions of the fingers so that the coordinates can be mapped to the computer monitor and be used for human-computer interfacing. The thesis focuses on the proof-of-concept of the system. The system pipeline consists of three parts: signal processing, system self-calibration and real-time sound source tracking.

In the signal processing step four different signal methods are constructed and evaluated. It is shown that multiple signals can be used in parallel. The signal method with the best performance uses a number of dampened sine waves stacked on top of each other, with each sound wave having a different frequency within a specified frequency band. The goal was to use ultrasound frequency bands for the system but experimenting showed that they gave rise to a lot of aliasing, thus rendering the higher frequency bands unusable.

The second step, the system self-calibration, aims to do a scene reconstruction to find the positions of the microphones and the sound source path using only the received signal transmissions. First the time-difference of arrival (TDOA) values are estimated using robust techniques centred around a GCC-PHAT. The time offsets are then estimated in order to convert the TDOA problem into a time-of-arrival (TOA) problem so that the positions of the receivers and sound events can be calculated. Finally a "virtual screen" is fitted to the sound source path to be used for coordinate projection.

The scene reconstruction was successful in 80 % of the test cases, in the sense that it managed to estimate the spatial positions at all. The estimates for the microphones had errors of $11.8 \pm 5$ centimetres on average for the successful test cases, which is worse than the results presented in previous research. However, the best test case outperformed the results of another paper. The newly developed and implemented technique for finding the virtual screen was far from robust and only found a reasonable virtual screen in 12.5 % of the test cases.

In the third step the sound events were estimated, one sound event at a time, using the SRP-PHAT method with the CFRC improvement. Unfortunate choices of the search volumes made the calculations very computationally heavy. The results were comparable to those of the system self-calibration when using the same data and the estimated microphone positions.

# Acknowledgements

I would like to thank my supervisor Kalle Åström for all the input, help and encouraging discussions during the work with this thesis. You are a source of inspiration! I would also like to thank Mikael Swartling and Nedelko Grbic from the department of Electrical and Information Technology for all the help with the signal processing and the hands-on experimenting with sound waves. The completion this thesis stretched over a long period of time with a one-year break in the middle. Thank you, Maja, for the support during all this time and for being an awesome person.

This thesis marks the end of my time at LTH. Thus I would also like to give my sincerest thanks to all the student organisations welcoming me with open arms. Lund, the university and LTH would not be what it is today without the student life and all the time and effort people commit. Thank you F-sektionen, Spegatspexet and Teknologkåren for the time we shared together.

Finally a huge thanks to all my friends and family!

# Table of Contents

Chapter 1

# Introduction

*In the first chapter a background to the problem is given, as well as a proper problem formulation with goals and limitations.*

## 1.1 Motivation

The ability to control a computer remotely using hand gestures have been a concept explored by the sci-fi genre for a long time. One example is the movie *The Minority Report* [1] where the user interface of a computer system is controlled by a detective using a set of gloves in combination with gesture tracking. A similar setup exists in the movie *Avatar* [2]. The systems leave the moviegoers with a certain wow-factor. Thus the idea for this thesis was born.

Systems like these offer an alternative way for users to interact with computers compare to classical mouse/keyboard setups. The term coined for this is *human-computer interaction* (HCI). Sometimes they may offer an easier and more intuitive way, whilst in other situations the experience might be worse. There are environments where physical interactions with a computer or a computer monitor might be undesirable, such as sterilized or dirty environments, where a remote interaction is preferable. Another example where a mouse/keyboard setup will not do much good is when the users want to communicate by sign language, an application where hand and finger gesture tracking will do the trick.

Techniques to accomplish this already exists in research and on the market today. One way is to track the user's hands using a camera by mapping the gestures of the hand with image analysis and computer vision algorithms. This is effective and a lot of resources are put into research of the subject. However there are situations where filming the user might lead to a lot of problems concerning personal privacy. Therefore it is interesting to explore alternative solutions.

## 1.2 Related work

Using camera imaging to solve the problem of finger tracking is broad field of study. In Hongyong and Youling [3], finger tracking and gesture recognition is accomplished using a Microsoft Kinect camera. The depth information that the camera provides is used to segment the human from the video stream and a K-nearest neighbours algorithm is used for object tracking.

In Ma et al. [4], visual light patterns are projected on a surface. Together with light sensors attached to both the camera device and the fingers of the user, a real-time tracking of the fingers is accomplished. Popa et al. [5] offers a way to track fingers based on quasi contiguous clusters of line strips using a common web camera. In Shaker and Zliekha [6] a dual-camera setup is used to enable finger tracking. No 3D model is constructed and the positions of the index fingers are estimated directly from the line intersections and image processing.

Doing sound source localization using ultrasound techniques is an area far less researched. No papers on finger- and gesture tracking have been found, however systems to track other objects exist and a few examples are presented here. In Ens et al. [7] and Ens et al. [8] an ultrasound communication system using time-difference of arrival is developed. Fixed transmitters are locating a mobile receiver and a self-calibration is used to lower installation costs of the system. In Matsumoto et al. [9], human body parts are tracked using distance measurements from sound waves. The system is implemented using wearable devices.

In Zhayida et al. [10] a self-calibration system for locating sound sources and a moving sound source is done using estimated time-difference of arrival values. The sound waves used in the experiments are an actual pop song from radio. The techniques used in this thesis are presented as the system design unfold in chapter 3.

## 1.3 Problem statement

### 1.3.1 Goals

The goal of this master's thesis is to build a system for interaction between a user and a computer using sound transmissions. The system should

- be easy to use for ordinary people,
- be easy to setup at the user's workplace or home,
- not interfere with the user's workflow in a significant way,
- be able to track the user's fingers in real-time,
- convert the finger movements to coordinate paths on a computer monitor, and
- be configurable to the extent that the user can choose how much the fingers need to be moved to create a comfortable user experience.

To achieve these goals the problem needs to be divided into smaller parts. The sub-goals can be categorized into the following groups:

- **Usability and ease of setup.** The system needs to
  - assume no fixed positions of any components,
  - have a simple input/output system,
  - allow the user to calibrate the system,
  - feature small components, and
  - require no knowledge about sound waves on the user's part.
- **Localization.** The system needs to
  - be able to locate the fingers,
  - be able to locate the microphones, and
  - be able to convert the spatial positions of the fingers to coordinates on a computer screen.
- **Signal processing.** The system needs to
  - process and generate sound in real-time, and
  - transmit sounds that do not disturb the user.

### 1.3.2 Proposed solution

Attach a small sound source to each finger of a user. An easy way to do this would be to integrate the sound sources into some sort of a glove. The user would also place a number of microphones around his or her workstation, either in direct contact with the computer monitor or spread all over the desk. Let the sound sources transmit sound waves outside the audible frequency spectrum as the user moves his or her fingers around. By recording the sound waves the system will reconstruct the movement of the fingers in real-time. The 3D coordinates of the fingers can then be projected onto the computer monitor in order to steer the mouse.

### 1.3.3 Scope of the project

An implementation of a system like this is a very large project and unfortunately outside the scope of this master's thesis. Things that fall outside the scope are:

- Whilst a real-time system is a big goal, this thesis is more of a proof-of-concept. The implementation will be done in MATLAB which slows things down by a considerable amount. To produce a real-time system one would have to implement it in a faster programming language. However, the techniques used will take into account that they will be run for small discrete time steps. Also, the speed of the techniques will be discussed.
- Gesture tracking (i.e. clicking, swiping and zooming) could also be obtained from the paths of the finger movement by using machine learning techniques. This thesis will not feature gesture tracking.
- A customized glove with attached sound sources will not be produced. Instead two hand-held speakers will be used as a proof-of-concept.
- Mouse steering will not be implemented, nor an actual a projection of coordinates as they arrive. A plane will be estimated onto which it is trivial to project and convert coordinates in a real application.
- No experiments with five fingers will be conducted. To exemplify that more than one sound source can be used, some of experiments will be run with two sound sources instead of one.

## 1.4 Outline of the report

The report is divided into the following chapters:

1. **Introduction.** A background to the problem is given, as well as a proper problem formulation with goals and limitations.
2. **Basics and conventions.** The common terminology and conventions are presented. The underlying acoustic model is defined and the fundamental localization problems are introduced.
3. **System design.** The design of the system pipeline is described. Arguments are made to defend why certain choices have been made and why other choices have been tossed aside.
4. **Techniques and theory.** Descriptions of all the algorithms, derivation of formulas and mathematics used in the system implementation are presented.
5. **System implementation.** The system design, theory and techniques from the previous chapters are put together into an actual implementation.
6. **Experiments.** In order to evaluate the system a number of experiments have to be conducted. The process of setting up these experiments and what limitations they give rise to are presented in this chapter.
7. **Results.** The results as well as some evaluation of said results are presented.
8. **Analysis.** The analysis of the results and evaluations are presented. The chapter is structured so that the sections from the previous chapter maps to the same sections in this chapter.
9. **Conclusions and future work.** The work presented in thesis is summarized. The main findings from the results, evaluations and analysis are presented briefly. Also some thoughts about what angles to pursue next are put into words.

Chapter 2

# Basics and conventions

*In this chapter the common terminology and conventions are presented. The underlying acoustic model is defined and the fundamental localization problems are introduced.*

## 2.1 Terminology and conventions

In this section some terminology is introduced that will be used throughout this thesis. First a number of concepts are introduced, followed by lists of common abbreviations, notations and mathematical operations.

### 2.1.1 List of concepts

| | |
|---|---|
| Sound source | A **sound source** is an entity that generates some sort of sound. It can be anything from humans, birds and loudspeakers to squeaks from doors and high-pitched noises from electronic devices. All sound sources generate sound over varying periods of time, be it short bursts of energy or long continuous signals. For each point in time (assuming the time periods can be discritized), a sound source generates a **sound event**. |
| Sound event | A **sound event** is an emission of a **sound source** at a specific point in time (assuming that the time can be discretized). |
| Receiver | A **receiver** is a sensor that registers sound events, i.e. a microphone or an eardrum. |
| Output device | See **sound source**. |
| Microphone | See **receiver**. |
| Transmitter | See **sound source**. |
| Sound recording | In this thesis a **sound recording** refers to a recording of one or more **sound sources** recorded by one or more **receivers**. A sound recording has $M$ channels, one for each receiver. |
| Channel | A **channel** refers to the signal recorded by a specific **receiver** in a **sound recording**. |
| Inlier | A “good” value. Complement of the **outliers**. See section 4.4. |
| Outlier | A “bad” value. Complement of the **inliers**. See section 4.4. |

### 2.1.2 List of common abbreviations

| | |
|---|---|
| DOA | Direction of arrival. See section 2.2.2. |
| TDOA | Time-difference of arrival. See section 2.3. |
| TOA | Time-of-arrival. See section 2.4. |
| GCC-PHAT | Generalized cross-correlation with the Phase Transform. See section 4.1. |
| CFRC | Coarse-to-fine region contraction. See section 4.2.2. |

| | |
|---|---|
| SRC | Stochastic region contraction. See section 3.4. |
| G.T. | Ground truth values. See section 6.5. |
| Est. | Estimated values. |

### 2.1.3 List of common notations

| | |
|---|---|
| $M$ | Number of receivers or channels in a recording. |
| $N$ | Number of sound events in a recording. |
| $K$ | Dimensionality of the affine space spanned by $M$ and $N$, in this thesis the Euclidean space $\mathbb{R}^3$. |
| $\mathbf{r}$ | An arbitrary receiver with spatial Euclidean coordinates $(x, y, z)$. |
| $\mathbf{R}$ | A $K \times M$ matrix of receiver positions. |
| $\mathbf{r}_i$ | A column vector of $\mathbf{R}$ containing the coordinates of a single receiver. |
| $\mathbf{s}$ | An arbitrary sound source with spatial Euclidean coordinates $(x, y, z)$. |
| $\mathbf{S}$ | A $K \times N$ matrix of sound event positions. |
| $\mathbf{s}_j$ | A column vector of $\mathbf{S}$ containing the coordinates of a single sound event. |
| $u_{i,j}$ | An element $(i, j)$ of a matrix $\mathbf{u}$ containting TDOA values. See section 2.3 |
| $c$ | The signal propagation speed, in this thesis the speed of sound. |
| $o_j$ | The offset in the TDOA problem. See section 2.3. |
| $\tau$ | A delay between two signals. See section 2.3 . |
| $r_{p,q}(\tau)$ | The GCC-PHAT method. See section 4.1 |

### 2.1.4 List of common operators

| | |
|---|---|
| $\mathbf{x} * \mathbf{y}$ | The convolution of the vectors $\mathbf{x}$ and $\mathbf{y}$. |
| $\lvert\lvert\mathbf{x}\rvert\rvert$ | The Euclidean norm for vectors. |
| $\lvert\lvert\mathbf{x}\rvert\rvert_2$ | The $L_2$ norm for matrices. |
| $\lvert\lvert\mathbf{x}\rvert\rvert_F$ | The Frobenius norm for matrices. |
| $\overline{\mathbf{x}(\mathbf{t})}$ | The complex conjugate of $x(t)$. |
| $\{a_1, \ldots, a_n\}$ | A set containing the elements $a_1, \ldots, a_n$. |
| $A \setminus B$ | The set of elements in $A$ but not in $B$. |

## 2.2 Acoustic Model

In this section the acoustic model used in this thesis is defined. This includes assumptions about the model and concepts relating to the signal theory.

### 2.2.1 Assumptions and prerequisites

To build a complete and accurate model consisting of a number of sound sources and receivers is difficult. The sound sources all have different directionality and spatial attenuation. The speed of sound vary with temperature and altitude. Receivers are tuned differently. Therefore a few assumptions have been made about the sound sources, the receivers and the environment used in this thesis:

- All sound sources and receivers are omnidirectional and small. This means that they can be modelled as single points in the Euclidean space $\mathbb{R}^3$ and that the sound sources emit spherical sound waves from point sources.
- A direct path between a sound source and a receiver can always be found. There are no objects blocking the propagation path between them.

- The signal propagation speed (the speed of sound) is constant within one experiment. For this thesis the propagation speed is set to 340 m/s, which corresponds to sound waves travelling through air, at sea level and at 15 °C.

### 2.2.2 Near-field and far-field

When working with sound sources and receivers two different cases exists, namely the *far-field* situation and the *near-field* situation.

- The far-field situation describes a situation where the distance between a sound source and a receiver is much larger than the distances between the different receivers.
- The near-field situation describes a situation where the distance between a sound source and a receiver is smaller than or about the same length as the distances between the different receivers.

Each direct path between a sound source $\mathbf{s}$ and a receiver $\mathbf{r}$ has a direction vector described by

$$\overline{\mathbf{d}} = \frac{\mathbf{s}-\mathbf{r}}{|\mathbf{s}-\mathbf{r}|}. \tag{2.1}$$

The vector $\overline{\mathbf{d}}$ is called the *direction of arrival* (DOA) and is illustrated in fig. 2.1.

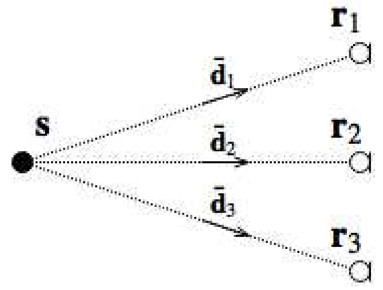


Figure 2.1: Illustation of the DOA concept. As the distance between the sound source $\mathbf{s}$ and the receivers $\mathbf{r}_i$ grows, the vectors $\overline{\mathbf{d}}_i$ merge into one single DOA $\bar{d}_o$. Circles filled with black represent sound sources and circles with a touching tangent represent receivers.

In the far-field case the DOA vectors spanned between a sound source and a number of different receivers can be assumed equal in a practical setting. In other words it is impossible to distinguish the different DOA vectors from each other when the sound source is very far away.

In this thesis it is assumed that the sound sources and the receivers are in the near-field situation. One of the goals in this thesis is to estimate the location of the sound sources, and thus it is necessary to have more information than just a single DOA vector for all the receivers.

### 2.2.3 Signal propagation model

A signal travelling along a direct path between a sound source and a receiver can be modelled by an *anechoic signal propagation model.* This means that no reverberations are taken into account. The signal is modelled as

$$x(t) = as(t-\tau) + w(t) \tag{2.2}$$

where $x(t)$ is the received signal at time $t$, $s(t)$ is the emitted signal with time delay $\tau$ between emission and reception, $a$ is the attenuation and $w(t)$ is the noise.

The environment in this thesis is however likely to have walls and other obstacles in the experimental setup. This will generate reverberation as the signal will bounce off the different surfaces and create a lot of indirect paths. To describe the multi-path propagation, equation (2.2) is remodelled into a *convolutive signal propagation model*

$$x(t) = s(t) * h(t) + w(t) \tag{2.3}$$

where $h(t)$ is a filter that models the propagation delay and the attenuation. Later, in section 4.1.3, it will be shown that the exact model for $h(t)$ does not need to be described as the effects will be mitigated with a smart choice of $h(t)$.

## 2.3 The time-difference of arrival problem

Let $\mathbf{s}$ be the spatial coordinates of a sound source which is emitting a single sound event. Let $\mathbf{r}_p$ and $\mathbf{r}_q$ be the spatial coordinates of two receivers. The *time-difference of arrival* (TDOA) is then defined as the difference in time it takes for a signal emitted from $\mathbf{s}$ to reach $\mathbf{r}_p$ compared to $\mathbf{r}_q$.

Let the timestamp of the signal emission be denoted $t_0$, with $t_p$ and $t_q$ denoting the reception times for $\mathbf{r}_p$ and $\mathbf{r}_q$, respectively. The distance between $\mathbf{s}$ and $\mathbf{r}_p$ can then be described by

$$||\mathbf{r}_p - \mathbf{s}|| = c\,(t_p - t_0), \tag{2.4}$$

where $c$ is the signal propagation speed. Hence the distance $u_{p,q}$ between $\mathbf{r}_p$ and $\mathbf{r}_q$ can be derived as

$$\begin{aligned} u_{p,q} &= ||\mathbf{r}_p - \mathbf{s}|| - ||\mathbf{r}_q - \mathbf{s}|| \\ &= c\,(t_p - t_0) - c\,(t_q - t_0) \\ &= c\,(t_p - t_q) \\ &= ||\mathbf{r}_p - \mathbf{r}_q||. \end{aligned} \tag{2.5}$$

An illustration of the concept can be found in fig. 2.2.

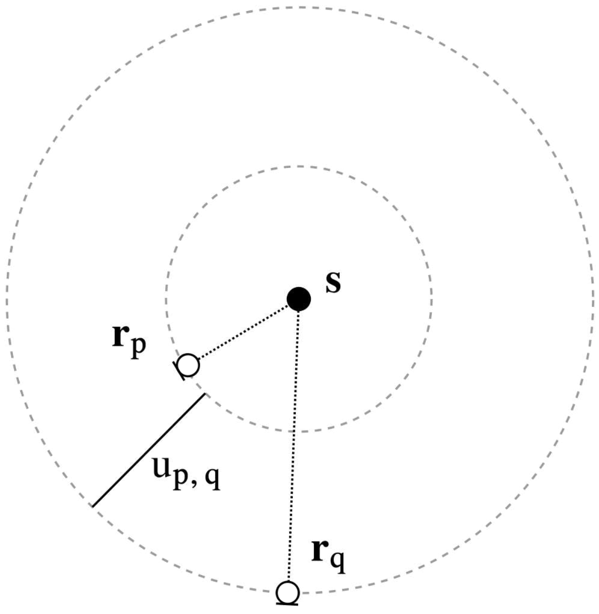


Figure 2.2: Illustration of equation (2.5), where the distance $u_{p,q}$ is shown. Circles filled with black represent sound sources and circles with a touching tangent represent receivers.

The input to the system in this thesis consists of $M$ sound recordings. Each channel corresponds to a receiver $\mathbf{r}_i$ at an unknown spatial position, for $i = 1, \ldots, M$. Fix channel one as a reference channel. For each channel pair $(i, 1)$ and a sound event from a sound source $\mathbf{s}$, equation (2.5) then gives $M$ distances

$$u_i = ||\mathbf{r}_i - \mathbf{s}|| - ||\mathbf{r}_1 - \mathbf{s}||. \tag{2.6}$$

The last part of the equation is equivalent for each index $i$, and can be rewritten as

$$o = -||\mathbf{r}_1 - \mathbf{s}|| = -c\,(t_1 - t_0), \tag{2.7}$$

yielding the equation

$$u_i = ||\mathbf{r}_i - \mathbf{s}|| + o. \tag{2.8}$$

A single sound source **s**, which is moved around, generates $N$ sound events. Each sound event is denoted $\mathbf{s_j}$ for $j = 1, \ldots, N$. Equation (2.8) then becomes

$$u_{i,j} = ||\mathbf{r}_i - \mathbf{s}_j|| + o_j, \tag{2.9}$$

for $i = 1, \ldots, M$ and $j = 1, \ldots, N$. An illustration of the concept can be found in fig. 2.3. The time-difference of arrival problem is thus the problem of estimating the unkonwn parameters $\mathbf{r}_i, \mathbf{s}_j$ and $o_j$ so that equation (2.9) holds true.

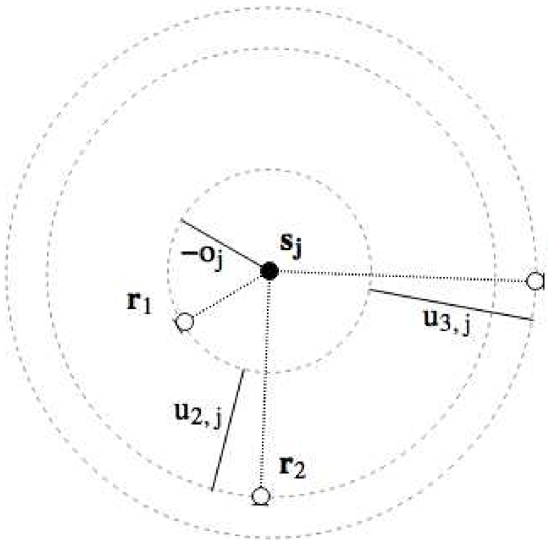


Figure 2.3: An illustration of equation (2.9), where the distances $u_{i,j}$ for $i = 1, \ldots, 3$ and offsets $o_j$ are shown. Circles filled with black represent sound sources and circles with a touching tangent represent receivers.

In the case of multiple simultaneously moving sound sources, the signals are separated before anything else is calculated. Thus the equations can be used as if there was only a single moving sound source.

## 2.4 The time of arrival problem

Closely related to the TDOA problem is the *time-of-arrival* (TOA) problem. The difference between the two is that for the TOA problem the absolute distances $d_{i,j}$ between each receiver and each sound event are considered known. Hence the problem, given distances $d_{i,j}$, is to estimate $\mathbf{r}_i$ and $\mathbf{s}_j$ such that

$$d_{i,j} = ||\mathbf{r}_i - \mathbf{s}_j|| \tag{2.10}$$

This means that once the offsets $o_j$ in equation (2.9) have been estimated the TDOA problem can be transformed into a TOA problem by setting $d_{i,j} = u_{i,j} - o_j$.

Chapter 3

# System Design

*The third chapter describes the design of the system pipeline. Arguments are made to defend why certain choices have been made and why other choices have been tossed aside.*

## 3.1 System overview

The system developed and implemented in this thesis is based on the research and implementations done by Zhayida et al. [10], Segerblom Rex [11] and Do [12]. Some parts remain the same while others are new. There are three big components in this system which can be seen as building blocks. They are summarized in the list below and then explained further in the other sections of this chapter. The three components are:

1. **Signal processing.** Generate, record and filter signals transmitted between the user's output devices and the receivers of the system.
2. **System self-calibration.** Estimate the locations of the receivers (as well as the output devices) to do a self-calibration of the system.
3. **Sound source tracking.** Estimate the locations of the user's output devices in real-time in order to track the movement on a computer monitor.

## 3.2 Signal processing

As described in the first chapter the motion tracking in this thesis will be done using sound waves as signals, instead of using a video feed. The system needs to be usable by a ordinary people within "normal" environments, for example a living room or an office space. This raises a few questions, namely:

- How will the sound emitted affect the user and the environment?
- How will the environment affect the system?
- How easy will it be for a user to use the system?

These questions will be addressed as the system design unfolds.

### 3.2.1 Sound frequencies

To avoid distracting the user, the sound wave frequencies would fall outside the audible spectrum. Such frequency bands are called *ultrasound*. The hearing range of ordinary people is commonly defined as 20 Hz - 20 kHz but the upper limit vary from person to person, and can often go as low as 8 kHz - 15 kHz for an adult [13, p. 747].

When talking about the frequency span in an application the *sampling rate* $F_s$ has to be taken into account. The Shannon-Nyquist theorem [14] states that the sampling rate should be at least twice the maximum frequency contained in the signal in order to avoid aliasing of the signal. Modern sound interfaces usually use fixed sample rates, commonly $F_s = 48$ kHz, 96 kHz or 192 kHz. With a signal in the ultrasound band, say around 30 kHz, a sampling rate of at least $F_s = 60$ kHz would be required. On the other hand the data processing requirements for a higher sampling rate are a lot higher than for a lower sampling rate. For a real-time system low data processing requirements are better as the system needs to be very fast. The aim of this thesis is to use a frequency range in the ultrasound band, but other bands will also be explored.

Another aspect to think about is that most mainstream microphones available in retail stores are built for human speech and normally ranges between about 500 Hz - 17 kHz. If the system is to be cheap and available to a ordinary people this has to be taken into account. More about this in section 6.2.

### 3.2.2 Microphones and output devices

Systems involving receivers and sound sources can be divided into two different cases, namely *closed systems* and *open systems*. In a closed system the signals emitted from the sound sources are known to the receivers beforehand. The receivers and sound sources form a “loop”. In other words the receivers have all the information about transmission times and the exact origins of the signals. In an open system the opposite is true. The receivers have no information about the transmitted signals beforehand. The only information available is the received signal and the timestamp at which it was received.

This thesis aims to have an open system. The reason for this being that the user has to be totally disconnected from the computer, so that the system to be comparable to other input methods. If it were a closed system the user would have to be connected to the computer by a wireless or wired communication line.

Now, the question of where to put the receivers and where to put the sound sources is easier to answer. This a question because there exists a duality in the problem to be solved. The techniques and algorithms used later on can be used in two ways. Either the receivers are stationary and are tracking one or more sound sources that are moved around, or the sound sources are stationary and the receivers are moved around. The first case matches the one where a number of microphones are placed around a computer screen and a small speaker is attached to each finger of the user.

There are two good arguments as to why the first case is the choice for this thesis. If all the receivers would have been strapped to the user’s fingers, the received signal would have to be relayed back to the system, i.e. it would have to be a closed system. With the sound sources on the user’s hand the signal could be generated locally with no connection to the system at all, enabling the user to use the hand device more like to a computer mouse. The second reason is that by transmitting the sound waves away from the user’s hand and towards the receivers, the sound waves get a directionality away from the user’s ear. If the opposite was true the user would be inside the sound direction all the time and exposed to potentially harmful high volume signals. Of course the user will still be hit by indirect sound wave paths but the exposure is a lot smaller.

### 3.2.3 Parallel signal transmissions from multiple sound sources

In order to track each of the user’s fingers individually the signals for each sound source needs to be separated in some way. Otherwise the system will not be able to differentiate individual fingers as the signals will interfere with each other. The principle is called *multiplexing* where the goal is to combine multiple signal transfers over a shared medium.

One method would be to let each sound source operate on a different frequency band. For example, the sound source $\mathbf{s}_1$ would operate on the band 30 kHz - 32 kHz and the sound source $\mathbf{s}_2$ would operate on the band 33 kHz - 35 kHz. This solution is called *frequency-division multiplexing* where the frequency band is split into multiple smaller bands. It is simple to implement but it does not scale very well as the number of available bands are limited.

Another method is called *time-division multiplexing*. Here the sound sources are only allowed to transmit for short intervals at given times, i.e. each sound source is given a transmission time slot. If the intervals are small enough the received signals should have enough data for simple interpolations. An upside to this method is that more frequency bands become available and multiple systems could be used parallel within the same environment. However it is likely to be more difficult to implement since the method increases the need to synchronize the system.

Both methods put a limit to the number of available bands as well as the number of parallel users, which is clearly a downside. Neither does the solutions take into account the digital security of the systems, leaving them open to malicious attacks. An ideal solution would encode the signals in some way, making them more secure and allowing users to share frequency bands. These methods are called *code-division multiple access*.

In this thesis a simple frequency division will be used, the reason being simplicity and to allow focus on other parts of the system. The only thing that needs to be implemented is a bandpass filter to separate the received signals.

## 3.3 System self-calibration

For the system to be easy to use for ordinary people the system setup and administration needs to be simple. Hence there should not be any requirements on how to place the receivers around the computer screen, nor should there be any distance measurements required. In other words the assumption is that no "correct" positioning of the receivers exists and they can be placed ad hoc.

Together with the fact that the sound sources will be non-stationary, in the sense that they are moved around, the system needs to be calibrated so that the positions of the sound sources can be correctly mapped onto the computer monitor. Since both the positions of the receivers and the sound sources are unknown this is called a *self-calibration*.

The self-calibration serves two purposes:

- To estimate the positions of the receivers for the real-time tracking (next section).
- To estimate the "virtual screen" from a sound source path so that the spatial positions of the fingers can be projected onto a computer monitor.

### 3.3.1 Scene reconstruction

Provided a number of receivers and a number of sound events, a *scene reconstruction* can be computed. The goal is to reconstruct the spatial positions of the receivers and sound sources from a number of recorded signal transmissions. The system will be based on the work by Zhayida et al. [10] and Segerblom Rex [11].

In Kuang and Åström [15] and in Ask et al. [16] solutions to the self-calibration are discussed in terms of different numbers of receivers and sound events as well as different room dimensions, i.e. whether the receivers and sound sources lie in a plane (2D), in the Euclidean space (3D) or any combination thereof. In this thesis it is assumed that both the receivers and the sound events have spatial positions in the Euclidean space.

The pipeline for the scene reconstruction has three components described below:

1. Estimate vectors $\mathbf{u}_j$ for the TDOA problem from the recorded signal transmissions.
2. Estimate the offsets $o_j$ from the vectors $\mathbf{u}_j$ using RANSAC [17] with minimal solvers and rank constraints [15].
3. Estimate receiver and sound event locations by solving the TOA problem.

The techniques used are robust in the sense that other sounds in the environment that could interfere with the system are negated.

### 3.3.2 The virtual screen

The output of the system are the locations of the receivers and the sound events for a sound source path. These locations are translation and rotation invariant, which means that the locations are correct in relation to each other, but for a fixed position the geometry can be rotated or translated. It turns out that for this thesis it is not a problem. By having the user "draw" a quadrilateral in the air following a certain direction and fixing one of the receiver positions in origo the system can work around these invariances. The quadrilateral is called a *virtual screen*.

The user can be given the instructions to move a finger along a rectangle in the air, starting from the bottom-right corner, to the bottom-left corner, to the top-left corner and so on. The size of the rectangle will be projected to the size of the computer monitor, giving the user an option to decide how large gestures he or she wants to use.

In this thesis the rectangle will be fitted to the sound event positions and RANSAC.

## 3.4 Sound source tracking and coordinate projection

Once the system is calibrated the receiver positions are considered to be known to the system. Using these, real-time tracking of the sound sources can be computed. The pipeline of the scene reconstruction could be used here as well, but as it requires multiple measurements of sound events it is not as good as other techniques.

There are (roughly) three categories into which algorithms for sound source localization can be divided. One of them is the aforementioned two-stage TDOA algorithms. Another category is steered beamformer algorithms [18, 19]. The last category is high-resolution spectrum estimation algorithms [20]. The TDOA-based algorithms have stood the test of time for realistic implementations in real-time. However, in Bradstein and Ward [21] and Birchfield [22] it has been shown that one-stage methods from the second category, using steered response power (SRP) algorithms, are more robust, although at the price of a higher computational cost.

In this thesis a method called *Coarse-to-fine region contraction* (CFRC) will be used to improve the SRP. The CFRC method is introduced in Do and Silverman [23] and explored further in Do [12]. It is an extension of the *Stochastic region contraction* (SRC) [24]. The CFRC method has a smaller computational cost than the SRC and both methods aims to reduce the computational cost of the SRP method. Another reason for using the CFRC method in this thesis is to get a comparison of the TDOA-approach and the CFRC approach.

After the three-dimensional position of a sound event is estimated it is projected onto the virtual screen using linear algebra.

Chapter 4

# Techniques and theory

*The techniques and theory chapter presents all the algorithms, derivation of formulas and mathematics used in the system implementation.*

## 4.1 Generalized cross-correlation with Phase Transform

The GCC-PHAT method is used to find the TDOA between two different signals so that they can be used for sound source localization. The GCC method is an extension of normal cross-correlation with an added weighting function. The GCC is studied in Knapp and Carter [25]. By using a specific filter called PHAT, the effect of noise and reverberation can be mitigated. The PHAT filter was first introduced in Carter et al. [26]. This section describes the derivation of the GCC-PHAT method.

### 4.1.1 Cross-correlation

To compare two signals cross-correlation can be used. This will give a measurement of how "similar" two signals are for different time delays $\tau$. The cross-correlation between two continuous signals $x_p(t)$ and $x_q(t)$ is given by

$$c_{p,q}(\tau) = \int_{-\infty}^{\infty} \overline{x_p(t)}\, x_q(t+\tau)\, dt, \tag{4.1}$$

where $\overline{x_p(t)}$ is the complex conjugate of $x_p(t)$. This function will reach its maximum value when the signals are perfectly aligned, i.e. when the corresponding time delay $\tau$ has the correct value. Mathematically this is equivalent to finding

$$\tau_{p,q} = \underset{\tau}{\operatorname{argmax}}\; c_{p,q}(\tau). \tag{4.2}$$

Applying the Fourier Transform to equation (4.1) yields

$$C_{p,q}(\omega) = \int_{-\infty}^{\infty} c_{p,q}(\tau) e^{j\omega\tau} d\tau. \tag{4.3}$$

Note that equation (4.1) is very similar to the definition of the convolution between $x_p(t)$ and $x_q(t)$, the only difference being the use of the complex conjugate of the term $x_p(t)$. With the convolutive properties of the Fourier Transform, $C_{p,q}(\omega)$ could also be written as

$$C_{p,q}(\omega) = \overline{X_p(\omega)} X_q(\omega), \tag{4.4}$$

where $X_p(\omega)$ and $X_q(\omega)$ are the Fourier Transforms of $x_p(t)$ and $x_q(t)$. By substituting equation (4.4) into equation (4.3) and taking the inverse Fourier Transform of the whole expression, a new expression for the cross-correlation as terms of a Fourier Transform is obtained as

$$c_{p,q}(\tau) = \int_{-\infty}^{\infty} \overline{X_p(\omega)} X_q(\omega) e^{j\omega\tau} d\omega. \tag{4.5}$$

### 4.1.2 Generalized cross-correlation

Introduce a filter function $g(t)$. A filtered signal is obtained by convolving the signal with the filter. For the new filtered signals $x_p(t) * g_p(t)$ and $x_q(t) * g_q(t)$, the cross-correlation defined by equation (4.5) then becomes

$$r_{p,q}(\tau) = \int_{-\infty}^{\infty} \overline{X_p(\omega)G_p(\omega)} X_q(\omega) G_q(\omega) e^{j\omega\tau} d\omega \tag{4.6}$$

where $G(\omega)$ is the Fourier Transform of $g(t)$. This also happens to be the cross-correlation between two signals modelled by the convolutive signal propagation model as defined in equation (2.3), with $h(t) = g(t)$ and the noise omitted.

Introducing a combined weighting function

$$\psi_{p,q}(\omega) = \overline{G_p(\omega)} G_q(\omega) \tag{4.7}$$

and substituting it into equation (4.6) yields

$$r_{p,q}(\tau) = \int_{-\infty}^{\infty} \psi_{p,q}(\omega) \overline{X_p(\omega)} X_q(\omega) e^{j\omega\tau} d\omega. \tag{4.8}$$

This is known as the *Generalized Cross-Correlation* (GCC).

### 4.1.3 The Phase Transform

The weighting function defined by equation (4.7) is supposed to mitigate the effects of noise and reverberation. It has been shown in Zhang et al. [27] that the filter

$$\psi_{p,q}(\omega) = \frac{1}{|\overline{X_p(\omega)} X_q(\omega)|} \tag{4.9}$$

works well for this purpose. This is known as the *Phase Transform* (PHAT). It performs worse than the maximum likelihood weighting function in reverberant-free environments [28], but as the system in this thesis will have a lot of reverberations it is a good fit. Thus, as mentioned in section 2.2.3, there is never any need to exactly define the filter function $h(t)$.

Combining the GCC with the PHAT gives the following formula

$$r_{p,q}(\tau) = \int_{-\infty}^{\infty} \frac{1}{|\overline{X_p(\omega)} X_q(\omega)|} \overline{X_p(\omega)} X_q(\omega) e^{j\omega\tau} d\omega, \tag{4.10}$$

which is known as the *GCC-PHAT* method. Note that the GCC-PHAT can have many local maxima.

## 4.2 Steered Response Power with Phase Transform

A *beamformer* is essentially a microphone array (either placed in a fixed grid or ad hoc) that has the ability to focus on signals emitted from specific locations. The beamformer can *steer* over an area to find the position of a sound source $\mathbf{s}$. The output of the beamformer for each point $\mathbf{p}$ in the area has a *steered response power* (SRP) that reaches its maxima when $\mathbf{p}$ equals $\mathbf{s}$.

A sophisticated beamformer uses a *filter-and-sum* algorithm where an adaptive filter is applied to the signals before they are summed. The filter used here will be the PHAT filter defined in section 4.1.3. This section will describe the derivation of the SRP-PHAT from Do [12].

### 4.2.1 Derivation of the SRP-PHAT

Recall from eq. (2.3) that the signal $x_k(t)$ is the received signal at a receiver $\mathbf{r}_k$ of an emitted signal $s(t)$ at time $t$ from a point $\mathbf{p}$. For a beamformer of $M$ receivers, the signal $s(t)$ is received at different times for each individual receiver. Thus the received signal is

$$x_k(t - \delta_k) \tag{4.11}$$

for $k = 1, \dots, M$. Here the delays $\boldsymbol{\delta}$ are called *steering delays.* In order to sum the signals they need to be time-aligned so that the mapping of the signals become correct. This process is called a *add-and-sum* algorithm and is defined as

$$y(t, \boldsymbol{\delta}) = \sum_{k=0}^{M} x_k(t - \delta_k). \tag{4.12}$$

If a filter $g(t)$ is applied to $x(t)$ eq. (4.12) becomes a *filter-and-sum* process

$$y(t, \boldsymbol{\delta}) = \sum_{k=0}^{M} x_k(t - \delta_k) * g(t), \tag{4.13}$$

which in the domain frequency becomes

$$Y(t, \boldsymbol{\delta}) = \sum_{k=0}^{M} G_k(\omega) X_k(\omega) e^{-j\omega\delta_k}. \tag{4.14}$$

Now, fix one of the steering delays $\delta_i$ as $\delta_0$ (preferably the smallest) so that the other steering delays $\delta_k$, $k \in \{1, \dots, M\} \setminus \{i\}$, can be expressed as time delays $\tau_k$ relative to the reference receiver. Define $\tau_k = \delta_k - \delta_0$. For a receiver pair $(p, q)$ the time delay is then

$$\tau_{p,q} = \tau_p - \tau_q = (\delta_p - \delta_0) - (\delta_q - \delta_0) = \delta_p - \delta_q. \tag{4.15}$$

When formulating the steering response power (SRP) it is more interesting to work with the relative time delays $\tau$. The SRP is defined as the output power of a filter-and-sum beamformer when steering over all points $\mathbf{p}$ in a given grid. In other words this is a sum over all pairs $(p, q)$ of filtered signals $x(t) * g(t)$ for different time delays $\tau_{p,q}$.

Thus, for each point $\mathbf{p}$, the SRP in the frequency domain is defined as

$$P(\boldsymbol{\tau}) = \int_{-\infty}^{\infty} \overline{Y_p(\omega, \boldsymbol{\tau})} Y_q(\omega, \boldsymbol{\tau}) d\omega, \tag{4.16}$$

much like the cross-correlation. Expanding $P(\boldsymbol{\tau})$ yields

$$\begin{aligned} P(\boldsymbol{\tau}) &= \int_{-\infty}^{\infty} \left( \sum_{p=1}^{M} \overline{X_p(\omega) G_p(\omega) e^{-j\omega\tau_p}} \right) \left( \sum_{q=1}^{M} X_q(\omega) G_q(\omega) e^{j\omega\tau_q} \right) d\omega \\ &= \int_{-\infty}^{\infty} \sum_{p=1}^{M} \sum_{q=1}^{M} \left( \overline{G_p(\omega)} G_q(\omega) \right) \left( \overline{X_p(\omega)} X_q(\omega) \right) e^{j\omega(\tau_q - \tau_p)} d\omega \\ &= \int_{-\infty}^{\infty} \sum_{p=1}^{M} \sum_{q=1}^{M} \psi_{p,q}(\omega) \overline{X_p(\omega)} X_q(\omega) e^{j\omega\tau_{q,p}} d\omega \end{aligned} \tag{4.17}$$

for a filter $\psi_{p,q}(\omega) = \overline{G_p(\omega)} G_q(\omega)$. Note that this is the same definition as in eq. (4.7).

Since both the signals and their filters have finite energy the intergral converges eventually. Thus the integral and the sums are interchangeable. $P(\boldsymbol{\tau})$ from eq. (4.17) then becomes

$$\begin{aligned} P(\boldsymbol{\tau}) &= \sum_{p=1}^{M} \sum_{q=1}^{M} \int_{-\infty}^{\infty} \psi_{p,q}(\omega) \overline{X_p(\omega)} X_q(\omega) e^{j\omega\tau_{p,q}} d\omega \\ &= \sum_{p=1}^{M} \sum_{q=1}^{M} r_{p,q}(\tau), \end{aligned} \tag{4.18}$$

where $r_{p,q}(\tau)$ is the GCC-PHAT from eq. (4.10) between signals $x_p(t)$ and $x_q(t)$. Note that the filtering function $\psi_{p,q}(\omega)$ is set to the PHAT filter from section 4.1.3.

Expressing this in its full form yields

$$P(\boldsymbol{\tau}) = \sum_{p=1}^{M} \sum_{q=1}^{M} \int_{-\infty}^{\infty} \frac{1}{|\overline{X_p(\omega)} X_q(\omega)|} \overline{X_p(\omega)} X_q(\omega) e^{j\omega\tau_{p,q}} d\omega. \tag{4.19}$$

The equation is known as the SRP-PHAT.

There is one last simplification to be done however. Since GCC-PHAT $r_{p,q}(\tau)$ is the same as $r_{q,p}(\tau)$ this means that the matrix of pairs $(p, q)$ forming the double sums in eq. (4.19), is a symmetric matrix with close-to-zero elements in its diagonal (the delay between receivers $p$ and $p$ should be zero). Hence it is enough to do a summation over the pairs in an upper-triangle since the output of the SRP-PHAT will just be lowered by a factor of two.

Redefine $P(\boldsymbol{\tau})$ as

$$P'(\boldsymbol{\tau}) = \sum_{p=1}^{M} \sum_{q=p+1}^{M} \int_{-\infty}^{\infty} \frac{1}{|\overline{X_p(\omega)} X_q(\omega)|} \overline{X_p(\omega)} X_q(\omega) e^{j\omega\tau_{p,q}} d\omega. \tag{4.20}$$

### 4.2.2 Coarse-to-fine region contraction

To find a sound source $\mathbf{s}$ with the SRP-PHAT method a full grid search over a predefined grid needs to be done. For every point $\mathbf{p}$ in the grid, the function $P'(\boldsymbol{\tau})$ has to be evaluated. This becomes very expensive in terms of computing cost. Thus a method to reduce the number of evaluations is needed. One such method is the Coarse-to-fine region contraction (CFRC) [23].

The idea is to select a number of equally spaced grid points, evaluate the SRP-PHAT for them and then find a bounding box for the $n$ best evaluations. Iteratively the bounding box, i.e. the region, is contracted until a small enough volume is trapped, so that the SRP-PHAT can be evaluated for every grid point contained therein. Hence the name coarse-to-fine region contraction; first a coarse grid net is evaluated and then it is contracted into a finer and finer region for each iteration.

The first step is to determine some parameters for the initial step of the algorithm. A target volume $V_t$ that traps the maxima of the SRP-PHAT is needed as the peak is most likely larger than a single grid point, especially considering spatial errors in the measurements. Also a search volume $V_S$ for which the grid is to be formed is required. In order to make sure at least one grid point lies in $V_t$ the initial grid points should be spaced no further apart than $\frac{V_S}{V_t}$ in each spatial direction. The initial grid points is denoted $J_0$. Finally the number of best results, $N_0$, of the SRP-PHAT needs to be set (should be significantly smaller than $J_0$).

The steps of the CFRC are:

1. **Initialization.** Set the iteration counter $i = 0$, target volume $V_t$, search volume $V_i = V_S$, grid points to be evaluated $J_i = J_0$, number of results to keep $N_i = N_0$ and saved points $G_i = 0$.

2. **Evaluation.** Evaluate $P'(\mathbf{p})$ for each point $\mathbf{p}$ in $J_i$.

3. **Sort and select.** Sort and save the $N_i$ best points of the union $J_i$ and $G_i$.

4. **Contract region.** Find the smallest region $V_{i+1}$ that traps the $N_i$ points.

5. **Termination test.**
   - If $V_{i+1} < V_t$ then terminate with the best point in $V_{i+1}$.
   - If $V_{i+1} = V_i$ then evaluate all the points in $V_{i+1}$ and terminate with the best point found.
   - Otherwise, from the $N_i$ points keep a subset of $G_{i+1}$ points that have values $\geq$ the mean $\mu$ of the $N_i$ points.

6. **Select grid points.** Setup a new grid $J_{i+1}$ of points to be evaluated and $N_{i+1}$ points to be saved.

7. **Iteration step**. Set $i = i + 1$ and continue from step 2.

Thus a way to compute the new $J_{i+1}$ grid points as well as the new number $N_{i+1}$ is required in the implementation.

## 4.3 Techniques for solving the TDOA and the TOA problems

When solving the TDOA and the TOA problems something called *minimal solvers* are used. Combined with RANSAC techniques these become very powerful and robust. First some useful expansions of the problem formulations are deduced. Begin by rewriting equation (2.9) by moving $o_j$ to the other side and then squaring both sides in order to obtain the equation

$$(u_{i,j} - o_j)^2 = ||\mathbf{r}_i - \mathbf{s}_j||^2 = (\mathbf{r}_i - \mathbf{s}_j)^T(\mathbf{r}_i - \mathbf{r}_j), \tag{4.21}$$

which is equivalent to

$$u_{i,j} - 2u_{i,j}o_j + o_j^2 = \mathbf{r}_i^T\mathbf{r}_i - 2\mathbf{r}_i^T\mathbf{s}_j + \mathbf{s}_j^T\mathbf{s}_j. \tag{4.22}$$

Let

$$\mathbf{R}_i = \begin{bmatrix} 1 \\ \mathbf{r}_i \\ \mathbf{r}_i^T\mathbf{r}_i \end{bmatrix} \tag{4.23}$$

be column vectors of $\mathbf{R}$ and

$$\mathbf{S}_j = \begin{bmatrix} \mathbf{s}_j^T\mathbf{s}_j - o_j^2 \\ -2\mathbf{s}_j \\ 1 \end{bmatrix} \tag{4.24}$$

be column vectors of $\mathbf{S}$. The size of the matrix $\mathbf{R}$ will be $(K+2) \times M$ and the size of the matrix $\mathbf{S}$ will be $(K+2) \times N$, where $K$ is the dimensionality of the affine space spanned by $\mathbf{r}_i$ and $\mathbf{s}_j$. Remember that for this thesis $K = 3$.

Now equation (4.22) can be written as

$$\mathbf{D} = \mathbf{R}^T\mathbf{S} \tag{4.25}$$

where $\mathbf{D}$ is a $M \times N$ matrix with elements $d_{i,j} = u_{i,j}^2 - 2u_{i,j}o_j$. Since $\mathbf{D}$ is a matrix product, $\text{rank}(\mathbf{D}) \leq \text{rank}(\mathbf{R})$ and $\text{rank}(\mathbf{D}) \leq \text{rank}(\mathbf{S})$, which means that $\text{rank}(\mathbf{D}) \leq (K+2)$.

Construct a $(M-1) \times (N-1)$ matrix $\mathbf{F}$ such that

$$\mathbf{F} = \mathbf{C}_M^T\mathbf{D}\mathbf{C}_N = \mathbf{C}_M^T\mathbf{R}^T\mathbf{S}\mathbf{C}_N = (\mathbf{R}\mathbf{C}_M)^T\mathbf{S}\mathbf{C}_N = \widetilde{\mathbf{R}}^T\widetilde{\mathbf{S}}, \tag{4.26}$$

where $\widetilde{\mathbf{R}} = \mathbf{R}\mathbf{C}_M$, $\widetilde{\mathbf{S}} = \mathbf{S}\mathbf{C}_N$, $\mathbf{C}_M = [-\mathbf{1}_{(M-1)}\ \mathbf{I}_{(M-1)}]^T$ and $\mathbf{C}_N = [-\mathbf{1}_{(N-1)}\ \mathbf{I}_{(N-1)}]^T$. Here $\mathbf{1}_{(k-1)}$ is the $(k-1) \times 1$ matrix consisting of ones and $\mathbf{I}_{(k-1)}$ is a $(k-1) \times (k-1)$ identity matrix.

Studying the matrices $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{S}}$ it becomes clear that the first row of $\widetilde{\mathbf{R}}$ and the last row of $\widetilde{\mathbf{S}}$ will be zero. Thus, when removing the last row of $\widetilde{\mathbf{R}}$ and the first row of $\widetilde{\mathbf{S}}$, the equality $\mathbf{F} = \widetilde{\mathbf{R}}^T\widetilde{\mathbf{S}}$ is preserved. Redefine $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{S}}$ as described above to obtain $\widetilde{\mathbf{R}}_i = [(\mathbf{r}_{i+1} - \mathbf{r}_1)]_{K\times 1}$, $\widetilde{\mathbf{S}}_j = [-2(\mathbf{s}_{j+1} - \mathbf{s}_1)]_{K\times 1}$ and $\mathbf{F}$ with elements

$$\begin{aligned} f_{i,j} = \ & u_{(i+1,j+1)}^2 - 2u_{(i+1,j+1)}o_{j+1} \\ & -u_{(1,j+1)}^2 + 2u_{(1,j+1)}o_{j+1} \\ & -u_{(i+1,1)}^2 + 2u_{(i+1,1)}o_1 \\ & +u_{(1,1)}^2 - 2u_{(1,1)}o_1. \end{aligned} \tag{4.27}$$

The matrix $\mathbf{F}(\mathbf{u}, \mathbf{o})$ is called *the double compaction matrix* and has at most rank $K$, that is $\text{rank}(\mathbf{F}) \leq K$. The arguments about the ranks of the matrices $\mathbf{F}$ and $\mathbf{D}$ are called *rank-constraints.*

## 4.3.1 Minimal solver for the TDOA problem

The column vectors of the double compaction matrix $\mathbf{F}(\mathbf{u}, \mathbf{o})$ are all functions of unknown offsets $\{o_1, \ldots, o_N\}$. To solve for these offsets, rank constraints are enforced on the sub-matrices of $\mathbf{F}$.

Let $\mathbf{Q}$ be all the $(K+1) \times (K+1)$ sub-matrices of $\mathbf{F}$. They will all be rank deficient and have a maximum rank $K$. Formulate the rank constraints using determinants: for each sub-matrix $\mathbf{Q}$ the determinant will be zero, i.e. $\det \mathbf{Q} = 0$. The number of such constraints are

$$N_{\mathbf{Q}} = \binom{M-1}{K+1} \cdot \binom{N-1}{K-1}, \tag{4.28}$$

and among these $(M-1-K) \cdot (N-1-K)$ are linearly independent. Thus, each constraint gives rise to a polynomial equation of degree $K+1$ in offsets $\{o_1, \ldots, o_N\}$.

The equation system is solvable and well-defined for a number of different combinations of $M$ and $N$. Different cases are explored in Kuang and Åström [15]. Some solvable cases for $K = 3$ can be found in table 4.1. The equations can be solved using numerically stable polynomial methods based on the techniques in Byröd et al. [29].

| $(M, N)$ | Solutions | Solved in |
|---|---|---|
| (6, 8) | 14 | [15] |
| (7, 6) | 5 | [15] |
| (9, 5) | 1 | [15] |
| (10, 5) | - | [30] |

Table 4.1: Different combinations of $(M, N)$ for which the minimal cases are solvable.

### 4.3.2 Rank-based non-linear optimization for offsets $o_j$

In order to improve the estimated offsets $o_j$ that are found by the minimal solvers in section 4.3.1 an iterative rank-based non-linear optimization technique is applied. It can also be applied to extend the solution of offsets to the remaining receivers for the subset of columns in the minimal solution. The technique can help with missing data, which the previous algorithms for the TDOA solver cannot handle. The scheme is found in Zhayida et al. [31].

Given the estimated TDOA values $u_{i,j}$ and an initial estimation of the offsets $o_j$ the double compaction matrix $\mathbf{F}(\mathbf{u}, \mathbf{o})$ can be formed. The matrix is at most of rank $K$. The idea is to find better offsets such that a matrix $A$ after compaction is as close to rank $K$ as possible. In other words it is a minimization problem

$$\min_{\mathbf{o},\mathbf{A}} ||\mathbf{F}(\mathbf{u}, \mathbf{o}) - \mathbf{A}||_{F,\Omega} \quad \text{subject to} \quad \text{rank}(\mathbf{A}) = K, \tag{4.29}$$

where $A$ is a $(M-1) \times (N-1)$ matrix and $|| \cdot ||_{F,\Omega}$ is the Frobenious norm on matrix entries observed, specified by the set $\Omega$. Basically, the problem described is a least-squares problem. It is solved using the iterative Newton-Gauss optimization algorithm [32].

Let $\mathbf{A}_k$ and $\mathbf{o}_k$ be the estimations for each iteration. A local parametrization of $\mathbf{A}_k$ can be found by doing a singular value decomposition. Let

$$\mathbf{A}_k = U E_k W, \tag{4.30}$$

where $U$ is the $M \times M$ unitary matrix from the singular value decomposition $\mathbf{A}_k = USV^T$, $W$ is the $K \times N$ matrix formed by the first $K$ rows of $SV^T$ and $E_k$ is a $M \times K$ matrix formed by the first $K$ columns of a $M \times M$ identity matrix.

The local parametrization is

$$\mathbf{A}_{k+1} = U\, e^{\sum_{i=1}^{N} z_i B_i}\, E_k \left( \sum_{j=1}^{KN} (W + w_j C_j) \right), \tag{4.31}$$

where $C_j$ form a basis for $K \times N$ matrices, $B_i$ form a basis for $M \times M$ antisymmetric matrices with zeros in the upper-left $K \times K$ block and zeros in the lower-right $(M-K) \times (M-K)$ block. Thus the parametrization has $M + K(M-K) + KN$ parameters $x = (y, z, w)$.

Calculate the analytic derivatives of $\mathbf{A}_{k+1}$ with respect to $x$ and the derivatives of $\mathbf{F}(\mathbf{u}, \mathbf{o})$ with respect to $o$ in order to make it possible to compute the residuals and the Jacobeans for the algorithm.

### 4.3.3 Extending the solution of the minimal TDOA solver for remaining TDOA values

The goal is to extend the offsets $o_j$ found by the minimal TDOA solver for the subset *hvecu* (see section 4.3.1) to cover all the columns of TDOA matrix $\mathbf{u}$, i.e. the columns $\mathbf{u}_j, j = \{1, \ldots, N\} \setminus \{j \in \hat{\mathbf{u}}\}$. The new offsets are calculated columnwise.

After the calculations in section 4.3.2 the relation $\widetilde{\mathbf{F}}(\mathbf{u}, \mathbf{o}) \approx \mathbf{A}$ holds true. Using eq. (4.26) to substitute $\widetilde{\mathbf{F}}(\mathbf{u}, o)$ yields

$$\mathbf{A} \approx \mathbf{C}_M^T \mathbf{F} \mathbf{C}_N = \mathbf{C}_M^T \{u_{i,j} - 2u_{i,j} o_j\} \mathbf{C}_N. \tag{4.32}$$

Now, fix $j$ and the right hand side can be written as

$$\mathbf{C}_M^T\{\mathbf{u}_i - 2\mathbf{u}_i o\}\mathbf{C}_N \tag{4.33}$$

where $u_i$ is a column vector. Multiplying with $\mathbf{C}_N$ from the right is equivalent to subtracting the first column, which in this case will be the first column of the solved TDOA subset $\hat{\mathbf{u}}$, from $\mathbf{u}_i^2 - 2\mathbf{u}_i o$. This yields

$$\begin{aligned}\mathbf{C}_M^T\{\mathbf{u}_i^2 - 2\mathbf{u}_i o\}\mathbf{C}_N &= \mathbf{C}_M^T\{\mathbf{u}_i^2 - 2\mathbf{u}_i o \ - \underbrace{(\hat{\mathbf{u}}_1^2 - 2\hat{\mathbf{u}}_1 o_1)}_{=\gamma}\} \\ &= \mathbf{C}_M^T \underbrace{\{\mathbf{u}_i^2 - \gamma\}}_{=\boldsymbol{\alpha}_i} - \mathbf{C}_M^T \underbrace{2\{\mathbf{u}_i\}}_{=\boldsymbol{\beta}_i} o.\end{aligned} \tag{4.34}$$

Likewise multiplying with $\mathbf{C}_M$ from the left is equivalent to removing the first row from all elements of the column vectors. The first element of the column vector (which is now a zero) is removed so that eq. (4.32) holds. The resulting expression is

$$(\boldsymbol{\alpha}_k - \alpha_1) - (\boldsymbol{\beta}_k - \beta_1)o, \tag{4.35}$$

where $k = 1, \dots, (M-1)$.

Now pick $K$ random column vectors from $\mathbf{A}$ to form a basis $\mathbf{A}_{\text{basis}}$. Since $\mathbf{A}$ is at most rank $K$ this means that all columns vectors $\mathbf{A}_j, j = 1, \dots, (N-1)$, of $\mathbf{A}$ is linearly dependent on $\mathbf{A}_{\text{basis}}$ and can be written as

$$\mathbf{A}_j = \mathbf{A}_{\text{basis}}\mathbf{x}, \tag{4.36}$$

for some column vector $\mathbf{x}$. Note that $\mathbf{x}$ has $K = 3$ elements.

Using the same fixed $j$ as above, eq. (4.35) combined with eq. (4.36) gives

$$\mathbf{A}_{\text{basis}}\mathbf{x} \approx \underbrace{(\boldsymbol{\alpha}_k - \alpha_1)}_{=\mathbf{a}} - \underbrace{(\boldsymbol{\beta}_k - \beta_1)}_{=\mathbf{b}} o \tag{4.37}$$

which can be rewritten as

$$\mathbf{a} \approx \mathbf{A}_{\text{basis}}\mathbf{x} + \mathbf{b}o. \tag{4.38}$$

In matrix notation this becomes

$$\mathbf{a} \approx [\mathbf{A}_{\text{basis}}\ \mathbf{b}] \cdot [\mathbf{x}\ o]^T \tag{4.39}$$

and can be solved for $o$ and $\mathbf{x}$ using the MATLAB backslash operator in

$$[\mathbf{x}\ o]^T \approx [\mathbf{A}_{\text{basis}}\ \mathbf{b}] \setminus \mathbf{a}. \tag{4.40}$$

### 4.3.4 Solving the TOA problem

Reducing the TDOA problem to a TOA problem is done by setting $d_{i,j} = u_{i,j} - o_j$ since the offsets $o_j$ are known. Then, for the TOA problem, the following equation is true

$$d_{i,j} = ||\mathbf{r}_i - \mathbf{s}_j||, \tag{4.41}$$

for column vectors $\mathbf{r}_i$ and $\mathbf{s}_j$ and distance estimations $d_{i,j}$. Square both sides to get

$$d_{i,j}^2 = (\mathbf{r}_i - \mathbf{s}_j)^T(\mathbf{r}_i - \mathbf{r}_j) = \mathbf{r}_i^T\mathbf{r}_i - 2\mathbf{r}_i^T\mathbf{s}_j + \mathbf{s}_j^T\mathbf{s}_j \tag{4.42}$$

For the index pairs $(i,1)$, $(1,j)$ and $(1,1)$ the following three equations are derived

$$d_{i,1}^2 = \mathbf{r}_i^T\mathbf{r}_i - 2\mathbf{r}_i^T\mathbf{s}_1 + \mathbf{s}_1^T\mathbf{s}_1 \tag{4.43a}$$

$$d_{1,j}^2 = \mathbf{r}_1^T\mathbf{r}_1 - 2\mathbf{r}_1^T\mathbf{s}_j + \mathbf{s}_j^T\mathbf{s}_j \tag{4.43b}$$

$$d_{1,1}^2 = \mathbf{r}_1^T\mathbf{r}_1 - 2\mathbf{r}_1^T\mathbf{s}_1 + \mathbf{s}_1^T\mathbf{s}_1 \tag{4.43c}$$

Then add equation (4.43c) to equation (4.42) and subtract equations (4.43a) and (4.43b) from the same equation to get

$$\frac{d_{i,j}^2 - d_{i,1}^2 - d_{1,j}^2 + d_{1,1}^2}{-2} = (\mathbf{r}_i - \mathbf{r}_1)^T(\mathbf{s}_j - \mathbf{s}_1) \tag{4.44}$$

Thus, for $i = 2, \ldots, M$ and $j = 2, \ldots N$, there are $(M-1) \times (N-1)$ equations (4.44). Let $\mathbf{R}$ be a $3 \times (M-1)$ matrix, $\mathbf{S}$ be a $3 \times (N-1)$ matrix,

$$\begin{aligned} \mathbf{R}_i &= [(\mathbf{r}_i - \mathbf{r}_1)]_{3\times 1} && \text{be column vectors of } \mathbf{R}, \\ \mathbf{S}_j &= [(\mathbf{s}_j - \mathbf{s}_1)]_{3\times 1} && \text{be column vectors of } \mathbf{S} \text{ and} \\ \tilde{\mathbf{D}} &= \left\{ \frac{d_{i,j}^2 - d_{i,1}^2 - d_{1,j}^2 + d_{1,1}^2}{-2} \right\} && \text{be a } (M-1) \times (N-1) \text{ matrix.} \end{aligned}$$

Then $\tilde{\mathbf{D}} = \mathbf{R}^T\mathbf{S}$ where the rank depends on the affine span of the sound events and the receivers. Hence $\tilde{\mathbf{D}}$ will have rank 3.

Using singular value decomposition $\tilde{\mathbf{D}} = \tilde{\mathbf{R}}^T\mathbf{L}^{-1}\mathbf{L}\tilde{\mathbf{S}} = \mathbf{R}^T\mathbf{S}$ is computed for an unknown full-rank matrix $\mathbf{L}$. This is done using the Eckart-Young theorem [33] by partitioning the singular value decomposition

$$\tilde{\mathbf{D}} = \mathbf{U\Sigma V}^T \tag{4.45}$$

into

$$\mathbf{U} = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix}, \qquad \mathbf{\Sigma} = \begin{bmatrix}\mathbf{\Sigma}_1 & 0 \\ 0 & \mathbf{\Sigma}_2\end{bmatrix} \qquad \text{and} \qquad \mathbf{V} = \begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2\end{bmatrix} \tag{4.46}$$

where $\mathbf{\Sigma}_1$ is a $r \times r$ matrix, $\mathbf{U}_1$ is a $(M-1) \times r$ matrix and $\mathbf{V}_1$ is a $(N-1) \times r$ matrix. Then $\tilde{\mathbf{R}} = \mathbf{U}_1^T$ and $\tilde{\mathbf{S}} = \mathbf{\Sigma}_1\mathbf{V}_1^T$.

Now parameterize so that $\mathbf{R} = \mathbf{L}^{-T}\tilde{\mathbf{R}}$ (hence $\mathbf{R}^T = \tilde{\mathbf{R}}^T\mathbf{L}^{-1}$) and $\mathbf{S} = \mathbf{L}\tilde{\mathbf{S}}$. Let

$$\mathbf{r}_1 = \mathbf{0}_{3\times 1} \qquad \text{i.e. fix receiver 1 in origo,} \tag{4.47}$$

$$\mathbf{s}_1 = \mathbf{Lb}, \tag{4.48}$$

$$\mathbf{r}_i = \mathbf{L}^{-T}\tilde{\mathbf{R}}_i, \qquad i = 2, \ldots, M, \tag{4.49}$$

$$\mathbf{s}_j = \mathbf{L}(\tilde{\mathbf{S}} + \mathbf{b}), \qquad j = 2, \ldots . N \tag{4.50}$$

This results in $M-1$ equations

$$d_{i,1}^2 - d_{1,1}^2 = \mathbf{r}_i^T\mathbf{r}_i - 2\mathbf{r}_i^T\mathbf{s}_1 = \tilde{\mathbf{R}}_i^T(\mathbf{L}^T\mathbf{L})^{-1}\tilde{\mathbf{R}}_i - 2\mathbf{b}\tilde{\mathbf{R}}_i. \tag{4.51}$$

Let $\mathbf{H} = (\mathbf{L}^T\mathbf{L})^{-1}$ be a symmetric matrix with 6 unkonwns, and $\mathbf{b}$ be a vector with 3 unknowns:

$$\mathbf{H} = \begin{bmatrix} z_1 & z_2 & z_3 \\ z_2 & z_4 & z_5 \\ z_3 & z_5 & z_6 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} z_7 \\ z_8 \\ z_9 \end{bmatrix}$$

Then

$$d_{i,1}^2 - d_{1,1}^2 = \tilde{\mathbf{R}}_i^T\mathbf{H}\tilde{\mathbf{R}}_i - 2\mathbf{b}\tilde{\mathbf{R}}_i. \tag{4.52}$$

To solve for the unknowns $z_k$, let $\mathbf{r}_i = (r_i^x, r_i^y, r_i^z)^T$. Rewrite $\mathbf{H}$ to

$$\mathbf{H} = \sum_{k=1}^{6} z_k\mathbf{B}_k, \tag{4.53}$$

where $\mathbf{B}$ are base matrices for $\mathbf{H}$. Then expand the right-hand side of equation (4.52) into

$$\tilde{\mathbf{R}}_i^T\mathbf{H}\tilde{\mathbf{R}}_i = \sum_{k=1}^{6} z_k(\tilde{\mathbf{R}}_i^T\mathbf{B_k}\tilde{\mathbf{R}}_i), \tag{4.54}$$

and

$$-2\mathbf{b}\tilde{\mathbf{R}}_i = -2(r_i^x z_7 + r_i^y z_8 + r_i^z z_9). \tag{4.55}$$

Each product $\tilde{\mathbf{R}}_i^T\mathbf{B_k}\tilde{\mathbf{R}}_i$ in equation (4.54) is just a scalar and the following equation is derived from equation (4.52), using equations (4.54) and (4.55):

$$\underbrace{\begin{bmatrix}((r_i^x)^2 & 2r_i^xr_i^y & 2r_i^xr_i^z & (r_i^y)^2 & 2r_i^yr_i^z & (r_i^z)^2 & -2r_i^x & -2r_i^y & -2r_i^z)\end{bmatrix}}_{=A_{(M-1)\times 9}} \begin{bmatrix} z_1 \\ \vdots \\ z_9 \end{bmatrix} = \underbrace{\begin{bmatrix}(d_{i,1} - d_{1,1})\end{bmatrix}}_{=C_{(M-1)\times 1}}, \tag{4.56}$$

for $i = 2, \dots M$. Using the MATLAB backslash operator it is possible to solve equation (4.56) with $z = A \setminus C$. Reconstruct $\mathbf{H}$ and $\mathbf{b}$ using the calculated $z$. Then compute $\mathbf{L}$ using Cholesky factorization

$$\mathbf{L} = \text{chol}\,(\mathbf{H}^{-1}). \tag{4.57}$$

Using the paramterizations it is now possible to reconstruct $\mathbf{R} = \mathbf{L}^{-T}\tilde{\mathbf{R}}$ and $\mathbf{S} = \mathbf{L}(\tilde{\mathbf{S}} + \mathbf{b})$. Thus the positions of all the receivers $\mathbf{r}_i$ and the sound events $\mathbf{s}_j$ have been estimated.

### 4.3.5 Bundle adjustment

With $\mathbf{r}_i, \mathbf{s}_j, o_j$ and $u_{i,j}$ known it is possible to optimize $\mathbf{r}_i, \mathbf{s}_j$ and $o_j$ a little bit further. Construct the equation

$$\min_{\mathbf{r}_i, \mathbf{s}_j, o_j} \sum_{i,j} (u_{i,j} - (||\mathbf{r}_i - \mathbf{s}_j|| + o_j))^2. \tag{4.58}$$

This is a non-linear least squares optimization problem where the sum of squared distances between the TDOA values $u_{i,j}$ and the estimated values $\mathbf{r}_i$, $\mathbf{s}_j$ and $o_j$ are minimized. It can be locally optimized using techniques such as the iterative Levenberg-Marquardt algorithm [34, 35] with the estimated $\mathbf{r}_i$, $\mathbf{s}_j$ and $o_j$ as initial parameters. This will yield the maximum likelihood estimation of the parameters.

Let $f_{i,j}(\beta_{i,j}) = ||\mathbf{r}_i - \mathbf{s}_j|| + o_j$ where $\beta_{i,j}$ is a parameter pack containing $\mathbf{r}_i$, $\mathbf{s}_j$ and $o_j$. Then the adjustment step $\boldsymbol{\delta}$ is computed for each iteration of the algorithm using the formula

$$(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})\boldsymbol{\delta} = \mathbf{J}^T(\mathbf{f}(\boldsymbol{\beta}) - \mathbf{u}), \tag{4.59}$$

where $\mathbf{J}$ is a *Jacobian matrix* with components $\mathbf{J}_i$ and $\mathbf{f}(\boldsymbol{\beta})$ is a matrix with components $f_{i,j}(\beta_{i,j})$. Here $\mathbf{J}_i$ is the gradient of $f_{i,j}$ with respect to $\beta_{i,j}$. The sparsity of the top left corner of $\mathbf{J}$ can be seen in fig. 4.1.

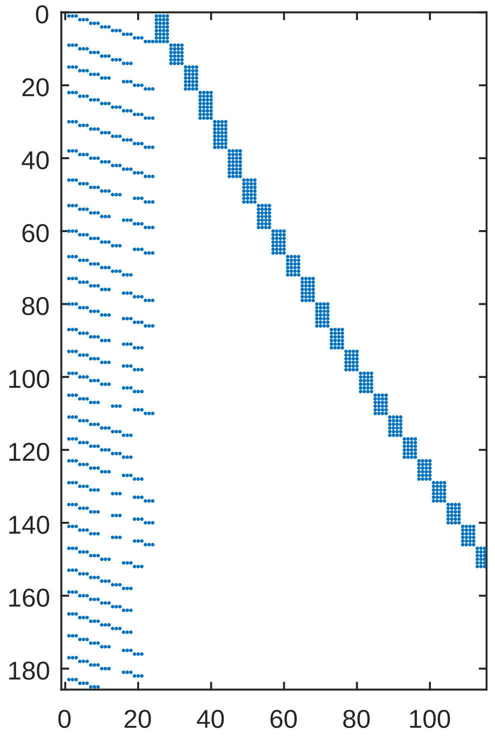


Figure 4.1: The sparsity of the top-left corner of $\mathbf{J}$. The rest of $\mathbf{J}$ extends in a similar pattern.

## 4.4 RANSAC: Random Consensus Sampling

Most mathematical methods have a number of parameters that need to be estimated from some observed data. The data can be measurements of a set of points or sound levels for example. Some of the data may be good and relevant to the model while some of the measurements may be noise or extreme values. These two categories are usually called *inliers* and *outliers*, representing “good values” and “bad values”.

A number of different ways to estimate the model parameters from said data exist. In this thesis *Random Consensus Sampling* (RANSAC) will be explored further. Other estimators include M-estimators of which least-squares estimators are a subclass. However no other estimators have been explored due to the prevalent use of RANSAC in the papers this thesis is based upon.

The RANSAC algortihm was first developed by Fischler and Bolles [17] and has since been improved. It is an iterative method that minimizes the effect of outliers on the parameter estimation by not taking them into account. Advantages of using RANSAC are that the algorithm can detect outliers with a high precision even when their numbers are significant, in other words it is a robust estimator. A downside is that it has no upper bound time limit if the aim is to find an optimal solution.

The outline of the RANSAC algorithm is:

1. Estimate the parameters of the model with as few data points as possible. The subset of data points is selected at random among all of the data points.

2. With the model parameters estimated, compute error residuals using the rest of the data points. The *consensus set* is defined as the data points for which the error residuals are less than a predefined threshold.

3. Repeat step 1 and 2 for a given number of iterations and then choose the model parameters that yield the largest consensus set. The consensus set then consists of the inliers to the problem.

An illustrative example of the algorithm is the problem of fitting a line to a set of points $(x, y)$ in a two-dimensional space $\mathbb{R}^2$. A line is described by $y = ax + b$ with $a$ being the slope of the line and $b$ the offset.

First $a$ and $b$ are estimated using a subset of the points (this can be as few as two points). The error residual is chosen as the distance between the line and point being evaluated. So step two consists of computing distances from the line to the remaining points and then the score is given by the number of distances below a certain value. The score is assigned to the parameters $a$ and $b$. These two steps are repeated until the upper bound of iterations is reached. The model parameters with the highest score are then selected as the best fit.

## 4.5 Procrustes superimposition

A way to compare two different but similar objects is to superimpose one of the objects onto the other object. To superimpose means to place one thing onto another in such a way that they “overlap” as much as possible. In a perfect world the result of a superimposition would be that the two objects are translated, rotated and uniformly scaled in such a way that they both occupy exactly the same space. One way to achieve this is to use *Procrustes superimposition* which will be derived in this section.

Let $\mathbf{A}$ and $\mathbf{B}$ be two $K \times M$ point matrices with row vectors $\mathbf{A}_i$ and $\mathbf{B}_i$, $i = 1, \dots, K$. In other words each matrix represent $M$ points of dimension $K$. The task at hand is to superimpose $\mathbf{A}$ onto $\mathbf{B}$.

Begin by moving the matrices centres of mass to origo in order to remove the translational component. The means are defined by

$$\begin{aligned} \overline{\mathbf{A}}_i &= \frac{a_{i,1} + a_{i,2} + \ldots + a_{i,M}}{M}, \\ \overline{\mathbf{B}}_i &= \frac{a_{i,1} + a_{i,2} + \ldots + a_{i,M}}{M}, \end{aligned} \tag{4.60}$$

and resulting translated matrices are $\hat{\mathbf{A}} = \mathbf{A} - \overline{\mathbf{A}}$ and $\hat{\mathbf{B}} = \mathbf{B} - \overline{\mathbf{B}}$.

The next step is to remove the scale component by uniformly rescaling $\hat{\mathbf{A}}$ to $\hat{\mathbf{B}}$. The scaling factor $s$ can be calculated by

$$s = \frac{||\hat{\mathbf{A}}||_2}{||\hat{\mathbf{B}}||_2}. \tag{4.61}$$

The final step is to compute the rotation factor so that $\hat{\mathbf{A}}$ is aligned to $\hat{\mathbf{B}}$. This is a complex task called *Orthogonal Procrustes problem* where the goal is to calculate an orthogonal matrix $\mathbf{R}$ that maps $\hat{\mathbf{A}}$ to $\hat{\mathbf{B}}$ as closely as possible, i.e.

$$\mathbf{R} = \underset{\Omega}{\operatorname{argmin}} ||\Omega\hat{\mathbf{A}} - \hat{\mathbf{B}}||_F \tag{4.62}$$

such that $\Omega^T\Omega = \mathbf{I}$.

A solution can be found in Schönemann [36]. Let $\mathbf{M} = \hat{\mathbf{B}}\hat{\mathbf{A}}^T$. Finding the nearest orthogonal matrix to $\mathbf{M}$ is equivalent to solving the problem above. The singular value decomposition of $\mathbf{M}$ can be written as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{4.63}$$

The solution to the problem is

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T. \tag{4.64}$$

Putting all this together yields an expression for superimpositioning $\mathbf{A}$ onto $\mathbf{B}$

$$\widetilde{\mathbf{A}} = \frac{1}{s}\mathbf{R}\hat{\mathbf{A}} + \hat{\mathbf{B}}, \tag{4.65}$$

where $\widetilde{\mathbf{A}}$ is the matrix resulting from the superimposition of $\mathbf{A}$. Note the term $\overline{\mathbf{B}}$ which is added at the end in order to translate the superimpositioned points back to the original position of $\mathbf{B}$.

In order to use the superimposition for more points than the ones in $\mathbf{A}$ (this can be the case if $\mathbf{A}$ is a subset of a larger matrix that happened to have a known mapping in $\mathbf{B}$ for $\mathbf{A}$) an alternative form is computed.

Expand eq. (4.65) to

$$\begin{aligned}\widetilde{\mathbf{A}} &= \frac{1}{s}\mathbf{R}(\mathbf{A} - \overline{\mathbf{A}}) + \hat{\mathbf{B}} \\ &= \frac{1}{s}\mathbf{R}\mathbf{A} - \frac{1}{s}\mathbf{R}\overline{\mathbf{A}} + \hat{\mathbf{B}},\end{aligned} \tag{4.66}$$

and collect the terms as

$$\mathbf{Q} = \frac{1}{s}\mathbf{R} \quad \text{and} \quad \mathbf{T} = \hat{\mathbf{B}} - \frac{1}{s}\mathbf{R}\overline{\mathbf{A}}, \tag{4.67}$$

so that $\widetilde{\mathbf{A}} = \mathbf{Q}\mathbf{A} + \mathbf{T}$. Then this formula can easily be used to compute the superimposition for other points than those in $\mathbf{A}$.

## 4.6 Basic geometry for planes and lines

In this section some basic definitions and formulas for point, line and plane geometry in three dimensions are introduced. First recall two important linearly algebraic definitions.

The *dot product* between two vectors $\mathbf{u} = (u_1, \ldots, u_n)$ and $\mathbf{v} = (v_1, \ldots, v_n)$ is given by

$$\mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^{n} u_i v_i. \tag{4.68}$$

The *projection* of a vector $\mathbf{u}$ onto a vector $\mathbf{v}$ is given by

$$P_{\mathbf{v}}(\mathbf{u}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{v}||^2}\mathbf{v}. \tag{4.69}$$

### 4.6.1 Definition of a line $L$

A line $\mathbf{L}$ in $\mathbb{R}^3$ can be described by a point $\mathbf{p}$ and a vector $\mathbf{v}$

$$\mathbf{L}(t) = \mathbf{p} + t\mathbf{v}, \tag{4.70}$$

for some scalar $t$. The vector $\mathbf{v}$ is the direction of the line. The line $\mathbf{L}$ passes through the point $\mathbf{p}$. For two points $\mathbf{p}$ and $\mathbf{q}$ the line $L_{p,q}$ is then described by

$$\mathbf{L}_{p,q}(t) = \mathbf{p} + t(\mathbf{q} - \mathbf{p}). \tag{4.71}$$

Alternatively the line $\mathbf{L}$ can be described as the intersection between two planes, where $(x, y, z) \in \mathbf{l}$ are given by

$$\{(x, y, z) \in \mathbb{R}^3 : a_1x + b_1y + c_1z + d_1 = 0 \text{ and } a_2x + b_2y + c_2z + d_2 = 0\}. \tag{4.72}$$

### 4.6.2 Projection of a point $p$ onto a line $L$

Given two points $\mathbf{q}_1$ and $\mathbf{q}_2$ the equation for a line $\mathbf{L}$ is

$$\mathbf{L}(t) = \mathbf{q}_1 + t\mathbf{v}. \tag{4.73}$$

as described by eq. (4.71), where $\mathbf{v} = \mathbf{q}_2 - \mathbf{q}_1$. To project a point $\mathbf{p}$ onto $\mathbf{L}$ the projection formula 4.69 is used. The difference it that the "offset" for the line has to be taken into account. This yields the projection forumla

$$P_{\mathbf{L}}(\mathbf{p}) = \mathbf{q}_1 + \frac{(\mathbf{p} - \mathbf{q}_1) \cdot \mathbf{v}}{||\mathbf{v}||^2}\mathbf{v}. \tag{4.74}$$

### 4.6.3 The intersection between two lines $L_1$ and $L_2$

Given two lines $\mathbf{L}_1$ and $\mathbf{L}_2$ constructed from two point pairs $(\mathbf{a}, \mathbf{b})$ and $(\mathbf{c}, \mathbf{d})$ the two line equations become

$$\mathbf{L}_1 = \mathbf{a} + t(\mathbf{b} - \mathbf{a}) \quad \text{and} \quad \mathbf{L}_2 = \mathbf{c} + s(\mathbf{d} - \mathbf{c}). \tag{4.75}$$

An intersection is found if and only if $\mathbf{L}_1 = \mathbf{L}_2$ has a solution $(t, s)$. In a solution $(t_0, s_0)$ the value of $t_0$ is the displacement from point $\mathbf{a}$ on the line $\mathbf{L}_1$ and similarly for $s_0, \mathbf{c}$ and $\mathbf{L}_2$.

### 4.6.4 Definition of a plane $\pi$

A plane $\pi$ in $\mathbb{R}^3$ can be described by the equation

$$ax + by + cz + d = 0, \tag{4.76}$$

for some parameters $a, b, c$ and $d$. The normal $\mathbf{n} = (a, b, c)$ of the plane is by definition perpendicular to the plane. A plane whose normal is $\mathbf{n}$ and passes through a point $\mathbf{p} = (x_0, y_0, z_0)$ can be calculated by solving the equation

$$a(x - x_0) + b(y - y_0) + c(z - z_0) = 0. \tag{4.77}$$

Using the dot product, it is then simple to calculate $d = -\mathbf{n} \cdot \mathbf{p} = -(ax_0 + by_0 + cz_0)$.

### 4.6.5 Normal of a plane $\pi$

The normal of a plane $\pi$ can be computed in a few different ways. One way is to use the cross product. Given three points $\mathbf{p}, \mathbf{q}$ and $\mathbf{r}$, all in $\mathbb{R}^3$, construct the vectors $\mathbf{u} = \mathbf{p} - \mathbf{q}$ and $\mathbf{v} = \mathbf{p} - \mathbf{r}$. These two vectors will be linearly independent if $\mathbf{q} \neq \mathbf{r}$. The definition of the cross product says that the result of the cross product $\mathbf{a} \times \mathbf{b}$ will be perpendicular to $\mathbf{a}$ and $\mathbf{b}$.

Thus the normal $\mathbf{n} = (a, b, c)$ can be calculated as

$$\mathbf{n} = \mathbf{u} \times \mathbf{v}. \tag{4.78}$$

since $\mathbf{u}$ and $\mathbf{v}$ spans a two-dimensional plane.

### 4.6.6 Projection of a point $p$ onto a plane $\pi$

A plane $\pi$ can be defined by a normal $\mathbf{n}$ and point $\mathbf{p}_\pi$ as shown in section 4.6.4. To project a point $\mathbf{p}$ onto $\pi$ the projection formula in eq. (4.69) is used. The difference it that the point $\mathbf{p}_\pi$ has to be taken into account.

This yields the projection forumla

$$P_\pi(\mathbf{p}) = \mathbf{p}_\pi + \frac{(\mathbf{p} - \mathbf{p}_\pi) \cdot \mathbf{n}}{||\mathbf{n}||^2}\mathbf{n}. \tag{4.79}$$

### 4.6.7 Distance between a point $p$ and a plane $\pi$

The shortest distance between a point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ and a plane $\pi$ is along a perpendicular line to the plane, parallel to the normal. Here the unit normal is used, defined as

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{||\mathbf{n}||} = \frac{(a, b, c)}{\sqrt{a^2 + b^2 + c^2}}. \tag{4.80}$$

Let $\mathbf{q} = (x_\pi, y_\pi, z_\pi)$ be an arbitrary point on the plane and form the vector $\mathbf{v} = \mathbf{p} - \mathbf{q}$. The distance $D$ is then the length of the projection of $\mathbf{v}$ onto $\hat{\mathbf{n}}$. Since $\hat{\mathbf{n}}$ is of length one, this is equivalent to the absolute value of the dot product between $\mathbf{v}$ and $\hat{\mathbf{n}}$,

$$D = |\mathbf{v} \cdot \hat{\mathbf{n}}|. \tag{4.81}$$

Recall from eq. (4.77) that for a point $\mathbf{r} = (x_0, y_0, z_0)$ belonging to a plane with normal $\mathbf{n}$, parameter $d$ from the plane eq. (4.76) is $d_\mathbf{r} = -(ax_0 + by_0 + cz_0)$. Rewrite $D$ as

$$\begin{aligned} D &= \frac{|a\mathbf{v} + b\mathbf{v} + c\mathbf{v}|}{\sqrt{a^2 + b^2 + c^2}} \\ &= \frac{|a(x - x_\pi) + b(y - y_\pi) + c(y - y_\pi)|}{\sqrt{a^2 + b^2 + c^2}} \\ &= \frac{|ax + by + cz - (ax_\pi + by_\pi + cz_\pi))|}{\sqrt{a^2 + b^2 + c^2}}, \end{aligned} \tag{4.82}$$

and it is easy to see that $-(ax_\pi + by_\pi + cz_\pi)$ can be substituted for $d_\pi$ as $\mathbf{q} \in \pi$. Thus the final equation after substituting is

$$D = \frac{|ax + by + cz + d_\pi|}{\sqrt{a^2 + b^2 + c^3}}. \tag{4.83}$$

Chapter 5

# System Implementation

*In this chapter the system design, theory and techniques from the previous chapters are put together into an actual implementation.*

## 5.1 Generation and separation of signals

A simple signal $y(t)$ can be generated by a sine wave as

$$y(t) = A\sin(2\pi f t + \theta), \tag{5.1}$$

where $y(t)$ is the signal at a time $t$, $A$ is the amplitude, $f$ is the frequency and $\theta$ is the phase.

For practical implementations this means that the amplitude $A$ controls the volume of the output signal, and lie within the interval $[0, 1]$. The frequency $f$ will control the tone of the output signal. For example, $f = 440$ Hz will correspond to the musical tone A. The phase $\theta$, which controls where the wave will be at $t = 0$ will not be used and is therefore set to zero.

As the system uses actual recordings the signals will be sampled by the audio processor, hence signals are discrete and not continuous. The sampling frequency $F_s$ has to be at least twice the size of $f$ [14], i.e. $F_s \geq 2f$, in order to avoid aliasing. The audio will also arrive from the audio processor in blocks of $n_B$ sample points. Therefore eq. (5.1) can be rewritten as a sampled signal

$$y[o, x_1, x_2, \ldots, x_{n_B}] = A\sin\left(2\pi\frac{f}{F_s}(o + [x_1, \ldots, x_{n_B}])\right), \tag{5.2}$$

where $x_1, \ldots, x_{n_B}$ represent the $n_B$ sample points in the current block, located at an offset $o$ from the first sample point at $t = 0$.

### 5.1.1 The different signal methods

It was hypothesized in this thesis that a plain sinusoid would not yield the best results when running the GCC-PHAT on a recording. To experiment with this, four different signal types were constructed, namely

- a Continuous Level signal method,
- a Periodic Level signal method,
- a Periodic Scaled signal method, and
- a Stacked Periodic Random signal method.

**The Continuous Level signal method (CL)**

The Continuous Level signal method is simply the signal defined in eq. (5.2).

### The Periodic Level signal method (PL)

The Periodic Level signal method emits short audio pulses and is therefore not continuous. The pulses are controlled by a pulse length $P_l$, measured in seconds, and the number of pulses per second $P_n$.

Let the number of sample points in a pulse be $s_p = P_l F_s$. Given an offset $o$, the offset in the current pulse can be calculated $o_p = o \mod \frac{F_s}{P_n}$. To determine if the start of a block lies within a pulse, compute $C = s_p - (o_p + n_b)$. Initialize $y[o_p, x_1, x_2, \ldots, x_{n_B}]$ as a zero vector of length $n_B$. If $C > n_B$ then the signal equation is given by

$$y[o_p, x_1, x_2, \ldots, x_{n_B}] = A \sin\left(2\pi \frac{f}{F_s}(o + [x_1, \ldots, x_{n_B}])\right). \tag{5.3}$$

Otherwise, if $C < n_B$ and $C \geq 0$ the signal equation is given by

$$y[o_p, x_1, x_2, \ldots, x_C] = A \sin\left(2\pi \frac{f}{F_s}(o + [x_1, \ldots, x_C])\right). \tag{5.4}$$

If neither of the two conditions above are true, the signal remains zero, indicating that it lies between two pulses.

In this thesis the values of $P_l$ and $P_n$ were selected after experimenting: GCC-PHAT images were generated for different combinations of $P_l$ and $P_n$ values. The values yielding the most pristine image were chosen, namely $P_l = 0.2$ seconds and $P_n = 3$ pulses per second.

### The Periodic Scaled signal method (PS)

The Periodic Scaled signal method has a varying frequency within a given frequency band. It is called scaled because the frequency varies between $f_{\min}$ and $f_{\max}$ in a musical scale instead of having just a static frequency $f$. The only difference from the PS signal method is that $f$ is computed instead of given.

Given a base frequency $f_{\min}$, the width $w$ of the frequency band $[f_{\min}, f_{\max}]$ and a scale $\mathbf{m}$ containing a number of values in the interval $[0, 1]$ the frequency $f$ can be computed. Let

$$z = \left(\frac{o}{s_p} \mod n_m\right) + 1, \tag{5.5}$$

where the $n_m$ is the length of $\mathbf{m}$. The frequency is then given by

$$f = f_{\min} + w m_z. \tag{5.6}$$

In this thesis the width of the frequency span $[f_{\min}, f_{\max}]$ was set to $w = 1000$ Hz and the scale $\mathbf{m}$ was the ratios for the musical tones in the C Major scale, namely

$$\mathbf{m} = [0.0000, 0.1224, 0.2599, 0.3348, 0.4983, 0.6818, 0.4983, 0.3348, 0.2599, 0.1224].$$

By doing experiments in the same way as for the PL signal method, the values $P_l = 0.3$ seconds and $P_n = 3$ pulses per second were established as the best combination.

### The Stacked Periodic Random signal method (SPR)

The Stacked Periodic Random signal method is the one that deviates most from the other signal methods. It consists of a number of damped sine waves, $n_s$, stacked on top of each other. Each sine wave will have a random frequency $f$ in the interval $[f_{\min}, f_{\max}]$. The dampening coefficient is denoted $d$.

The different sine waves will be stored in a list $\mathbf{P}$ containing up to $n_s$ elements. Only one damped sine wave will be added for each incoming audio block and thus the SPR has a windup effect. Let the frequency of a sine wave be

$$f = f_{\min} + w\gamma, \tag{5.7}$$

where $\gamma$ is a random number in the interval $[0, 1]$. Then, for each incoming audio block, check if the size of $\mathbf{P} \geq n_s$. If not, add a new sine wave to $\mathbf{P}$ with the offset $o$ as the starting sample point.

Initialize $y[o_p, x_1, x_2, \ldots, x_{n_B}]$ as a zero vector of length $n_B$. To generate the signal, iterate over all the elements $P_i$ of $\mathbf{P}$. For each iteration, set $o_p = o - o_i$. If $o_p \geq P_l$, remove $P_i$ from the list as it has been phased out. Otherwise calculate the signal part

$$y_i[o_p, x_1, x_2, \ldots, x_{n_B}] = A \sin\left(2\pi \frac{P_i}{F_s}(o_p + [x_1, \ldots, x_{n_B}])\right) e^{d(o_p + [x_1, \ldots, x_{n_B}])}. \tag{5.8}$$

The signal part $y_i$ is then added to the signal $y[o_p, x_1, x_2, \ldots, x_{n_B}]$. Finally the signal is normalized to the interval $[-1, 1]$.

The frequency width $w$ was kept at 1000 Hz. By doing experiments similar to the previous ones, the combination of values $n_s = 10$, $P_l = 0.1$ and $d = -0.005$ were found to be the best.

### 5.1.2 Separation of multiple signals

When the recording contains multiple signals they need to be separated before the system can be run. This is done by using a bandpass filter that only allows a specific frequency band to pass. The bandpass filter was designed with the Signal Processing Toolbox 7.0 in Matlab, using a Parks-McClellan optimal FIR filter order estimation.

The frequency bands $[f_{\min}, f_{\max}]$ in the SPR signal method have the width $w = 1000$ Hz. The edges of the frequency band (defined by two values) are set to

$$F = \frac{2}{F_s}((f_{\min} - 100, f_{\min} + 100), (f_{\max} - 100, f_{\max} + 100)).$$

The amplitudes outside the frequency band edges should be zero, whilst it should be one between them.

Finally the Matlab implementation also requires a vector $D$ that, to cite the Matlab reference manual, "specifies the maximum allowable deviation or ripples between the frequency response and the desired amplitude of the output filter for each band" and a density factor $d_f$ that was set to 20. The vector $D$ was designed with the toolbox and set to

$$D = (0.000177827941, 0.0057563991496, 0.000177827941).$$

The filter function is designed as in algorithm 1.

**Algorithm 1** Creating the bandpass filter with the MATLAB Signal Processing Toolbox 7.0

```
1: $[N, F_o, A_o, W] = \textsc{firpmord}(F, [0\ 1\ 0], D)$
2: $b = \textsc{firpm}(N, F_o, A_o, W, \{d_f\})$
3: $Hd = \text{dfilt.dffir}(b)$
```

## 5.2 System self-calibration

In order to find the positions of the receivers a scene reconstruction needs to be done. This is a process involving many steps which are described in this section. In the end this will result in two matrices.

- One $3 \times M$ matrix containing the positions of the receivers within the recording.
- One $3 \times N$ matrix containing the positions of the sound events within the recording.

A virtual screen will also be fitted to the second matrix and presented as four corner points $c_k$.

### 5.2.1 Estimation of the TDOA values $u_{i,j}$

The first step in the scene reconstruction is to get a good estimation of the TDOA values for each sound event in each channel of the recording. The simplest solution would be to use the maximum value of each use of the GCC-PHAT and use these values for the remainder of the scene reconstruction. However this will not generate a smooth path of sound events later on since the reverberations, reflections and background has to be taken into consideration. By chopping the continuous signal into smaller time frames a smooth path would also be difficult be obtain because of the discretization.

Most of the work here is based on the work done by Segerblom Rex [11]. The main idea is to use the inital values from the GCC-PHAT and then reduce the number of outliers while keeping the number of inliers high. When a good enough path is found it is smoothed to yield a nice path.

The algorithm described below is run once for each channel pair between the reference channel and the other channels. The steps are:

1. **GCC-PHAT.** Use GCC-PHAT to get all possible initial TDOA values from the channel pairs.

2. **Peak selection.** Find peaks within the resulting GCC-PHAT matrix for each time frame. Then run a refining algorithm using the data from all the other channel pairs.
3. **Fit tracklets with RANSAC.** Place the points into different groups and fit a line to each point group using RANSAC to create a number of *tracklets*.
4. **Connect tracklets.** Connect a number of tracklets to form a *tracklet chain*.
5. **Smoothen and fit a spline.** Smoothen the tracklet chain and fit a spline to the original peaks to fill the gaps and create a continuous path of sound events.
6. **Convert to metric system.** Finally the estimated TDOA values are converted from sample points to meters.

It is also noteworthy that each channel pair may contain multiple paths (tracklets chains) since there are reflections and aliasing of the signal. The strongest path should correspond to the direct path between the sound source and the receiver.

### Step 1: GCC-PHAT

In order to estimate the initial TDOA values of $\mathbf{u}$, for the different channel pairs the signals need to be processed using GGC-PHAT. The matrix $\mathbf{u}$ is usually displayed as a greyscale heatmap where the different paths easily can be seen and interpreted. The heatmap is referred to as the GCC-PHAT image.

Assuming that the sound source is moving, the signal needs to be divided into a number of time frames, the reason being that GCC-PHAT needs to be applied to a time frame small enough to consider the sound source stationary (and hence becoming a sound event). The bigger time the frames, the less stationary the sound sources become within a time frame.

To decide the width of the time frames, boundary conditions need to be set. Here it is assumed that the sound source is moving at $v_{\text{ss}} = 1$ m/s. This might be a tad slow for a hand moving across a normal $22''$ computer screen, but it is an estimation good enough. The next thing that needs to be decided is how big the spatial error can be. With a search space of about 1 m$^3$ an error of about $\epsilon = \pm 2$ cm can be tolerated without too much user impact.

With a sampling rate $F_s = 48000$ Hz and an error $\epsilon = 0.04$ m the time frame width should be

$$w_{\text{gcc}} = \frac{F_s \cdot \epsilon}{v_{\text{ss}}} = \frac{48000 \cdot 0.04}{1} = 1920. \tag{5.9}$$

For practical reasons the frame width should be a power of two in order to speed up the computations in the Fast Fourier Transform. Hence $w_{\text{gcc}} = 2048$ is chosen. An overlap of 1048 sample points between the frames will be used to make the movement of the sound source linear when considering adjacent frames. This will put the frames 1000 sample points apart.

Running the GCC-PHAT results in a GCC-PHAT matrix $\mathbf{u}$ with elements $u_{i,j}$ which contains TDOA values measured in sample points.

### Step 2: Peak selection

The goal in this step is to find a number of peaks in each time frame of the GCC-PHAT matrix $\mathbf{u}$. The points are hopefully inliers to the path of TDOA values. The maximum number of peaks per time frame are here denoted $Q$. In this thesis $Q = 4$ was chosen as a good fit. However, not every frame has $Q$ distinct peaks and thus a minimum energy threshold at 0.01 was chosen to avoid outliers).

First all the local maxima in each time frame are located. After that the $Q$ top local maxima above the minimum energy threshold are selected as points of interest. By using the local maxima instead of the highest valued peaks, peaks from different paths are hopefully located. The result of the initial peak selection can be found on the left-hand side in fig. 5.1.

In order to remove more outliers, the peaks are refined. This is done by using the data from the remaining channel pairs (which in turn also has up to $Q$ points for each time frame). Hence the initial peak selection must be run for all channel pairs before the refinement step.

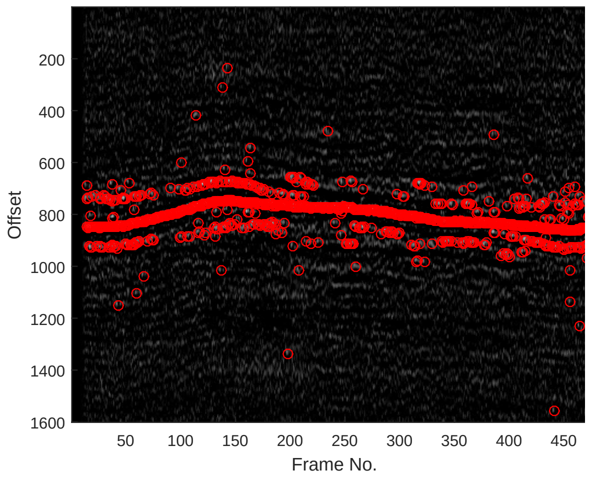
The reason this is possible is because $u_{i,j}$ from equation (2.9) can also be rewritten as the difference between two other receivers. Rewrite $u_{i,j} = {}^{q}_{p}u_j$, denoting the TDOA between receivers $r_p$ and $r_q$ from a sound source $s_j$. Then the following is true:

$$ {}^{q}_{p}u_j = {}^{p}_{k}u_j - {}^{q}_{k}u_j, \qquad k \in \{1, \dots, M\} \setminus \{p, q\}. \tag{5.10}$$

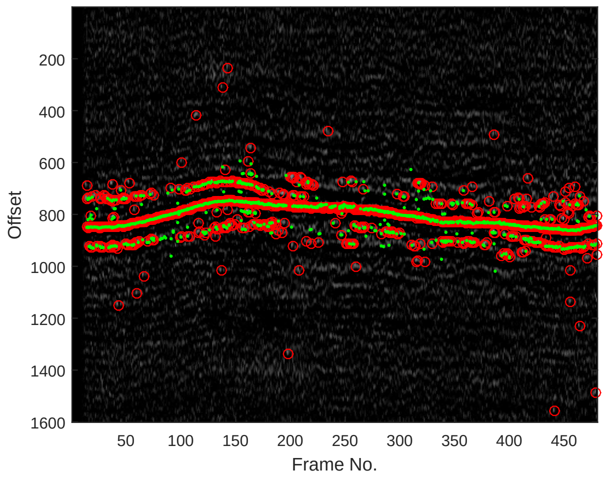
This results in $M - 2$ new equations for $u_{i,j}$ where each equation will have up to $Q \cdot Q$ inliers. Then, together with the original $Q$ inliers for $u_{i,j}$, each frame will contain up to $Q + Q^2(M - 2)$ inliers in total.

All the available inliers for a time frame are then sorted into a number of bins. The width of the bins was set to five sample points. For all bins with at least two inliers, a median is calculated using the values of the points in the bin. The newly calculated medians are used as new inliers, replacing the old ones.

A comparison between the initial inliers and the refined inliers can be seen in fig. 5.1.



(a) The initial peak selection (red circles) with up to $Q = 4$ peaks in each time frame.



(b) Refined points (green dots) using the medians from different bins as new peaks.

Figure 5.1: A comparison between the initial peaks from the local maxima and the refined inliers.

### Step 3: Fit tracklets with RANSAC

When a good selection of inlier peaks are found it is time to fit a number of short lines to them. These short lines are called *tracklets*. The reason for using a number of short tracklets instead of fitting a curve directly to all the peaks is that the peaks can belong to many different paths of TDOA values in the GCC-PHAT matrix **u**. By using the strengths of RANSAC more robust paths can be found.

First all of the inliers from the previous step are rearranged into a new number of frames called *RANSAC frames*. A RANSAC frame width of 15 time frames were used, with each RANSAC frame overlapping the neighbouring frames with 3 time frames. The RANSAC scheme is described below. The algorithm was run for a maximum of 350 iterations for each RANSAC frame.

- Instead of finding a single tracklet for a given RANSAC frame, multiple tracklets were computed. The tracklets chosen were the ones with the highest number of inliers. In this thesis up to 4 tracklets were selected for each frame and the tracklets within one frame could share a maximum of 4 inliers with the other tracklets in the same frame (to avoid multiple tracklets stacked on top of each other).
- The error residual were chosen as the distances from the points to the computed tracklet. The threshold for this was set to 3.5 sample points.
- A threshold for the slope of a tracklet, as well as a minimum number of inliers for a tracklet, was implemented. The slope threshold was set to 2.5 and the minimum number of inliers was set to 4.

A number of tracklets for each RANSAC frame in a recording is shown in fig. 5.2.

### Step 4: Connect tracklets

The newly computed tracklets now need to be connected in order to form a *tracklet chain*, which essentially is a number of 2D points stretching from the left side to the right side of the GCC-PHAT image. The method for doing this differs significantly from Segerblom Rex [11].

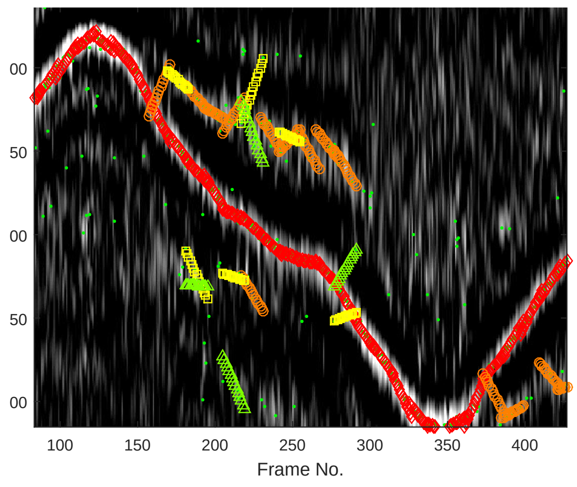


Figure 5.2: A number of tracklets (1 to 4) fitted to the peaks in each RANSAC frame.

To connect the tracklets, a depth first search was implemented recursively, using dynamic programming for efficiency. The output consists of a number of tracklet chains, each starting at a unique tracklet. The tracklet chains are then sorted by the number of tracklets in the chains. Finally the top candidate for each channel pair is returned as the best fit.

The algorithm works as follows (for each channel pair):

1. Construct a $N_F \times N_L$ tracklet chain matrix $C$, where $N_F$ is the number of RANSAC frames and $N_L$ is the maximum number of tracklets for each RANSAC frame. The goal is to fill each element in the matrix with a tracklet chain.

2. Iterate over the RANSAC frames $n_i \in \{1, \ldots, N_F\}$, and the tracklets $n_j \in \{1, \ldots, N_L\}$. Unless the tracklet chain at position $(n_i, n_j)$ in the tracklet chain matrix $C$ is already computed, do the following steps:

   (a) Call the recursive function for the position $(n_i, n_j)$.

   (b) If $n_i$ happens to refer to the same frame as previously, return an empty tracklet chain and go to *(2)*.

   (c) In this step the first thing to do is to set the *current tracklet*. This is done by checking if the tracklet at position $(n_i, n_j)$ exists. If it does, set the *current tracklet* to that tracklet. Otherwise search backwards through the RANSAC frames in the tracklet chain until an existing tracklet is found and set the tracklet as the *current tracklet*.

   Next, there exists an exit condition if there are no tracklets in frame $n_{i+1}$ to connect to. If that is the case proceed to step *(d)*.

   A comparison with each of the tracklets in the next frame $n_{i+1}$ is executed. For each such tracklet two different alternatives exist.

      i. The number of frames between the *current tracklet* and the tracklet in the next frame $n_{i+1}$ is below a certain threshold. In this thesis the threshold was set to 3 RANSAC frames.

      ii. The number of frames between the *current* and the tracklet in the next frame $n_{i+1}$ is above the threshold.

   In the case of alternative *(i)* the last three points (the overlapping points as defined in section 5.2.1) of the *current tracklet* and the first three points of the tracklet from $n_{i+1}$ are used to compute the mean vertical distance between the point pairs. If the distance mean is small enough the tracklet as a candidate for the *next tracklet* in the tracklet chain. Small enough in this case is a distance less than 25 sample points.

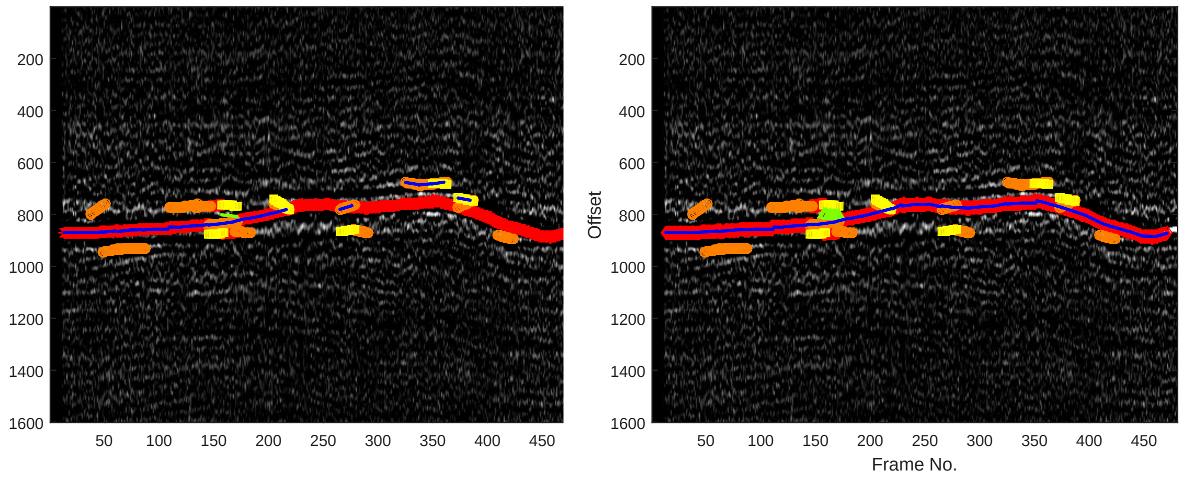
   In the case of alternative *(ii)* the same points as described in *(i)* are used to fit a line between them. If the number of inlier points within a certain distance from the line is small enough, add the tracklet is a candidate for the *next tracklet* in the tracklet chain. Points within a distance of 25 sample points were considered inliers and the ratio of inliers had to be above 0.04.

(d) Each potential *next tracklet* found in step *(d)* is now expanded by executing step *(a)* with $n_i = n_{i+1}$ and $n_j$ the value of the $j$-position of the *current tracklet*.

(e) When the expansion is complete compare all returned tracklet chains. The best tracklet chain is saved into the tracklet chain matrix $C$ at position $(n_i, n_j)$. The best tracklet chain is the one with the highest number of tracklets. If number of tracklets is equal between two chains, the result with the smallest mean distance between the overlapping points is chosen.

3. Sort the tracklet chain matrix $C$ by number of tracklets in each chain and return the tracklet chain with the highest number of tracklets.

For a comparison between the algorithm developed in Segerblom Rex [11] and the improved one, see fig. 5.3.



(a) Algorithm described in [11].

(b) Algorithm described in this thesis.

Figure 5.3: A comparison of the tracklet connection step of the algorithm described in this thesis and the algorithm described in Segerblom Rex [11]. Here the channel pair (2, 5) can be seen, where the best tracklet chain found is marked with blue dots. The different tracklets for each frame are shown in red, orange, yellow and green.

### Step 5: Smoothen and fit spline

The last step is to smooth the tracklet chain and then fit a spline to fill out potentially missing parts of the tracklet chain, i.e. RANSAC frames where no tracklets were found to be a good enough fit. In the implementation, the underlying points for the each tracklet in the tracklet chain are used instead of the points computed. The points closest to the values used in the tracklet are selected.

The smoothing is accomplished by using a moving average filter along the tracklet chain. When the smoothing is finished a $M \times N$ matrix with all the TDOA values $u_{i,j}$ between the reference receiver $\mathbf{r}_1$ and every other receivers for each sound event $\mathbf{s}_j$ will be exported. The row for the reference receiver, in this case the first row, will be zero as the TDOA values there will be non-existing.

### Step 6: Convert to metric system

The TDOA values $u_{i,j}$ are converted to meters from sample points using the formula

$$\mathbf{u}_{i,j}\frac{v_{ss}}{F_s}. \tag{5.11}$$

### 5.2.2 Estimate offsets $o_j$ using TDOA values $u_{i,j}$

This part is already implemented previously and can be found at [37]. This section explains briefly how the implementation is done.

Once the TDOA values $u_{i,j}$ are estimated, the offsets $o_j$ can be estimated. This is using a RANSAC scheme with the following steps:

1. Use a minimal solver defined in section 4.3.1 with $(M, N) = (7, 6)$. Solve for the unknown offsets $o_j$ in the subset $\hat{\mathbf{u}}$, consisting of 7 rows and 6 columns of $\mathbf{u}$. The subset represents 7 receivers $\mathbf{r}_i$ and 6 sound events $\mathbf{s}_j$. This results in up to five solutions. The minimal solver will use pre-computations if possible.

2. Explore each solution found. Valid solutions have real-valued offsets $o_j$. Extend the solution for the remaining rows using the rank-based non-linear optimization described in section 4.3.2. With the estimated matrix $\mathbf{A}$, another RANSAC scheme can be used to estimate the offsets for the remaining columns in $\mathbf{u}$ using the technique in section 4.3.3.

3. Repeat steps (1) and (2) and keep the solution with the maximum number of inliers among the calculated offsets $o_j$.

This RANSAC scheme is shown in greater detail, with actual values for each parameters, in algorithm 2.

**Algorithm 2** Estimate offsets $o_j$ using RANSAC and minimal solvers

```
Precondition: A $M \times N$ matrix $\mathbf{u}$ of TDOA values.
Precondition: The number of RANSAC iterations $n$.
Precondition: The RANSAC threshold $\xi$.

 1: function EstimateOffsetsFromTDOA($\mathbf{u}$, $n$, $\xi$)
 2:     $s_b \leftarrow$ A struct containing the best solution.
 3:     for $i \leftarrow 1$ to $n$ do
 4:         $\hat{\mathbf{u}} \leftarrow$ Random subset of 7 rows (always include the first row) and 6 columns of $\mathbf{u}$
 5:         $\hat{\mathbf{o}} \leftarrow$ MinimalOffsetSolver($\hat{\mathbf{u}}$)
 6:         if Length($\hat{\mathbf{o}}$) $< 0$ then
 7:             Continue with the next RANSAC iteration.

 8:         for Solution $\hat{o}$ in $\hat{\mathbf{o}}$, do
 9:             if $\hat{o} \notin \mathbb{R}$ then
10:                 Continue with next solution $\hat{o}$.

11:             $\hat{o} \leftarrow$ RankBasedNonLinearOptimization($\hat{\mathbf{u}}, \hat{o}$)        ▷ See section 4.3.2
12:             $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{u}}$ extended to all rows in $\mathbf{u}$
13:             $(\hat{o}, A) \leftarrow$ RankBasedNonLinearOptimization($\hat{\mathbf{u}}, \hat{o}$)

14:             $(U, S, V) \leftarrow$ SVD($A$)        ▷ Single value decomposition
15:             $A_b \leftarrow$ The first three columns of $U$        ▷ A basis for $A$

16:             $o \leftarrow$ A vector of length $N$
17:             for $j \leftarrow 1$ to $N$ do
18:                 $\mathbf{u}_j \leftarrow$ The $j$:th column of $\mathbf{u}$
19:                 $(x, o_c) \leftarrow$ EstimateOffsetForColumn($\mathbf{u}_j, A_b$)        ▷ See section 4.3.3
20:                 if Distance($A_b x, \mathbf{u}_j$) $< \xi$ then
21:                     $o_j \leftarrow o_c$
22:                 else
23:                     $o_j \leftarrow 0$

24:             if Length($o \neq 0$) >Length($s_b.o \neq 0$) then
25:                 $s_b.o \leftarrow o$
26:     return $s_b.o$
```

### 5.2.3 Estimate receiver positions $r_i$ and sound events $s_j$

This part is already implemented previously and can be found at [37]. This section explains briefly how the implementation is done.

With offsets $o_j$ and the TDOA values $u_{i,j}$ estimated, the positions of the receivers $\mathbf{r}_i$ and the sound events $\mathbf{s}_j$ can be estimated. This is done using the technique in section 4.3.4. Once this is completed the bundle adjustment technique in section 4.3.5 can be used to optimize the estimations for $\mathbf{r}_i, \mathbf{s}_j$ and $o_j$.

Finally a second optimization is done to find more inliers among the column vectors of $\mathbf{u}$. Form the problem

$$\min_{\mathbf{r}_i} \sum_{j=1}^{N} (u_{i,j} - ||\mathbf{r}_i - \mathbf{s}_j||_2), \tag{5.12}$$

and use trilateralation followed by a bundle adjustment to refine $\mathbf{r}_i$.

### 5.2.4 Fit a virtual screen

Once a good estimation for the sound events $\mathbf{s}_j$ is found, the virtual screen can be calculated. This is a process consisting of three steps where each step makes use of the results in the previous step. The result is a virtual screen which lies in a plane and is defined by its four corner points.

The steps are

1. Fit a plane to the sound events $\mathbf{s}_j \in \mathbf{S}$ using RANSAC.
2. Project the points in $\mathbf{S}$ onto the plane and fit four lines using RANSAC.
3. Compute the intersections between all lines and find the four intersections that are the best fit for corner points.

#### Step 1: Fit a plane to the sound events

To fit a two-dimensional plane to the sound events $\mathbf{s}_j$, RANSAC is used. The algorithm can be seen in algorithm 3 and uses the points of $\mathbf{S}$ which contains all the sound events $\mathbf{s}_j$. By experimenting it was found that $n = 400$ iterations and a threshold $\xi = 0.02$ meters were a good fit.

**Algorithm 3** Fit a plane using RANSAC

**Precondition:** $\mathbf{S}$ are points $\in \mathbb{R}^3$.
**Precondition:** The number of RANSAC iterations $n$.
**Precondition:** The RANSAC threshold $\xi$.

```
1:  function FitPlaneToPoints($\mathbf{S}_p$, $n$, $\xi$)
2:      $\pi \leftarrow$ A struct containing the plane parameters.
3:      for 1 to $n$ do
4:          $p \leftarrow$ Random point from $\mathbf{S}$
5:          $q \leftarrow$ Random point from $\mathbf{S}$
6:          $r \leftarrow$ Random point from $\mathbf{S}$
7:          $n \leftarrow (p - q) \times (p - r)$                          ▷ see eq. (4.78)
8:          $d \leftarrow p \cdot n$                                       ▷ see eq. (4.77)
9:          Distances $D \leftarrow$ DistancePointsToPlane($\mathbf{S}, n, d$)   ▷ see section 4.6.7
10:         if Length($D < \xi$) > Length($\pi$.inliers) then
11:             $\pi$.inliers $\leftarrow \mathbf{S}(D < \xi)$
12:             $\pi.p \leftarrow p$
13:             $\pi.q \leftarrow q$
14:             $\pi.r \leftarrow r$
15:             $\pi.n \leftarrow n$
16:             $\pi.d \leftarrow d$
17:     return $A$
```

### Step 2: Fit four lines to the sound events and the plane

To construct a virtual screen all the lines should lie in a plane. All points in $\mathbf{S}$ are therefore projected onto the computed plane (see eq. (4.79)) to form the new points $\mathbf{S}_p$. Using these points and a RANSAC scheme, a line in the plane can then be found.

To fit four different lines an iterative method is used. When a line has been fitted all points considered inliers are removed from $\mathbf{S}_p$ before the next iteration is run. Thus the number of available points decreases and the likelihood of finding another line increases.

The algorithm used is presented in algorithm 4. By experimenting it was found that $n = 400$ iterations and a threshold $\xi = 0.15$ meters were a good fit. A result of this can been seen in fig. 5.4a.

**Algorithm 4** Fit four lines using RANSAC

**Precondition:** $\mathbf{S}_p$ are points $\in \mathbb{R}^3$ projected onto a plane.
**Precondition:** The number of RANSAC iterations $n$.
**Precondition:** The RANSAC threshold $\xi$.

```
1:  function FitFourLines($\mathbf{S}_p$, $n$, $\xi$)
2:      $A \leftarrow$ A struct containing four lines
3:      for $i \leftarrow 1$ to 4 do
4:          $\mathbf{P} \leftarrow \mathbf{S}_p \notin A_{1 \dots (i-1)}$.inliers
5:          for 1 to $n$ do
6:              $p \leftarrow$ Random point from $\mathbf{P}$
7:              $q \leftarrow$ Random point from $\mathbf{P}$
8:              $\mathbf{P}_{\text{proj}} \leftarrow$ ProjectPointsOnLine($\mathbf{P}, p, q$)        ▷ see section 4.6.2
9:              Distances $d \leftarrow ||\mathbf{P}_{\text{proj}} - \mathbf{P}||$
10:             if Length($d < \xi$) > Length($A_i$.inliers) then
11:                 $A_i$.inliers $\leftarrow \mathbf{P}(d < \xi)$
12:                 $A_i.p \leftarrow p$
13:                 $A_i.q \leftarrow q$
14:     return $A$
```

### Step 3: Find the corners of the virtual screen

After the four lines are fitted, the corners of the virtual screen can be found. This is done by first finding the intersections between all different line combinations. A score is then computed using the line parameters $t$ and $s$ for each line combination, in order to find out which two lines every line would prefer to share corners with. The score is the sum of the absolute values of $t$ and $s$.

After the scores are computed, each of the four lines are connected to two other lines. However this often leads to bad results and hence a refinement step is executed. During the refinement it is also ensured that the corners (and lines) form a cyclic quadrilateral. This makes it easier to compute the area of the virtual screen, amongst other things.

The algorithm used to find all intersections can be seen in algorithm 5 and the algorithm to find the best corners can be found in algorithm 6. Thus, with the four corners found, the virtual screen is completed. The result can be seen in fig. 5.4b.

**Algorithm 5** Find the intersection between all the line combinations

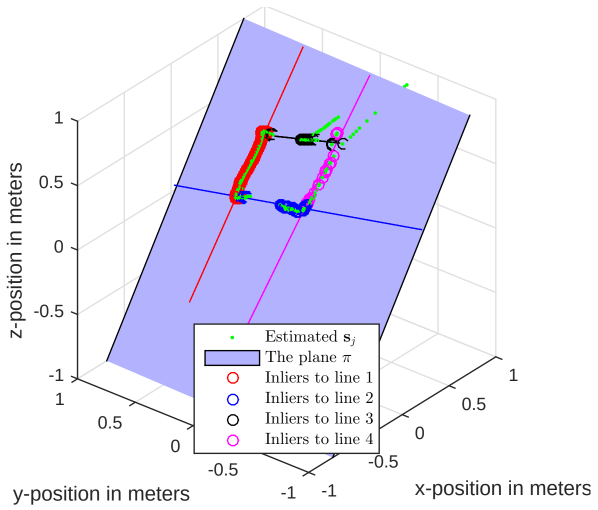
**Precondition:** The variable $A$ consists of four lines $L$.
**Precondition:** A line $L$ has two points $p$ and $q$ defining the line.

```
function FindIntersections($A$)
    for $i \leftarrow 1$ to 4 do
        $L_i \leftarrow A_i$
        for $j \neq i \leftarrow 1$ to 4 do
            $L_j \leftarrow A_j$
            $(a, b, c, d) \leftarrow (L_i.p, L_i.q, L_j.p, L_j.q)$
            $(t, s) \leftarrow$ Solution of the system $[b - a, -(d - c)][t, s]^T = [c - a, c - a]^T$   ▷ see section 4.6.3
            $L_i$.intersects($j$) $\leftarrow a + t(b - a)$   ▷ see section 4.6.3
            $L_i$.scores($j$) $\leftarrow |t| + |s|$
        $L_i$.preferred-corners $\leftarrow$ [Max($L_i$.scores)$_1$, Max($L_i$.scores)$_2$]
    return $A$
```

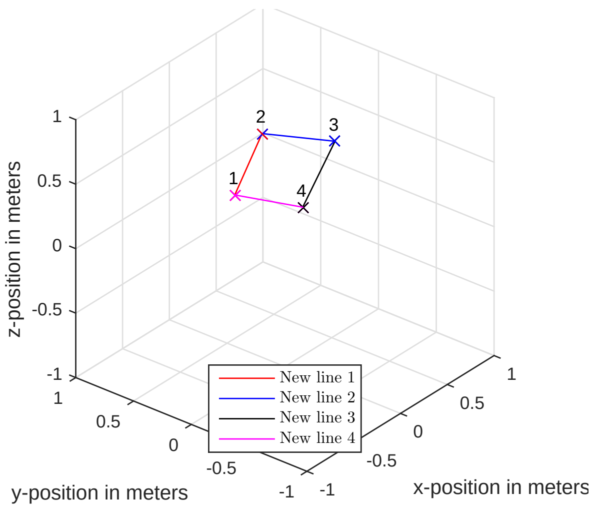
**Algorithm 6** Find the four corners of the virtual screen among the line intersections

**Precondition:** The variable $A$ consists of four lines $L$.
**Precondition:** A line $L$ has two preferred lines with whom it wants to share corners.
**Precondition:** A line $L$ has three intersection points with all the other lines.

```
function FindCorners($A$)
    $Q \leftarrow$ A $4 \times 2$ index matrix to contain the preferred-corners for each line.
    $\alpha \leftarrow$ Row indices of two lines $L$ in $A$ which have duplicate preferred-corners.
    $Q_\alpha \leftarrow A_\alpha$.preferred-corners
    $Q_{\{1,2,3,4\} \notin \alpha} \leftarrow [\alpha_1, \alpha_2]$
    for $i \leftarrow 1$ to 3 do
        $n \leftarrow Q_{i,2}$
        if $n \neq i + 1$ then
            Swap($Q_{i+1}, Q_n$)
            Swap($Q_{k=n}, Q_{k=i+1}$)   ▷ $k$ = any element in $Q$
            Swap($A_{i+1}, A_n$)
            for $j \leftarrow 1$ to 4 do
                Swap($A_j$.intersects($n$), $A_j$.intersects($i + 1$))
        if $i \neq Q_{i+1,1}$ then
            $Q_{i+1} \leftarrow$ FlipUpsideDown($Q_{i+1}$)
    for $i \leftarrow 1$ to 4 do
        $A_i$.corners(1) $\leftarrow A_i$.intersects($Q_{i,1}$)
        $A_i$.corners(2) $\leftarrow A_i$.intersects($Q_{i,2}$)
    return $A$
```



(a) After the four lines been fitted to the plane.



(b) After corner points have been found.

Figure 5.4: An example of the virtual screen in two different stages.

## 5.3 SRP-PHAT using Coarse-to-fine Region Contraction

The CFRC method is implemented very closely to its theoretical description in section 4.2.2. See algorithm 7. In this thesis the target volume $V_t$ was set to $4 \times 4 \times 4$ centimetres, using a similar error as in section 5.2.1. The initial search volume $V_S$ was set to an extra large volume to present good results, something that was done at a high computing power cost. The value set was $V_S = 2.5 \times 2.5 \times 2.5$ meters.

The number of points $J_i$ to evaluate each iteration was set to a constant after the initial $J_0$. The number of points to save each iteration $N_i$ was constant as well. The grid was searched with centimetre precision, the resolution $R$ being 0.01 meters.

**Algorithm 7** The CFRC method to estimate a sound event $\mathbf{s}_j$.

```
Precondition: A target volume $V_t$.
Precondition: An initial search volume $V_S$.
Precondition: The resolution $R$ of the search volume $V_S$.
Precondition: An audio segment $a$ of 2048 sample points.
Precondition: Microphone positions $\mathbf{r}$.

 1: function CFRC($V_t$, $V_S$, $R$, $a$, $\mathbf{r}$)
 2:     gcc ← UpperTriangleGccPhat($\mathbf{r}, a$)
 3:     $k \leftarrow (x, y, z)$

 4:     $J_k^0 \leftarrow \frac{1}{R} \frac{V_k^S}{V_k^t}$
 5:     $J_k \leftarrow \frac{1}{2} J_k^0$
 6:     $N_0 \leftarrow 50$
 7:     $N \leftarrow 30$

 8:     Step size $s_k \leftarrow$ Max($\frac{1}{J_k^0}$ SideLength($V_k^S$), 1)
 9:     Grid $D \leftarrow$ ConstructGrid($V_S, s$)

10:     Evaluated points $\mathbf{P} \leftarrow$ EvaluateSrpPhat($D, R, \mathbf{r}$, gcc)
11:     $\mathbf{P}_s \leftarrow$ Select(Sort($\mathbf{P}$), $N_0$)

12:     for $i \leftarrow 1$ to 100 do
13:         $B_k^{(i-1)} \leftarrow B_k^i$
14:         $B_k^i \leftarrow$ FindBoundaryBox($\mathbf{P}_s$)

15:         $s_k \leftarrow$ Max($\frac{B_k^i}{J_k}$, 1)
16:         $D_i \leftarrow$ ConstructGrid($B_k^i, s$)

17:         if $B_k^i \leq V_t$ or $B_k^i = B_k^{(i-1)}$ then
18:             $\mathbf{P} \leftarrow$ EvaluateSrpPhat($B_k^i, R, \mathbf{r}$, gcc)
19:             $p \leftarrow$ Sort($\mathbf{P}$)$_1$
20:             Break from the loop.
21:         else
22:             $\mathbf{G}_i \leftarrow \mathbf{P}_s >$ Mean($\mathbf{P}_s$)

23:         $\mathbf{P} \leftarrow$ EvaluateSrpPhat($D_i, R, \mathbf{r}$, gcc)
24:         $\mathbf{P}_s \leftarrow$ Select(Sort($\mathbf{P} \cup \mathbf{G}_i$), $N$)

25:     return $p$
```

Chapter 6

# Experiments

*In order to evaluate the system a number of experiments have to be conducted. This chapter describes the process of setting up these experiments and what limitations they give rise to.*

## 6.1 Experiment setup

Ideally the experiments would take place in a laboratory setting. The microphone positions could then be placed at locations measured in millimetre precision. A robot arm could control the movement of the sound source(s) to give an exact path for which all the transmission times are known. Noise and reverberation could be measured. All the comparisons to the estimated values would then be very easy to do and yield good results.

This was however not an option for this thesis and hence an experimental setup was created to mimic the laboratory setting just described.

## 6.2 Electronics

Eight microphones were used in this thesis to satisfy the minimal problem in section 4.3. The microphones were *Shure SV100* microphones and their frequency response can be seen in fig. 6.1.

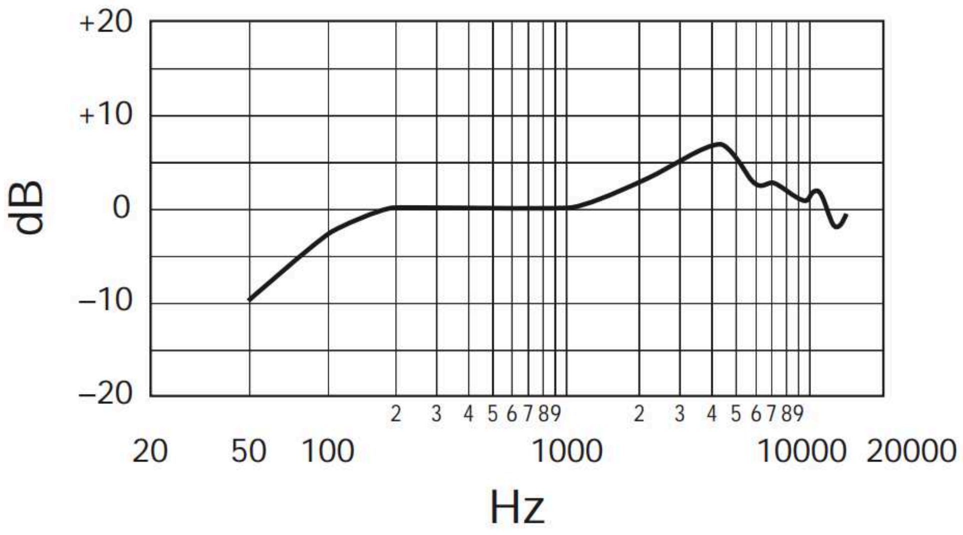


Figure 6.1: The frequency response of a Shure SV100 microphone. Image source is the user manual for the microphone.

To interface with the computer an audio interface from Roland was used, namely the STUDIO-CAPTURE. While both the microphones and the speakers were connected to the same device, the system still acted as an open system.

| Input jack | Microphone | Connection | Sensitivity |
|---|---|---|---|
| 1–4 | Shure SV100 | XLR – XLR | 58 dB |
| 5–8 | Shure SV100 | XLR – 6.35 mm TRS | 58 dB |

Table 6.1: Settings for the microphones.

| Output jack | Speaker type | DAW output level |
|---|---|---|
| 7 | Portasound | 5 dB |
| 8 | Cubee | -13 dB |

Table 6.2: Settings for the speakers.

The settings used for the microphones are found in table 6.1. A low cut filter with a cutoff frequency at 75 Hz was enabled on the device for all of the input jacks.

The speakers used to generate sound were a Roxcore Portasound speaker and a Roxcore Cubee Portabel Bluetooth-speaker. Since the experiments in this thesis never use more than two microphones as mentioned in section 1.3.3, only one speaker of each type was used. When recordings were produced with only one output channel the latter speaker was used.

The master output level was set to -20 dB on the STUDIO-CAPTURE. The other settings can be found in table 6.2.

## 6.3 Software implementation

The system in this thesis was implemented in MATLAB as mentioned in section 1.3.3. To interface with the STUDIO-CAPTURE it was connected to the computer via USB 2.0 using the driver ASIO4ALL, a Universal ASIO Driver For WDM Audio. The entire project was run on the Windows platform.

To generate sounds and record synchronized sounds the *Real-Time Audio Processing Framework for Matlab* developed by Swartling [18] was used. The audio was processed in blocks of 512 sample points at a time, both for the signal transmissions and for the recordings. The signals were pre-generated before the transmissions to avoid output lag. Likewise for the real-time part of the system pipeline, the entire signal was recorded before running the real-time experiments. However, the recording was split into the same block size as the recording had been recorded in, before being fed to the real-time system.

## 6.4 Experiment rig

A custom rig was built in order to maintain the setup between different experiments. Having known positions for the microphones and known positions for the sound source path(s) make it easier to do ground truth comparisons. The use case is to remotely control a monitor of a computer. Likely this could be a 22 inch monitor with the microphones placed ad hoc around it. However, the microphones should not be placed in a plane since this messes with the system and it becomes rank-deficient. The monitor, the microphones and the hand of the user would probably fit within one cubic meter.

An L-shaped particle-board with feet was constructed. At different measured locations holes were drilled, allowing microphones to be slotted in different configurations. Thus the microphones could be inserted both vertically and horizontally and a few configurations thereof made it possible to not slot them in a plane.

A couple of nails were also hammered onto the construction at measured locations. A thread was attached to a nail and then looped around via a couple of other nails, to make it possible to get a rough ground truth path for the sound source paths.

The construction can be seen in fig. 6.2.

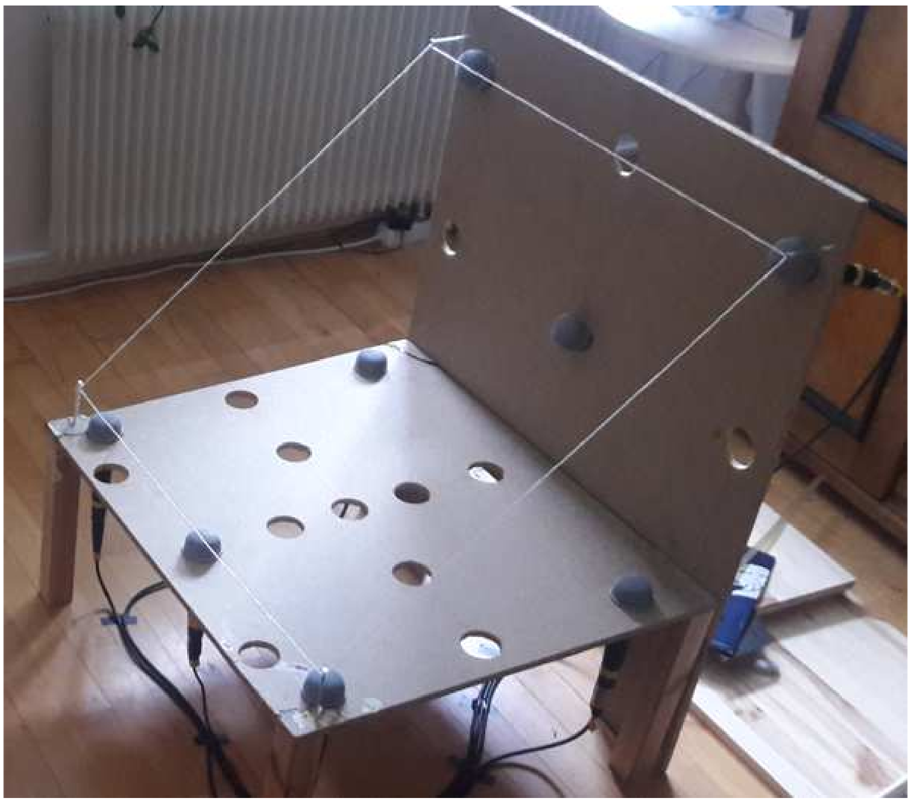


Figure 6.2: The experiment rig constructed for this thesis.

## 6.5 Used configuration for the microphones and sound source paths

Here the ground truth configuration used for the majority of the experiments is presented. The locations in meters of the $M = 8$ microphones used are

$$\hat{\mathbf{r}} = \begin{bmatrix} 0.54 & 0.00 & 0.45 \\ 0.54 & 0.60 & 0.45 \\ 0.45 & 0.00 & 0.01 \\ 0.45 & 0.60 & 0.01 \\ 0.00 & 0.00 & 0.01 \\ 0.00 & 0.60 & 0.01 \\ 0.00 & 0.30 & 0.01 \\ 0.54 & 0.30 & 0.20 \end{bmatrix}_{M \times (x,y,z)} . \tag{6.1}$$

When unscrewing one of the microphones it was found that the sensor membrane was not located at the exact middle line which the microphone rest on in the experiment rig. Hence the offset of one centimetre in either the $x$-axis or the $z$-axis. The location of microphone five is therefore found at $(0, 0, 0.1)$ instead of $(0, 0, 0)$.

The $k = 4$ corner points of the sound path are located at

$$\hat{\mathbf{c}} = \begin{bmatrix} -0.005 & -0.005 & 0.035 \\ -0.005 & 0.605 & 0.035 \\ 0.45 & 0.605 & 0.455 \\ 0.45 & -0.005 & 0.455 \end{bmatrix}_{k \times (x,y,z)} . \tag{6.2}$$

This is actually three centimetres inside and lower than the thread seen in fig. 6.2. The size of the Cubee speak is $6 \times 6 \times 6$ centimetres, thus half of the width is 2.5 centimetres. So when the speaker is moved along the thread (on the inside) the offset above is obtained. Touching the thread with the speaker generates a noise and hence an extra 0.5 centimetres are added as the path shrinks a little bit more.

The locations of the microphones and the sound source path can be seen in fig. 6.3.

The experiments conducted generated signals of 11 seconds. In those 11 seconds the speaker was moved, one loop along the sound source path thread, from the bottom-right corner to the bottom-left corner all away around until the bottom-right corner was reached again. If any time remained after the loop the speaker was moved towards the opposite corner of the path from the starting position.

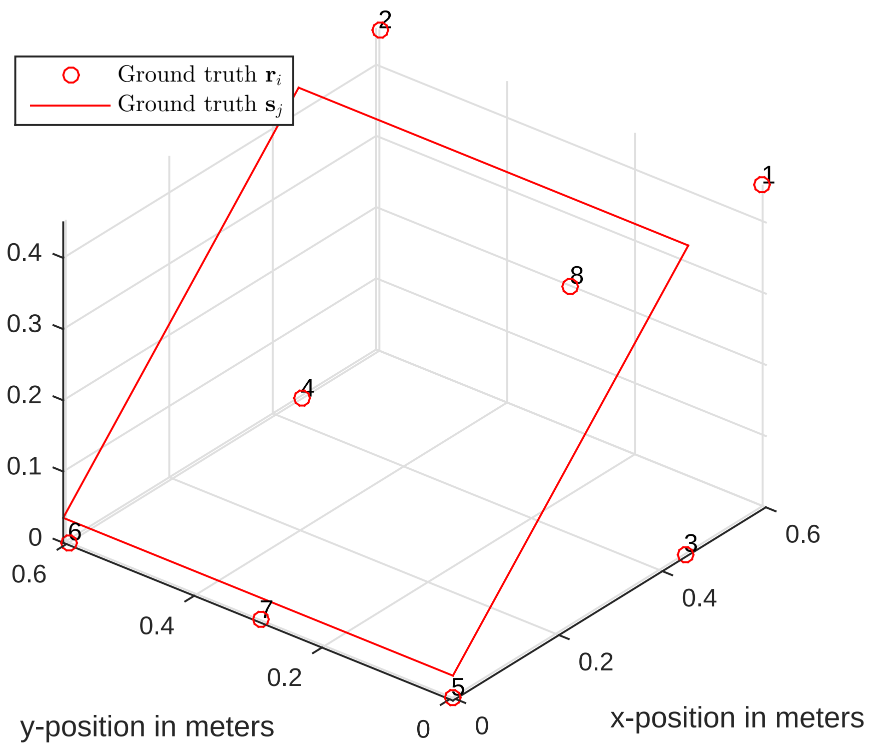


Figure 6.3: The ground truth positions for the microphones and the sound path in the experiments conducted.

Chapter 7

# Results

*In this chapter the results as well as some evaluation of said results are presented.*

## 7.1 Signal processing

### 7.1.1 A visual comparison using a single sound source

In order to decide which signal method and frequency band to use, a comparison of the four different signal methods were done for three different frequency bands. The GCC-PHAT images are plotted against each other in fig. 7.1 for the channel pair $(1, 3)$. From a visual inspection of the subfigures it seems like the SPR signal method preforms best for both 440 – 1 440 Hz (fig. 7.1j) and 6 500 – 7 500 Hz (fig. 7.1k), with the first band preforming slightly better. None of the signal methods seems to perform well in the frequency span 15 000 – 16 000 Hz (figs. 7.1c, 7.1f, 7.1i and 7.1l).

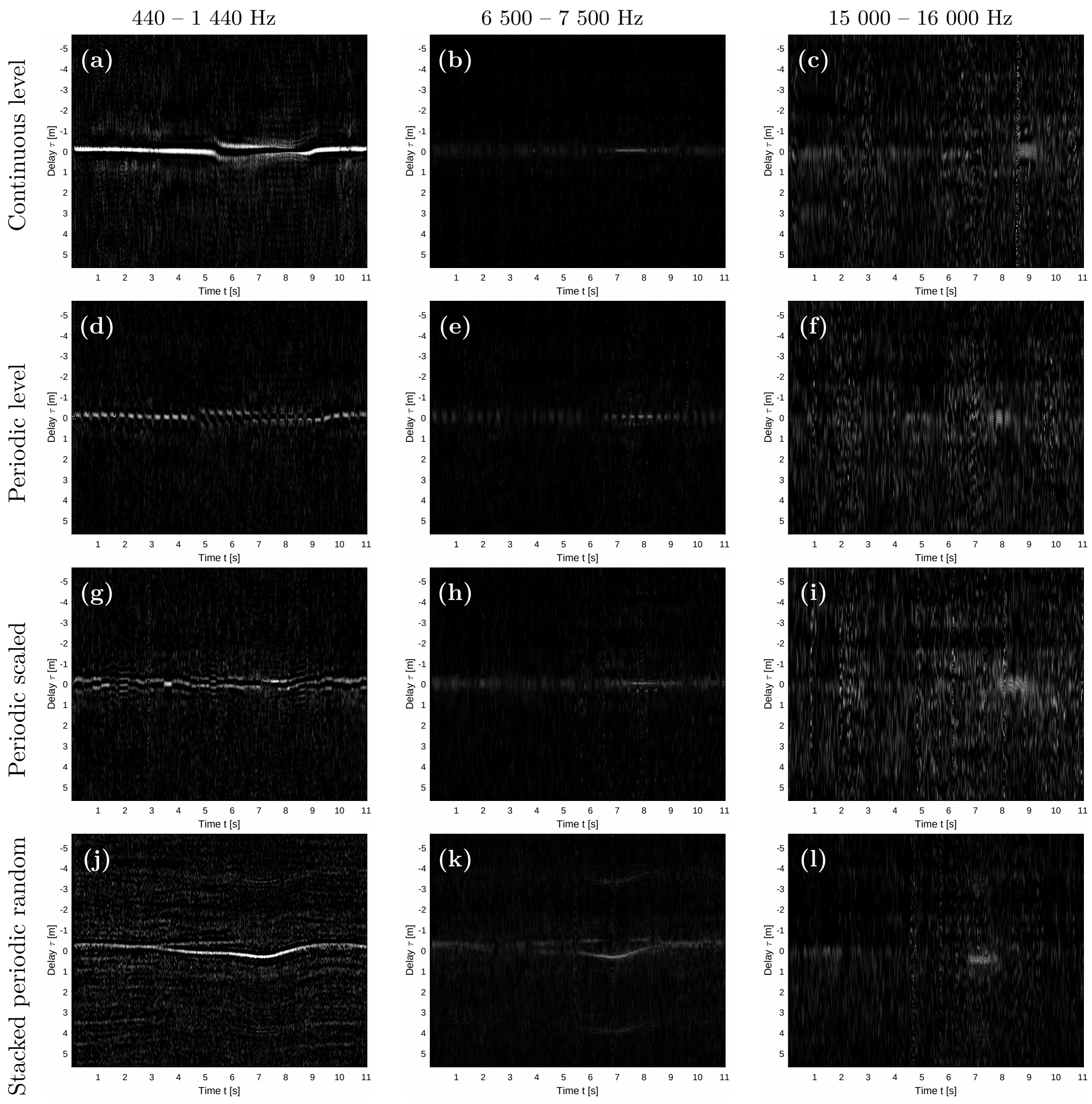


Figure 7.1: A comparison of different GCC-PHAT for channel pair $(1, 3)$, where the frequency and the signal method vary.

### 7.1.2 An in-depth comparison using a single sound source

A visual inspection is not the best comparison method. Therefore an alternative way to compare the different signal methods for different frequencies was developed. The method has two steps:

1. Compose a ground truth image.
2. Compare each ground truth image with the measured signal.

Constructing a ground truth image, ideally, consists of computing the TDOA value for each time frame within the GCC-PHAT, for each channel pair, for each different signal method and for each different frequency band to be tested. If the experimental setup had a way to automate the path of the sound source, so that at each point in time the location of the sound source would be known, the TDOA could easily be calculated. This setup was not available as described in section 6.1.

The alternative option is to construct ground truth images by hand. This has to be done individually for each recording as the exact path of the sound source is unknown. For the four different signal methods and three different frequencies this is $4 \cdot 3 \cdot m(M-1)! = 672$ different ground truth images. Which is way too many for the scope of this thesis. Therefore ground truth images were only created for channel pairs $(1, a_i)$ for each recording, where $a_i = 2, \dots, 8$. This yields a total number of $4 \cdot 3 \cdot 7 = 84$ images.

To produce the ground truth images a GCC-PHAT image was created for each of the 84 channel pairs. Then the most "likely" path was coloured in black and white to represent a bit mask, where the white (true values) represented the TDOA value chosen as the correct one. See fig. 7.2 for an example. This method has several drawbacks however:

- It is very easy to draw the "line" a few pixels off by accidental hand movements.
- It is constructed from the GCC-PHAT images, which means that
  - any errors produced by the GCC-PHAT method will follow into the ground truth images,
  - the evaluation step uses the same data that was the basis for the ground truth image, and
  - when the image is very fuzzy or unclear the ground truth path will be the result of guesswork.

Thus a visual confirmation should be done in conjunction with this method.

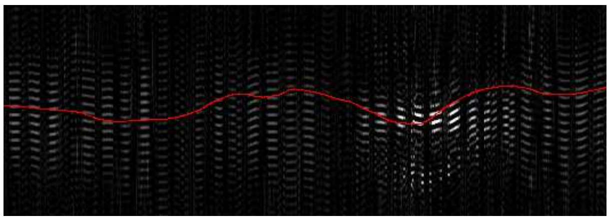


Figure 7.2: The ground truth image for channel pair $(1, 3)$ for the CL signal method with the frequency span 6 500 – 7 500 Hz. Here it is overlayed on top of the GCC-PHAT image in fig. 7.1b with the ground truth TDOA path represented in red.

The evaluation step consists of the following steps:

1. Compute peaks $p$ and refined peaks $p_r$ as described in section 5.2.1 for all of the 84 channels pairs.
2. For each channel pair, the distance (in sample points) between the ground truth TDOA value for each time frame and the closest TDOA value in $p$ and $p_r$ will be computed.
3. All distances smaller than a threshold $\xi$ in each recording are then summed and divided by the total number of time frames.

This results in a percentage that describes how well the signal method and the frequency band preforms. Here $\xi = 5.6$ sample points $= 0.04$ meters was chosen (as in section 5.2.1). The result for the same recordings as in fig. 7.1 can be seen in table 7.1.

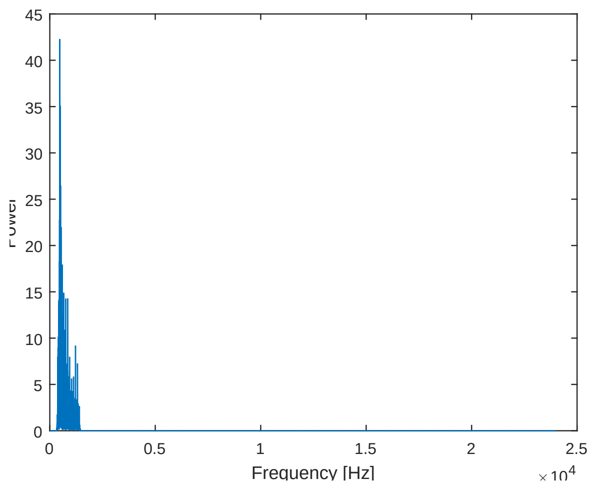
| | 440 – 1 440 Hz | | 6 500 – 7 500 Hz | | 15 000 – 16 000 Hz | |
|---|---|---|---|---|---|---|
| | $p$ | $p_r$ | $p$ | $p_r$ | $p$ | $p_r$ |
| Continuous level (CL) | 57.91% | 40.43% | 36.01% | 71.74% | 2.39% | 1.03% |
| Periodic level (PL) | 32.10% | 21.39% | 31.57% | 51.40% | 0.97% | 0.85% |
| Periodic scaled (PS) | 32.17% | 26.53% | 39.78% | 26.54% | 1.22% | 0.46% |
| Stacked periodic random (SPR) | 88.83% | 90.63% | 50.98% | 82.80% | 2.42% | 0.39% |

Table 7.1: A comparison of how well the different signal methods perform for some different frequency bands. Comparable to the results in fig. 7.1.

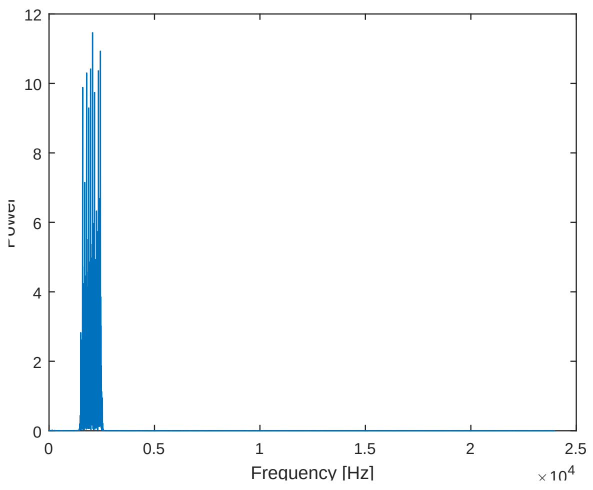
### 7.1.3 Results using multiple sound sources

In the experiment with multiple sound source, two speakers were used in this thesis. Each speaker emits a different signal. This results in a single $M$ channel recording that is separated into two new $M$ channel recordings using a bandpass filter. In fig. 7.3 the result of such a separation can be seen. The span of the peaks in each new $M$ channel recording are clearly shown as different intervals, indicating a successful separation.

The first output signal used the SPR signal method for the frequency band 400 – 1 400 Hz, and the second one used the SPR signal method for the frequency band 1 500 – 2 500 Hz. In fig. 7.4 the GCC-PHAT images shows two different TDOA paths confirming that the separation worked. Although it might be difficult to distinguish the path in the right image as there is aliasing of the signal due to the higher frequency band.

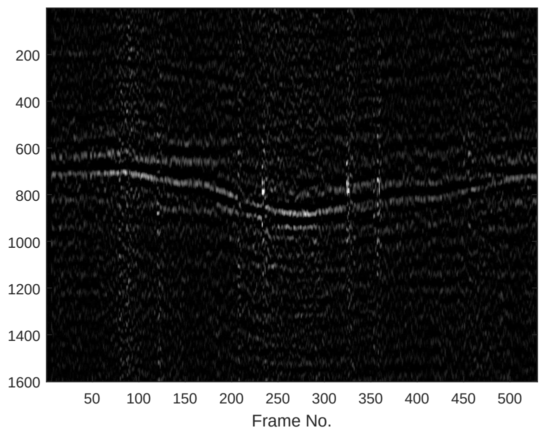


(a) Frequency band: 400 – 1 400 Hz.

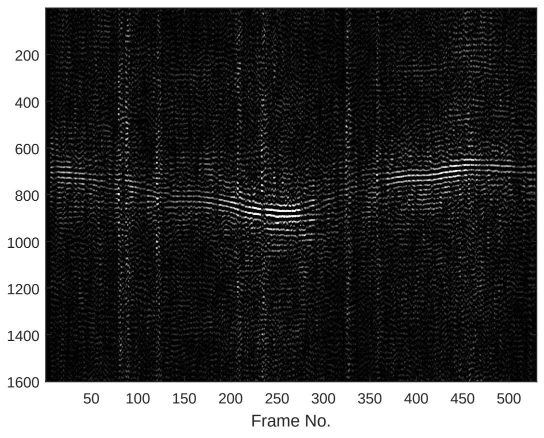


(b) Frequency band: 1 500 – 2 500 Hz

Figure 7.3: The frequency spectrum for the two extracted audio clips after the separation of the recording.



(a) Frequency band: 400 – 1 400 Hz.



(b) Frequency band: 1 500 – 2 500 Hz

Figure 7.4: The GCC-PHAT image for channel pair (2, 7) for the two extracted audio clips after the separation of the recording.

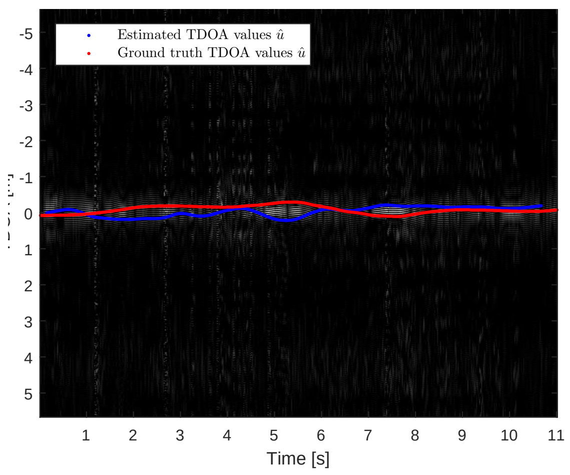
## 7.2 System self-calibration

### 7.2.1 Estimation of TDOA values $u_{i,j}$

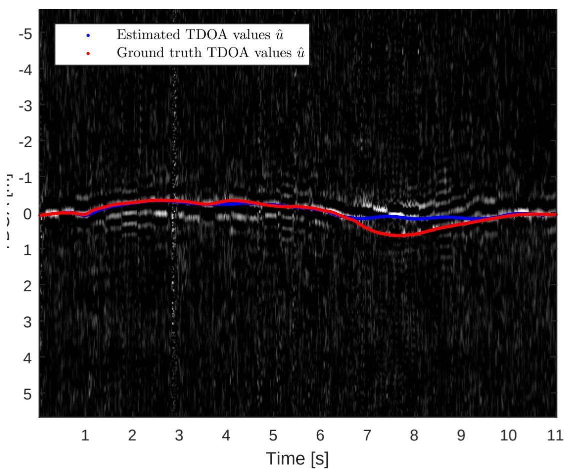
To evaluate the estimator for the TDOA values $u_{i,j}$ a comparison to the same ground truth images as created in section 7.1.2 is carried out. The ground truth TDOA values are referred to as $\hat{u}_{i,j}$.

The estimator for the TDOA values $u_{i,j}$ seems to be robust in the sense that it follows some path and is able to span the entire width of GCC-PHAT image. A few examples can be seen in fig. 7.5 for some different channel pairs, signal methods and frequency bands.

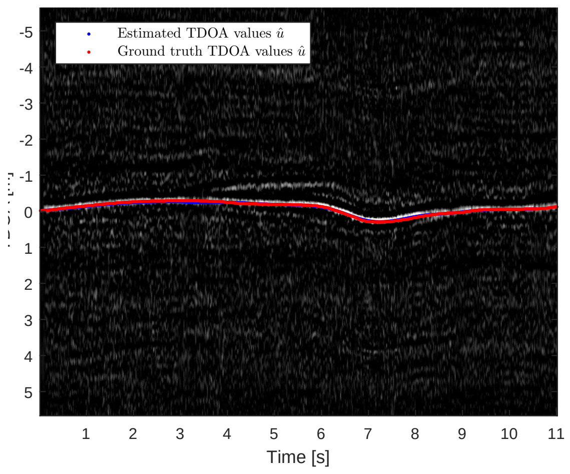
However, the ground truth often differs from the estimated TDOA path quite a bit. It is hard to say if this is because $\hat{u}_{i,j}$ is badly drawn in the ground truth image or if the estimated TDOA values $u_{i,j}$ are bad. The recording with the SPR signal method and the frequency band 440 – 1440 Hz, see figs. 7.1j and 7.5c, will be studied further in the thesis. This recording will be referred to as *SPR440*.



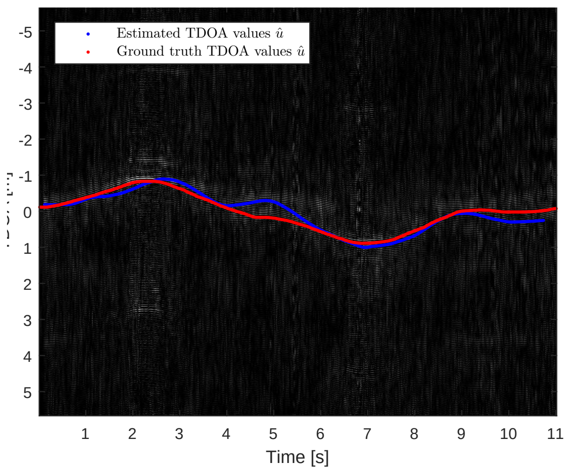
(a) The CL signal method for 6 500 – 7 500 Hz for the channel pair (1, 2).



(b) The PS signal method for 440 – 1 440 Hz for the channel pair (1, 4).



(c) The SPR signal method for 440 – 1 440 Hz for the channel pair (1, 8).



(d) The CL signal method for 6 500 – 7 500 Hz for the channel pair (1, 6).

Figure 7.5: Examples of comparisons between the estimated TDOA values $u_{i,j}$ (in red) and he ground truth TDOA values $\hat{u}_{i,j}$ (in blue), for some different channel pairs, signal methods and frequency bands.

Looking at the SRP440, it is time to compare the estimated TDOA values $u_{i,j}$ more closely to the ground truth TDOA values $\hat{u}_{i,j}$. First the distances $d$ between all the values $u_{i,j}$ and all the values $\hat{u}_{i,j}$ are calculated as

$$d_k = |w_k - \hat{w}_k| \tag{7.1}$$

where $\mathbf{w}$ is the flattened matrix $\mathbf{u}$ and $\hat{w}$ is the flattened matrix $\hat{u}$. That yields $n = M \times N$ distances $d$. The distances are used as an error measurement. The lower the distance, the better.

The mean of the error is calculated by

$$\mu = \frac{1}{n}\sum_{i=1}^{n} d_i. \tag{7.2}$$

The standard deviation of the error is calculated by

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \mu)^2}. \tag{7.3}$$

The root mean square of the error, RMS, is calculated by

$$\text{RMS} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} d_i^2}. \tag{7.4}$$

As only one recoding is used to evaluate the estimator, the error measurements defined above were calculated for 50 different iterations. The reason being that the estimator is not deterministic. The average of the distance errors were found to be (in sample points)

$$\mu = 1.692, \quad \sigma = 1.517 \quad \text{and} \quad \text{RMS} = 2.227.$$

In meters this becomes

$$\mu = 0.012, \quad \sigma = 0.011 \quad \text{and} \quad \text{RMS} = 0.016.$$

This is a very small error of about $1 \pm 1$ centimetres. Studying the other channel pairs visually for SPR440 it is hard to say if this is because of an incorrect ground truth image or not. The fact remains that it is very good nonetheless.

### 7.2.2 Estimation of microphone positions $\mathbf{r}_i$

Similarly to how the TDOA values $u_{i,j}$ were studied, the estimated microphone positions $\mathbf{r}_i$ can be studied. The ground truth microphones positions $\hat{\mathbf{r}}_i$ are defined in section 6.5. The distances $d$ between the estimated $\mathbf{r}_i$ and the ground truth $\hat{\mathbf{r}}_i$ are calculated by

$$d_i = \sqrt{(r_x^i - \hat{r}_x^i)^2 + (r_y^i - \hat{r}_y^i)^2 + (r_z^i - \hat{r}_z^i)^2} \tag{7.5}$$

where $\mathbf{r}_i = [r_x^i, r_y^i, r_z^i]^T$ and $\hat{\mathbf{r}}_i = [\hat{r}_x^i, \hat{r}_y^i, \hat{r}_z^i]^T$. Again the distances are used as error measurements. Then the mean $\mu$, the standard deviation $\sigma$ and the RMS can be calculated for the error measurements as in eqs. (7.2) to (7.4).

Again the tests are run on SPR440 with 50 iterations. Of the 50 iterations, one of them produced an error in the last optimization step of section 5.2.3 and nine of them failed to estimate any $\mathbf{r}_i$ (and $\mathbf{s}_j$) at all. Therefore these were removed yielding a total of 40 iterations instead.

The average of the error measurements were found to be (in meters)

$$\mu = 0.118, \quad \sigma = 0.050 \quad \text{and} \quad \text{RMS} = 0.129.$$

In fig. 7.6 the different means of the errors for each of the iterations can be studied in detail.

This time it is very interesting to make a visual comparison of the iterations since there is a geometric interpretation of the problem. In fig. 7.7 two good examples and two bad examples are shown as well as their individual $\mu, \sigma$ and RMS. It is somewhat difficult to see how far away the points are from each other in certain angles, this becomes easier when rotating them around in MATLAB however.

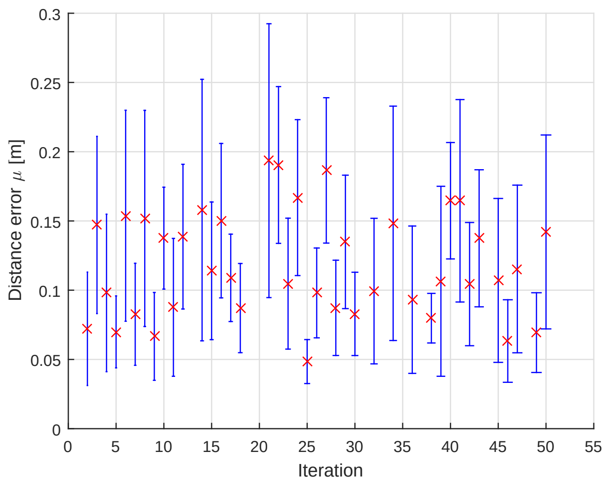
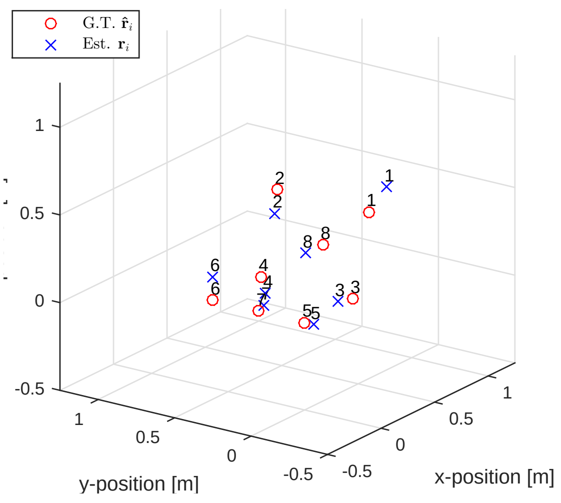
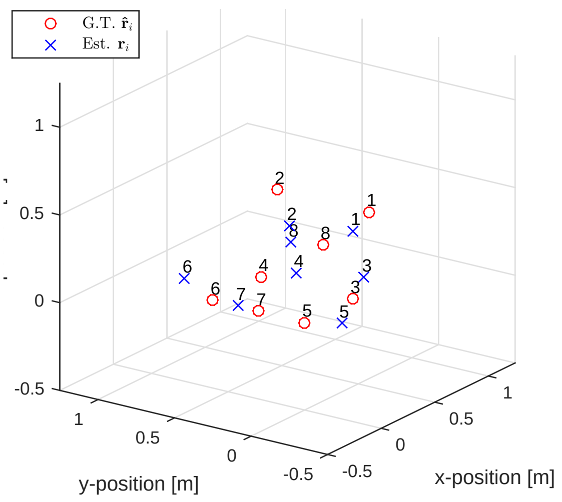


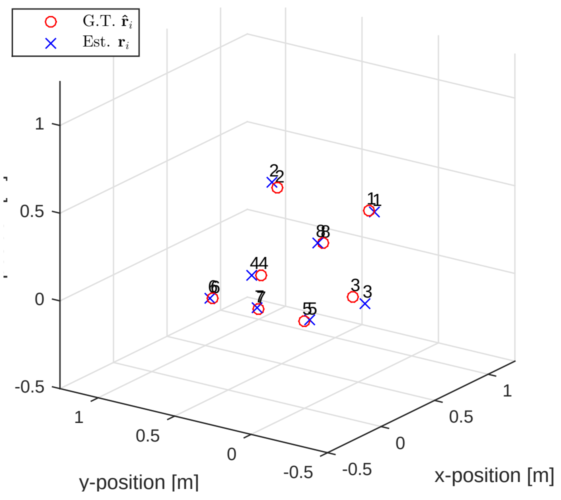
Figure 7.6: The means of the distance errors with their standard deviations for each iteration when evaluating $\mathbf{r}_i$.



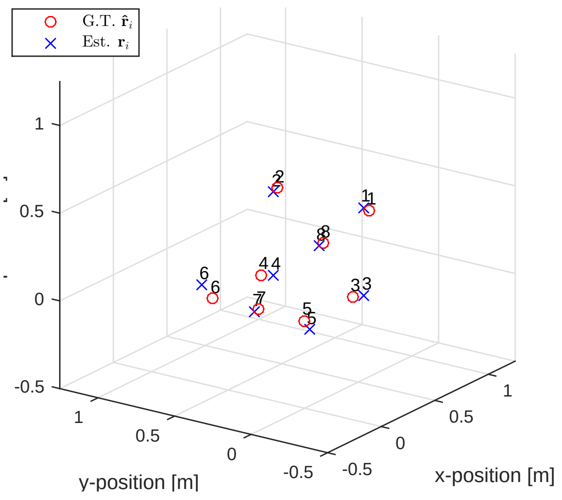
(a) Iteration 6 with $\mu = 0.154$ m, $\sigma = 0.076$ m and RMS $= 0.772$ m. (Bad.)



(b) Iteration 22 with $\mu = 0.190$ m, $\sigma = 0.057$ m and RMS $= 0.119$ m. (Bad.)



(c) Iteration 25 with $\mu = 0.048$ m, $\sigma = 0.016$ m and RMS $= 0.051$ m. (Good.)



(d) Iteration 46 with $\mu = 0.063$ m, $\sigma = 0.030$ m and RMS $= 0.070$ m. (Good.)

Figure 7.7: Examples of estimated $\mathbf{r}_i$ compared to their ground truths $\hat{\mathbf{r}}_i$ for SPR440.

### 7.2.3 Estimation of sound source path $\mathbf{s}_j$

Now the sound events $\mathbf{s}_j$ will be compared to the ground truth sound source path spanned by the corner points $\hat{\mathbf{c}}_k$ defined in section 6.5. In order to get some error metrics a few tricks has to be done.

First all the estimated $\mathbf{s}_j$ are projected down onto all of the four lines formed by $\hat{\mathbf{c}}_k$. Then the distances between all the projected points and all of the lines are computed. Each original point $\mathbf{s}_j$ is then assigned to the line closest to itself. The algorithm can be seen in algorithm 8.

**Algorithm 8** Assign each sound event $\mathbf{s}_j$ to a ground truth line.

**Precondition:** A variable $A$ consisting of four lines $L$ that define the ground truth sound source path.
**Precondition:** A line $L$ has two points $p$ and $q$ defining the line.
**Precondition:** Estimated sound event points $\mathbf{s}_j$ located in $\mathbf{S}$.

```
1:  function AssignPointToLine($A$, $\mathbf{S}$)
2:      $n \leftarrow$ Length($\mathbf{S}$)
3:      $D \leftarrow$ A $n \times 4$ matrix containing the distances from each point to each line.
4:      $P \leftarrow$ A $n \times 4$ matrix containing the projection of $\mathbf{S}$ onto each line.
5:      for $j \leftarrow 1$ to $n$)  do
6:          $s \leftarrow \mathbf{S}_j$
7:          for $k \leftarrow 1$ to 4 do
8:              $p \leftarrow A_k.p$
9:              $p \leftarrow A_k.q$
10:             $D_{j,k} \leftarrow$ DistancePointToLine$(s, p, q)$
11:             $P_{j,k} \leftarrow$ ProjectPointOntoLine$(s, p, q)$
12:     $c \leftarrow$ Min$(D, 2)$                    ▷ $c =$ index of the closest line for each point.
13:     for $k \leftarrow 1$ to 4 do
14:         $A_k$.distances $\leftarrow D_{c=k}$
15:         $A_k$.inliers $\leftarrow P_{c=k}$
16:     return $A$
```

The distance between each sound event $\mathbf{s}_j$ and a line can be computed by eq. (4.83) and distances will be used as the error measurements. The standard metrics; the mean $\mu$, the standard deviation $\sigma$ and the RMS; can be calculated for the errors as in eqs. (7.2) to (7.4). The metrics are computed for the exact same iterations as in section 7.2.2.

Before computing the errors some extreme outliers among the sound events $\mathbf{s}_j$ were removed. All $\mathbf{s}_j$ whose distance to a line were above four meters were considered extreme outliers. For the majority of the iterations the amount of such outliers were zero and for the other cases it was between five and twenty points.

The average of values were found to be (in meters)

$$\mu = 0.292, \quad \sigma = 0.273 \quad \text{and} \quad \text{RMS} = 0.405.$$

In fig. 7.8 the different means of the errors for each of the iterations can be studied in detail.

Again the visual comparison of the iterations is interesting. In fig. 7.9 a few examples are shown as well as their individual $\mu, \sigma$ and RMS. The colors of the points and the line segments indicate which points are measured to which line.

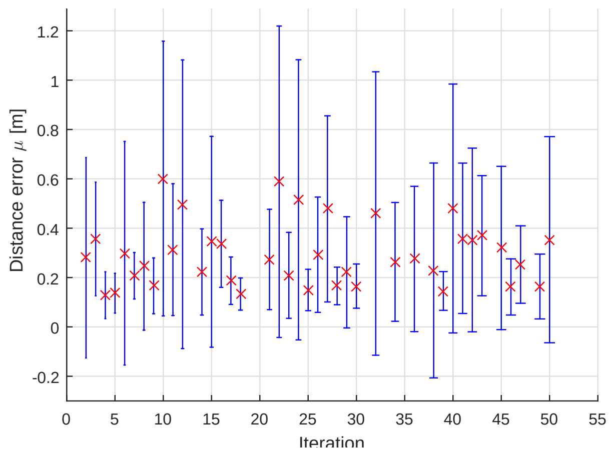
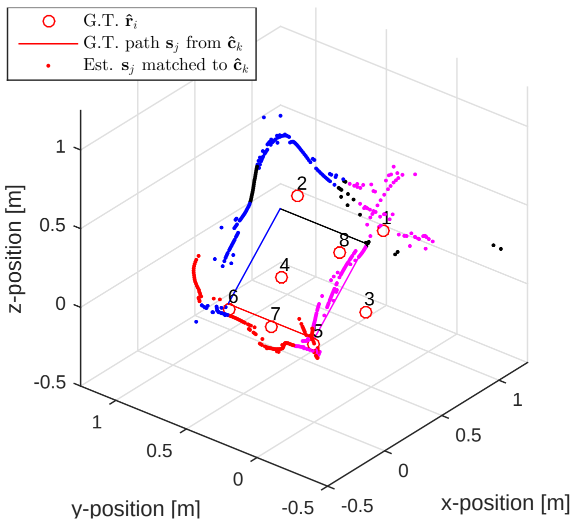
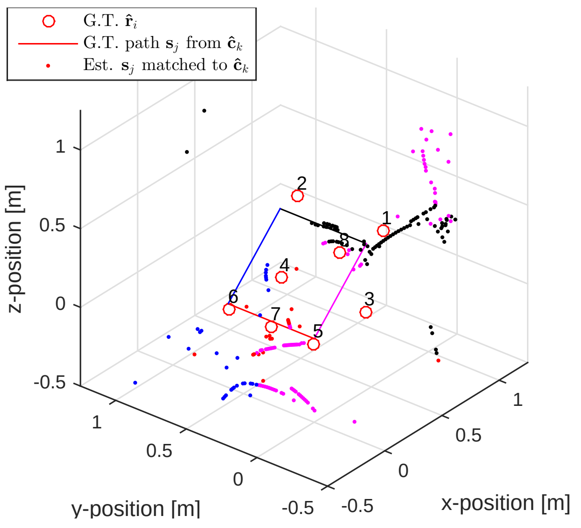


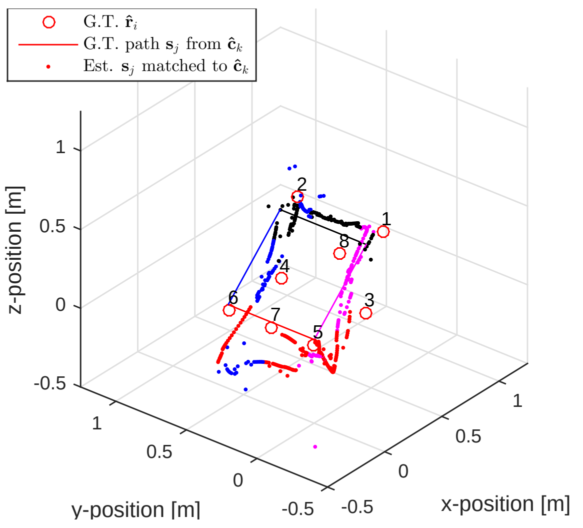
Figure 7.8: The means of the distance errors with their standard deviations for each iteration when evaluating $\mathbf{s}_j$.



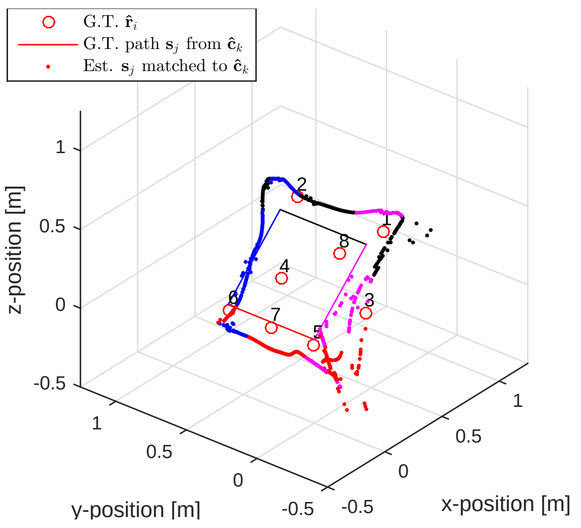
(a) Iteration 6 with $\mu = 0.298$ m, $\sigma = 0.453$ m and RMS $= 0.542$ m.



(b) Iteration 10 with $\mu = 0.601$ m, $\sigma = 0.557$ m and RMS $= 0.819$ m.



(c) Iteration 14 with $\mu = 0.223$ m, $\sigma = 0.175$ m and RMS $= 0.283$ m.



(d) Iteration 18 with $\mu = 0.133$ m, $\sigma = 0.065$ m and RMS $= 0.148$ m.

Figure 7.9: Examples of estimated $\mathbf{s}_j$ compared to the lines spanned by ground truth $\hat{\mathbf{c}}_k$ for SPR440.

### 7.2.4 Estimation of the virtual screen

A few examples of the virtual screen compared to the sound events $\mathbf{s_j}$ can be seen in fig. 7.10. The examples are generated from SPR440. Sometimes the quadrilaterals calculated are neither simple or convex, which is bad (namely complex and concave quadrilaterals). Other times the algorithm yields good results visually.

Complex quadrilateral are quadrilaterals where the diagonals of the quadrilateral intersect and concave quadrilateral are quadrilaterals where one of the interior angles of the quadrilateral is greater than 180 degrees. Examples of these shapes can be seen in fig. 7.10d. A better algorithm for finding the virtual screen would reject these complex and concave quadrilaterals.

Error metrics for the evaluation are calculated using the virtual screen corner points $\mathbf{c}_k$ and the ground truth virtual screen spanned by the ground truth corner points $\hat{\mathbf{c}}_k$. The points $\hat{\mathbf{c}}_k$ form a rectangular quadrilateral and the points $\mathbf{c}_k$ form a general quadrilateral as previously mentioned. To compare the quadrilaterals the following metrics are used as error measurements: the distances $d$ between the mid points, the difference between the areas $A$, and the difference between the circumferences $C$.

The area of a quadrilateral can be computed using the corner points. A quadrilateral spanned by the corners $\mathbf{a}, \mathbf{b}, \mathbf{c}$ and $\mathbf{d}$ has two diagonals $\mathbf{D}_1 = \mathbf{c} - \mathbf{a}$ and $\mathbf{D}_2 = \mathbf{d} - \mathbf{b}$. The area is then given by

$$A = \frac{|\mathbf{D}_1 \times \mathbf{D}_2|}{2}. \tag{7.6}$$

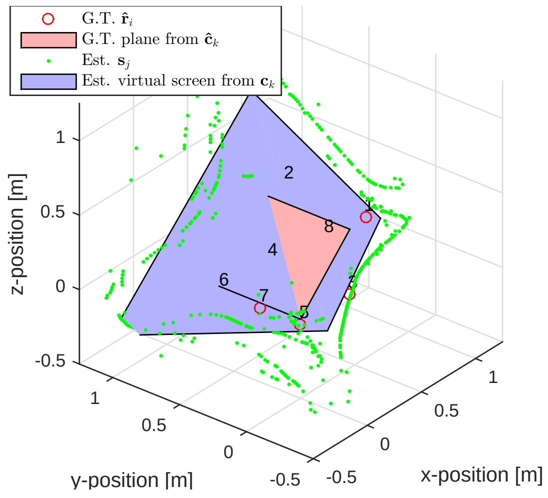
To compute the mid points, the intersection of the diagonals $\mathbf{D}_1$ and $\mathbf{D}_2$ is calculated using the technique in section 4.6.3. Finally the circumference is calculated as the sum of all the four sides $\mathbf{b}-\mathbf{a}, \mathbf{c}-\mathbf{b}, \mathbf{d}-\mathbf{c}$ and $\mathbf{a}-\mathbf{d}$.

The metrics are computed for the exact same iterations as in section 7.2.2. The average of values were found to be

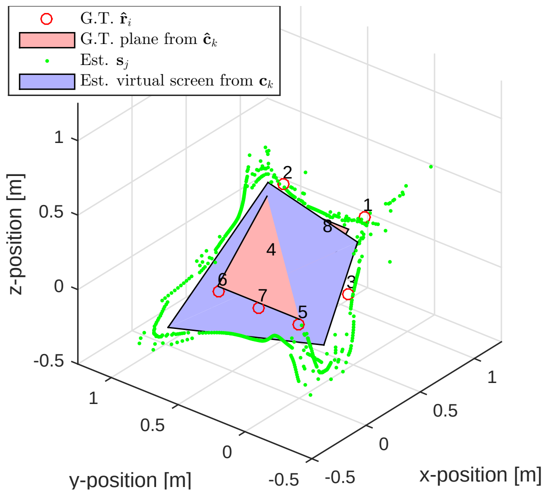
$$\mu_d = 1.044 \text{ m}, \quad \mu_C = 2.608 \text{ m} \quad \text{and} \quad \mu_A = 0.474 \text{ m}^2.$$

The individual errors for all the iterations can be found in fig. 7.11. In some places the error values are outside the graph since it is more interesting to study the majority of the error measurements. So if a value that existed in a previous error graph (figs. 7.6 and 7.8) is now missing it just lies at a much higher value than the graph displays.

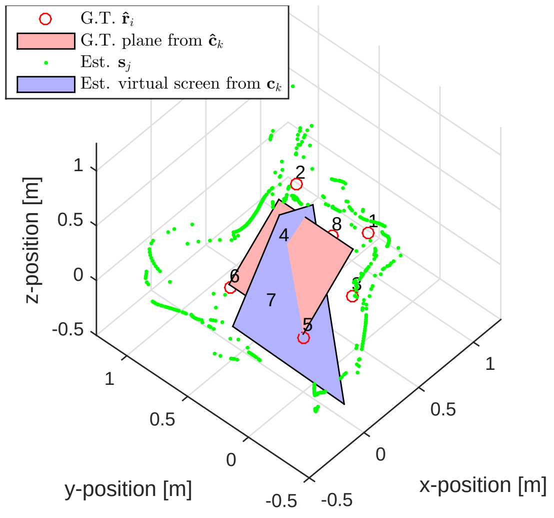
A visual comparison of all the 40 iterations was also done as the average of the errors is quite high. Only five iterations were found to be reasonably fitted, namely iterations 15, 17, 21, 25 and 49. Four of these can be seen in fig. 7.10.



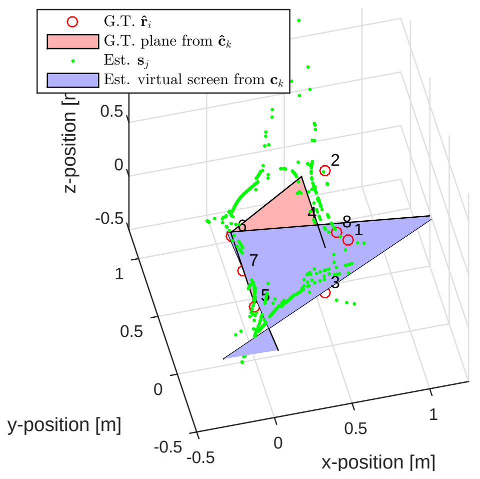
(a) A reasonable fitted virtual screen (iteration 15). The ground truth plane from $\mathbf{c}_k$ intersect with the virtual screen.



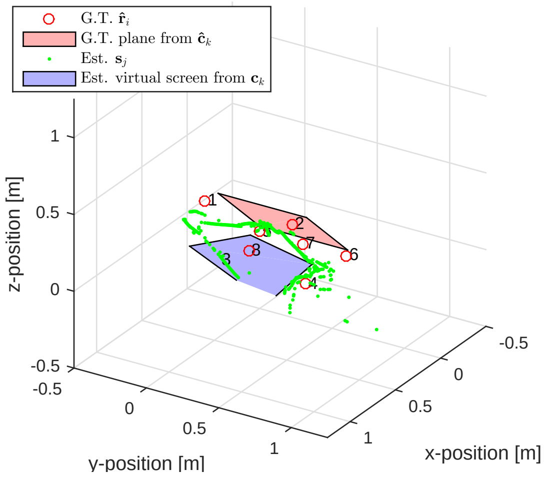
(b) A reasonable fitted virtual screen (iteration 17). The ground truth plane from $\mathbf{c}_k$ intersect with the virtual screen.



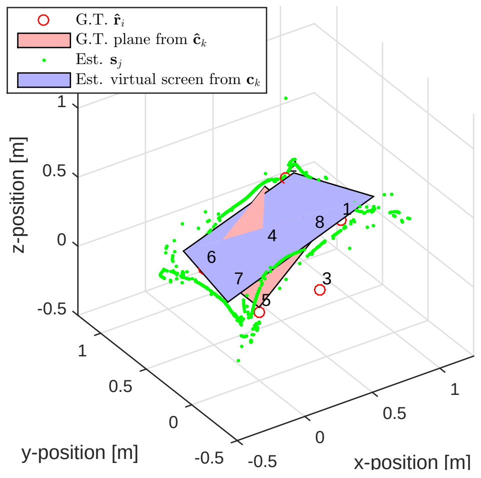
(c) A badly fitted virtual screen (iteration 16).



(d) A complex virtual screen (iteration 38).

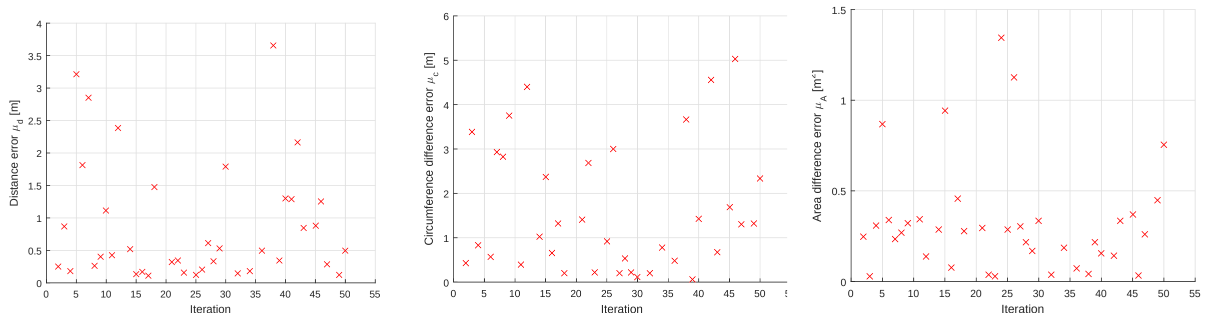


(e) A reasonable fitted virtual screen (iteration 21).



(f) A reasonable fitted virtual screen (iteration 25).

Figure 7.10: A few examples of virtual screen compared to the sound events $\mathbf{s}_j$ for SPR440.



(a) The mid point distance errors.

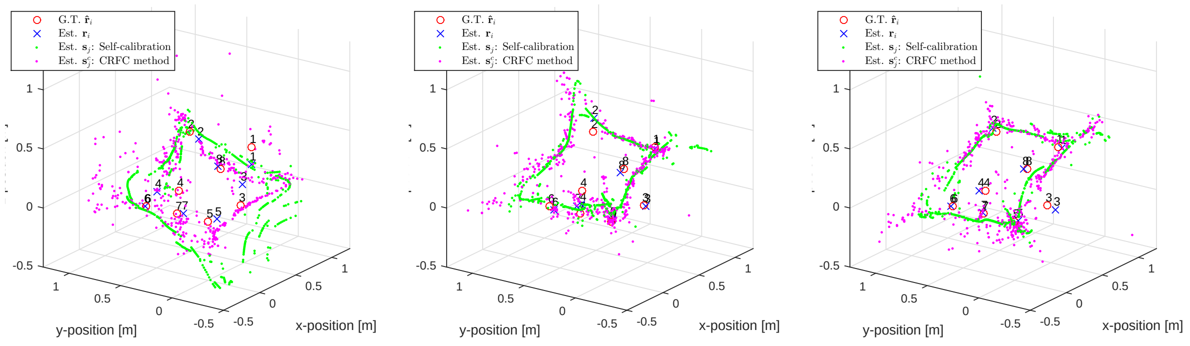
(b) The circumference difference errors.

(c) The area difference errors.

Figure 7.11: The means of the error measurements for each iteration when evaluating the virtual screen.

## 7.3 Sound source tracking

The result of the CFRC method in section 5.3 is a number of sound events $\mathbf{s}_j^c$, where the $c$ denoted them as CFRC. A few results can be seen in fig. 7.12 where they are compared to the $\mathbf{s}_j$ computed by the system self-calibration.



(a) Iteration 3 with $\mu = 0.331$, $\sigma = 0.269$ and RMS $= 0.426$.

(b) Iteration 9 with $\mu = 0.230$, $\sigma = 0.234$ and RMS $= 0.328$.

(c) Iteration 25 with $\mu = 0.239$, $\sigma = 0.302$ and RMS $= 0.386$.

Figure 7.12: Examples of estimated $\mathbf{s}_j^c$ for SPR440. Values in meters.

The experiments with the CFRC method had to be done with an extra large room volume as mentioned in section 5.3. For the exact same 40 iterations as in section 7.2.2 the script took about 2 full days to complete, far from a real-time solution. Nonetheless after running the 40 iterations, a visual inspection revealed another 9 iterations where the room volume had been to small and had generated artifacts in the results. Thus the error metrics were computed for a total of 31 iterations.

The error metrics are computed exactly in the same way as for section 7.2.3 with the same pre-computations. The average of the error distances were found to be (in meters)

$$\mu = 0.310, \quad \sigma = 0.293 \quad \text{and} \quad \text{RMS} = 0.429.$$

In fig. 7.13 the different means of the errors for each of the iterations can be studied in detail.

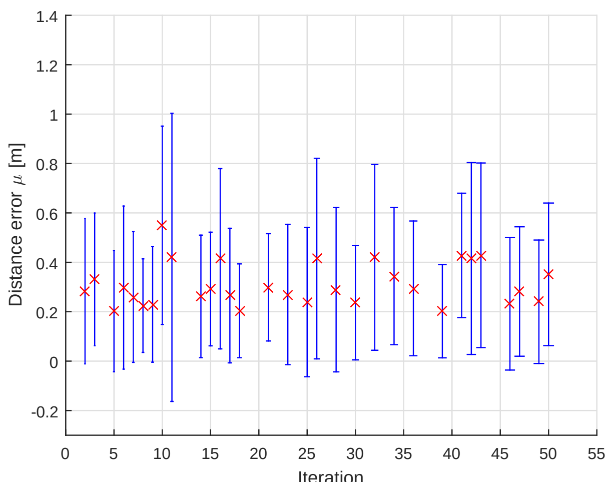


Figure 7.13: The mean values of errors with their standard deviations for the different iterations when evaluating $\mathbf{s}_j^c$.

Chapter 8

# Analysis

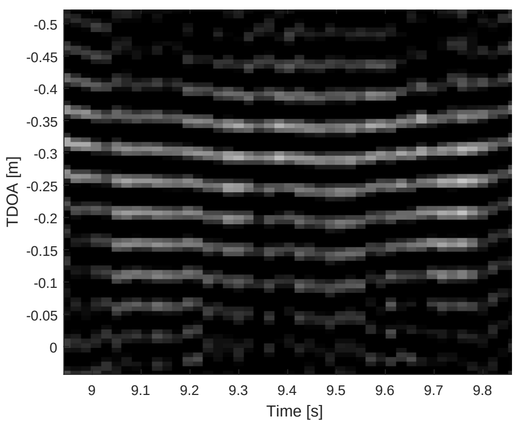
*This chapter presents the analysis of the results and evaluations. It is structured so that the sections in the previous chapter maps to the same sections in this chapter.*

## 8.1 Signal processing

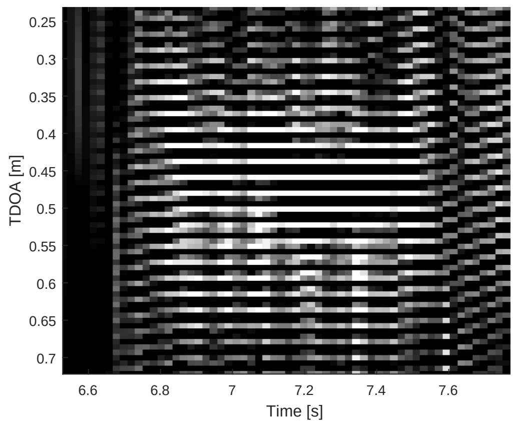
The results in table 7.1 confirms the visual findings in fig. 7.1. However there are some interesting points to make.

While the SPR signal method preforms significantly worse for 6 500 – 7 500 Hz than for 440 – 1 440 Hz when looking at $p$ the difference is not that larger with all channel pairs taken into account ($p_r$). This indicates that the refinement step of section 5.2.1 is quite robust. The same observation can be made for the CL signal method for the same frequency bands.

When comparing all the signal methods with each other it also seems like the CL signal method performs better than the periodic ones (PL and PS). One reason could be that the silence between the beeps results in non-existing or outlier values for $p$ and $p_r$, thus giving them a lower score when using the evaluation method.



(a) Close up of the SPR signal method for 6 500 – 7 500 Hz.



(b) Close up of the SPR signal method for 15 000 – 16 000 Hz.

Figure 8.1: A close up of the GCC-PHAT images of two different frequency spans for the SPR signal method.

Finally it is interesting to take a closer look at the GCC-PHAT images for some higher frequency bands. In fig. 8.1 a repetitive pattern can be seen, with more and tighter repetitions for the higher frequency band on the right-hand side of the figure. Measuring the distance between two repetitions gives a distance of $d_a = 7$ sample points for fig. 8.1a and $d_b = 3$ sample points for fig. 8.1b. Converting this to meters yields $d_a = 0.0496$ and $d_b = 0.0213$.

Most likely this has to do with aliasing. To avoid aliasing of the signal the distance $D$ between two microphones should be $D < \frac{\lambda}{2}$ [38] where $\lambda$ is the smallest wave length of the signal. The wave length is defined as $\lambda = \frac{v}{f}$ where $v$ is the signal propagation speed and $f$ is the frequency of the wave. Hence for a frequency band $[f_0, f_1]$ the distance between two microphones should be $D < \frac{v}{2f_0}$.

For $440 - 1\,440$ Hz this means that $D < 0.385$ meters for $v = 340$ meters per second. The configuration used in the experiment setup has distances that ranges between 0.3 meters to 0.92 meters. Thus it is no surprise that aliasing occurs for the higher frequency bands.

## 8.2 System self-calibration

### 8.2.1 Estimation of microphone positions $\mathbf{r}_i$

Naturally, it is very bad that the system is not robust enough to estimate $\mathbf{r}_i$ and $\mathbf{s}_j$ for all iterations. The algorithm implementations are the same as the ones in Zhayida et al. [10] uploaded at Zhayida et al. [37] so it is a pre-existing problem.

The average distance errors between the estimated microphone positions to the ground truth positions are $11.8 \pm 5$ centimetres. Considering that finger movements are to be tracked, the errors should be an order of magnitude smaller. Tot the greatest of results in other words.

Looking at fig. 7.6 it seems like there are two clusters of data points. The first cluster have its average errors located in the interval $5 - 12$ centimetres and the second cluster have its average errors in the interval $13 - 20$ centimetres. As to why this happens would be interesting to study further, but is unfortunately outside the time frame for this thesis.

Studying the first cluster, the average error distances would instead be $\mu_1 = 0.089$ meters with the average standard deviation $\sigma_1 = 0.041$ meters, while for the second cluster the values would be $\mu_2 = 0.157$ meters and $\sigma_2 = 0.064$ meters. The first cluster then have average distance errors of $8.9 \pm 4.1$ centimetres. The average RMS for cluster one is 9.8 centimetres.

For a comparison, numbers from two different papers can be found. In Kuang and Åström [15] an average distance error of 2.60 centimetres is obtained for the $\mathbf{r}_i$ and $\hat{\mathbf{r}}_i$. In Ask et al. [16] a RMS of 6.7 centimetres. The best average error distance in this thesis is found for iteration 25 with $4.8 \pm 1.6$ centimetres and an average RMS of 5.1 centimetres.

A concluding summary can be found in table 8.1.

| **Source** | **Avg. dist. error** | **Avg. $\sigma$** | **Avg. RMS** |
|---|---:|---:|---:|
| This thesis, all 40 iterations | 11.8 | 5 | 12.9 |
| This thesis, cluster one | 8.9 | 4.1 | 9.8 |
| This thesis, best iteration | 4.8 | 1.6 | 5.1 |
| Kuang and Åström [15] | 2.6 | - | - |
| Ask et al. [16] | - | - | 6.7 |

Table 8.1: A comparison between the different average distance errors between $\mathbf{t}_i$ and $\hat{\mathbf{r}}_i$. Values in centimetres.

### 8.2.2 Estimation of sound source path $\mathbf{s}_j$

An average distance error of $29.2 \pm 27.3$ centimetres is extremely high. Studying fig. 7.8 no clusters, like for the $\mathbf{r}_i$ distance errors, can easily be identified. The majority of the measurements lies relatively close together. However, whenever the standard deviation is “small” for an iteration for the $\mathbf{s}_j$ evaluation the corresponding $\mathbf{r}_i$ seems to be in its first cluster. Also the average distance error for $\mathbf{s}_j$ in that iteration tends to be among the lowest in fig. 7.8.

Inspecting the results visually in fig. 7.9 reveals that many of the estimated points $\mathbf{s}_j$ seems be mapped to the wrong ground truth line segment. This is a consequence of the algorithm applied in order to make the points comparable when calculating the error metrics. Look closer at the points in fig. 7.9a. Studying the leftmost line segment, a group of neighbouring points are coloured red (meaning that they should belong to the bottommost line). Likewise for a group of black points near the top of the line segment. Along the topmost line segment a lot of the points are instead blue, indicating that they should belong to the previous line segment.

The reason this happens is that the points are projected to their closest *line*, not the closest *line segment*. Imagine that the line segments extend outwards on both sides to infinity. Then it is not so strange that near an intersection the points could lie closer to the wrong line segment so to speak. Thus it is hard to make sense of the point-line mapping of the evaluation, which might give rise to errors in the error metrics.

There also seems to be a scaling problem in the system. For two of the images in fig. 7.9 the quadrilateral of points is a lot larger than the ground truth sound source path. This is also the case for a lot of the other images among the iterations. Why this is the case is unknown and it would be interesting to study the problem further.

### 8.2.3 Estimation of the virtual screen

The results from the virtual screen fitting is harder to analyse as the usual metrics are unavailable, due to the fact that for each iteration no “point clouds” are comparable. This gets harder when then algorithm seems to be far from perfect as only five good results is obtained out of 40, the remaining results being either bad fittings or complex quadrilaterals. As such, there can be a lot of improvements to the algorithm.

It is important to notice the fact that the virtual screen fittings are compared with the ground truth virtual screen arising from the ground truth corner points $\hat{\mathbf{c}}_k$ rather than the estimated points $\mathbf{s}_j$ themselves. The reason for this is that the points should only used to compute the virtual screen in an actual implementation, whether or not the vertical screen fits them perfectly is not as important. The algorithm should be robust in the sense that it works regardless if the estimation $\mathbf{s}_j$ is good or not.

In fig. 7.11 the three different error metrics are compared. When looking at the good iterations 15, 17, 21, 25 and 49 for the midpoint distance errors in fig. 7.11a it can be seen that all of them have small such errors. However, they are not exclusively small among all of the iterations. A lot of other iterations are small as well. So it is not a two-way implication.

For the circumference difference errors no conclusion can be drawn as they are spread along the y-axis. None of them are however that close to the bottom. Likewise for the area difference error. One reason could be that when marking results as “good” it is easy to choose virtual screens that represent the estimated $\mathbf{s}_j$ very well. None of the other virtual screens had good quadrilaterals however, so while there might be a bias it is not one of too great consequences for this analysis.

## 8.3 Sound source tracking

One thing that is easily noted when comparing the sound events $\mathbf{s}_j$ from the system self-calibration with the sound events $\mathbf{s}_j^c$ from the CFRC method is that $\mathbf{s}_j$ usually lies on a much smoother line. One reason could be that the this is one of the benefits when using all of the sound events to do the estimation. The CFRC only has access to the known positions of the microphones for eacj $\mathbf{s}_j^c$. The points mapped by the CFRC do however tend to follow the self-calibration points pretty well.

A comparison of the error metrics between the two methods can be found in table 8.2. The averages for the CFRC are a bit worse but they are not actually that far away from the ones for the system self-calibration. Using smoothing techniques together with some of the previous points $\mathbf{s}_{j-k}^c$ for some $k \in [1, j-1]$ could probably improve the results.

| **Method** | **No. iterations** | **Avg. dist. error** | **Avg. $\sigma$** | **Avg. RMS** |
|---|---|---|---|---|
| TDOA 2-step | 40 | 29.2 | 27.3 | 40.5 |
| SRP-PHAT with CFRC | 31 | 31.0 | 29.3 | 42.9 |

Table 8.2: A comparison between the sound events $\mathbf{s}_j$ from the TDOA 2-step method and the sound events $\mathbf{s}_j^c$ from the CFRC method. Values in centimetres.

For an actual implementation, instead of a proof-of-concept, the room volume problem would need to be handled as well. It could probably be set dynamically in some way using the system self-calibration with the virtual screen as a base measure. Hopefully this would save a lot of computing time as well.

Chapter 9

# Conclusions and future work

*In this chapter the work of this thesis is summarized. The main findings from the results, evaluations and analysis are presented briefly. Also some thoughts about what angles to pursue next are put into words.*

## 9.1 Conclusions

The main goal of this thesis was to build a proof-of-concept of a system where a user could interact with the computer using sound waves. A full pipeline have been implemented, where a synchronized array of microphones placed ad hoc can be self-calibrated to the degree that a virtual screen can be fitted. Multiple signal sources emitting generated sound waves can be moved around and then separated at the receiving end. Finally the positions of the sound sources can be tracked. Thus all of the goals have been reached in some way, except that the system can not handle frequencies outside the audible spectrum.

Out of the four different signal generation methods the most complicated method turned out to be the best one, beating the others by a large margin. Although the method used to estimate the TDOA values was very robust it could not handle frequencies above a certain threshold due to aliasing in the GCC-PHAT. The system runs well enough on lower frequencies however.

The system self-calibration did manage to to a scene reconstruction for 80% of the test cases. It did not show as good results as previous papers and it was hard to evaluate due to the lack of a proper lab setting. For all of the test cases the estimations yielded an average distance error of $11.8 \pm 5$ centimetres for the receivers and and average distance error of $29.2 \pm 27.3$ centimetres for the sound events generated by the moving sound source. The best test case did however beat a previous RMS score for the microphones. The implementation seem to have a scaling error for some unknown reason.

A new technique for estimating a so called *virtual screen* was introduced. The initial findings show that the method is not very robust. The algorithm resulted in good estimation for 12.5% of the test cases. The error metrics did not reveal that much information either and could be improved upon.

Finally the real-time sound source tracking turned out to be not so real-time for the implementation of the CFRC. Mainly because of the badly chosen search volumes. Compared to the TDOA 2-step method in the system self-calibration it preformed worse, although with a very small margin.

## 9.2 Future work

There are a lot of things that could be improved upon in this thesis, both regrading techniques and implementation details. Another thing that would be interesting to see is an implementation in a programming language suited for real-time processing. The different parts of the system could then be compared to an actual use case, “for real”.

The main thing to explore further would be to make the system self-calibration robust for higher frequencies. Right now the techniques pretty much ignore many of the distance requirements regarding aliasing due to the ad hoc approach of the receiver placements. The GCC-PHAT might not be the best suited method for the task at hand. An alternative would be to study its settings further.

Another big thing would be to pay closer attention to the implementation of the different estimators used to find the scene reconstruction. Remove the scaling errors and handle the problems arising from using different dimensionalities in the microphone setup and the sound source path. Also some sort of recovery procedure would be need to handle the cases where the estimators are unable find any good solutions.

The virtual screen can be improved upon in many ways as previously mentioned. Everything from handling complex quadrilaterals to review the order of subsystems. Maybe fit the lines first and then project them onto a plane? Finally it would be interesting to combine the system-self calibration with the SRP-PHAT with CFRC to dynamically improve the search volume for the CFRC algortihm. Right now the only relationship between the techniques is that the second one uses the estimated microphones positions. Making the CFRC SRP-PHAT aware of previous results to use smoothing and extrapolation techniques would be interesting to see as well.

# Bibliography

[1] Steven Spielberg (Director). Minority report. DreamWorks Pictures, 2002.

[2] James Cameron (Director). Avatar. 20th Century Fox, 2009.

[3] Tao Hongyong and Yu Youling. Finger tracking and gesture recognition with kinect. In *2012 IEEE 12th International Conference on Computer and Information Technology*, pages 214–218, October 2012.

[4] Shang Ma, Qiong Liu, Chelhwon Kim, and Phillip Sheu. Lift: Using projected coded light for finger tracking and device augmentation. In *2017 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 153–159, March 2017.

[5] Daniel Popa, Vasile Gui, and Marius Otesteanu. Real-time multi-cue finger tracking for human computer interaction. In *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, pages 1–7, July 2015.

[6] Noor Shaker and M. Abou Zliekha. Real-time finger tracking for interaction. In *2007 5th International Symposium on Image and Signal Processing and Analysis*, pages 141–145, September 2007.

[7] Alexander Ens, Leonhard M. Reindl, Joan Bordoy, Johannes Wendeberg, and Christian Schindelhauer. Unsynchronized ultrasound system for tdoa localization. In *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 601–610, October 2014.

[8] Alexander Ens, Leonhard M. Reindl, Thomas Janson, and Christian Schindelhauer. Low-power simplex ultrasound communication for indoor localization. In *2014 22nd European Signal Processing Conference (EUSIPCO 2014)*, pages 731–735, September 2014.

[9] Masatoshi Matsumoto, Kazumasa Kaneta, Miyabi Naruoka, Hiroshi Tanaka, and Kosuke Takano. Tracking positions of human body parts based on distance measurement with sound wave. In *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 514–518, March 2017.

[10] Simayijiang Zhayida, Fredrik Andersson, Yubin Kuang, and Kalle Åström. An automatic system for microphone self-localization using ambient sound. In *22nd European Signal Processing Conference (EUSIPCO 2014)*, pages 954–958, September 2014.

[11] Simon Segerblom Rex. Robust time difference estimation for unknown microphone positions with reverberation. Master's thesis, Lund University, 2015.

[12] Hoang Do. Real-time srp-phat source location implementations on a large-aperature microphone array. Master's thesis, Brown University, 2009.

[13] Thomas D. Rossing, editor. *Springer Handbook of Acoustics*. Springer, 2007.

[14] Hans Dieter Lüke. The origins of the sampling theorem. *IEEE Communications Magazine*, 37(4): 106–108, April 1999.

[15] Yubin Kuang and Kalle Åström. Stratified sensor network self-calibration from tdoa measurements. In *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5, June 2013.

[16] Erik Ask, Yubin Kuang, and Kalle Åström. A unifying approach to minimal problems in collinear and planar tdoa sensor network self-calibration. In *22nd European Signal Processing Conference (EUSIPCO 2014)*, pages 1935–1939, September 2014.

[17] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24 (6):381–395, June 1981.

[18] Mikael Swartling. *Direction of Arrival Estimation and Localization of Multiple Speech Sources in Enclosed Environments.* PhD thesis, Blekinge Institute of Technology, 2012.

[19] Joseph Hector DiBiase. *A high-accuracy, low-latency technique for talker localization in reverberant environments using microphone arrays.* PhD thesis, Brown University, 2000.

[20] Jacob Benesty. Adaptive eigenvalue decomposition algorithm for passive acoustic source localization. *The Journal of the Acoustical Society of America*, 107(1):384–391, January 2000.

[21] Michael Bradstein and Darren Ward, editors. *Microphone arrays: Signal Processing Techniques and Applications.* Springer-Verlag, 2001.

[22] Stanley T. Birchfield. A unifying framework for acoustic localization. In *12th European Signal Processing Conference (EUSIPCO 2004)*, pages 1127–1130, September 2004.

[23] Hoang Do and Harvey F. Silverman. A fast microphone array srp-phat source location implementation using coarse-to-fine region contraction (cfrc). In *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 295–298, October 2007.

[24] Hoang Do, Harvey F. Silverman, and Ying Yu. A real-time srp-phat source location implementation using stochastic region contraction (src) on a large-aperture microphone array. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '07)*, pages 121–124, April 2007.

[25] 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, August 1976.

[26] G.C. Carter, A.H. Nuttall, and P.G. Cable. The smoothed coherence transform. *Proceedings of the IEEE*, 61(10):1497–1498, October 1973.

[27] Cha Zhang, Dinei Florencio, and Zhengyou Zhang. Why does phat work well in lownoise, reverberative environments? In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '08)*, pages 2565–2568, March 2008.

[28] Michael S. Brandstein and Harvey F. Silverman. A robust method for speech signal time-delay estimation in reverberant rooms. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, volume 1, pages 375–378, April 1997.

[29] Martin Byröd, Klas Josephson, and Kalle Åström. Fast and stable polynomial equation solving and its application to computer vision. *International Journal of Computer Vision*, 84(3):237–256, September 2009.

[30] Marc Pollefeys and David Nister. Direct computation of sound and microphone locations from time-difference-of-arrival data. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '08)*, pages 2445–2448, March 2008.

[31] Simayijiang Zhayida, Simon Segerblom Rex, Yubin Kuang, Fredrik Andersson, and Kalle Åström. An automatic system for acoustic microphone geometry calibration based on minimal solvers. *arXiv preprint arXiv:1610.02392*, October 2016.

[32] Åke Björck. *Numerical Methods for Least Squares Problems.* Society for Industrial and Applied Mathematics, 1996.

[33] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, September 1936.

[34] Kenneth Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, July 1944.

[35] Donald Marquardt. An algorithm for least-squares estimation of nonlinear parameter. *SIAM Journal on Applied Mathematics*, 11(2):431—441, June 1963.

[36] Peter H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, March 1966.

[37] Simayijiang Zhayida, Simon Segerblom Rex, Yubin Kuang, Fredrik Andersson, and Kalle Åström. Structure from sound. GitHub, 2016. `https://github.com/kalleastrom/StructureFromSound` [Accessed: 2017-09-01].

[38] Jacek Dmochowski, Jacob Benesty, and Sofiène Affès. On spatial aliasing in microphone arrays. *IEEE Transactions on Signal Processing*, 57(4):1383–1395, April 2009.