# Estimation and correction of deviations between camera and audio images in acoustic imaging

Axel Rémi Hamran


Thesis submitted for the degree of
Master in Electrical Engineering, Informatics and Technology
(Signal Processing)
60 credits

Department of Physics
Department of Informatics
Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Spring 2020

# Estimation and correction of deviations between camera and audio images in acoustic imaging

Axel Rémi Hamran

Estimation and correction of deviations between camera and audio images in acoustic imaging

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

# Abstract

Acoustic imaging allows the detection and localisation of sound sources in industrial applications. Commonly, acoustic imaging involves combining an array of microphones with a digital camera module. While the camera module creates images with a lens and an array of light sensors, the microphone array creates audio images through beamforming, estimating audio level in a pixel grid. Locating sound sources then involves overlaying the audio image onto the camera image. Proper alignment between the two images is crucial for correct location of the sound source. This thesis proposes a low-complexity setup for calibrating the alignment between the camera and audio images. A sound source with a known colour is placed in front of an acoustic imaging array, playing a known signal. Sound source location is then found using both the digital camera and the microphone array. By combining several recordings of the sound source in different locations, we can measure any differences in the alignment between the camera and audio images. Alignment errors can be overcome with a least squares estimator used in estimating camera sensor offset and camera rotation. The offset and rotation is applied to the camera image giving near perfect alignment.

# Acknowledgements

First of all i would like to thank my thesis advisor Professor Sverre Holm of the University of Oslo, and my advisors Ines Hafizovic, and Bjørn Braathen of Squarehead Technology for being patient with me and giving me plenty of feedback and guidance on my ideas and my work.

I would also like to thank Squarehead Technology in general for not only providing me with this thesis but also providing me with a great place to work, copious amounts of coffee, as well as for being full of amazing people.

Also a big thanks to my parents Svein-Erik Hamran and Isabelle Lecomte for helping in proofreading my thesis and giving valuable feedback on its structuring and writing.

Finally a thank you to my fiancée Miriam Kosther for supporting me and being so patient, even if the final stretch of writing ended up taking all summer.

Author

Axel Rémi Hamran

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Locating sound sources

An unexpected noise can signify several different things, for instance a noise can reveal an issue with a piece of equipment or machinery indicating a need for inspection, and possibly repair. In addition the noise itself might be the issue, either violating local noise regulations for work or residential areas, or by leaking through what is supposed to be a noise reducing measure. In all of these cases locating the source of the noise is the first step of action. However locating the source of a noise is not always easy with human ears. Professional sound monitoring equipment exists in the form of sound level meters, such as the one in Figure 1.1. This equipment can produce an accurate estimate of sound level, and estimate signal parameters. However to find the source of the noise, manually searching is the only option, which might be difficult, especially in environments rich with other noise sources.

Figure 1.1: Sound level metre used to measure noise pollution, building acoustics and environmental noise. Image taken with permission from norsonic.com

An array of microphones like the ones in figure 1.2 is able to modify the direction it is sensitive to sounds through delays and weighting of the audio on each microphone. By estimating the sound at different positions in front of the array we are able to produce an image with an estimated sound level for different directions. By overlaying this acoustic image over a camera image like in figure 1.3 we are able to locate a sound source easily.


Figure 1.2: Different arrays used in acoustic imaging

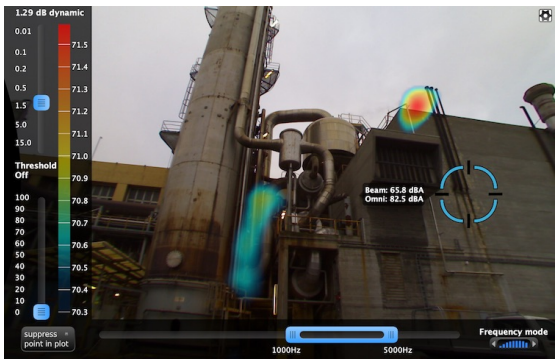
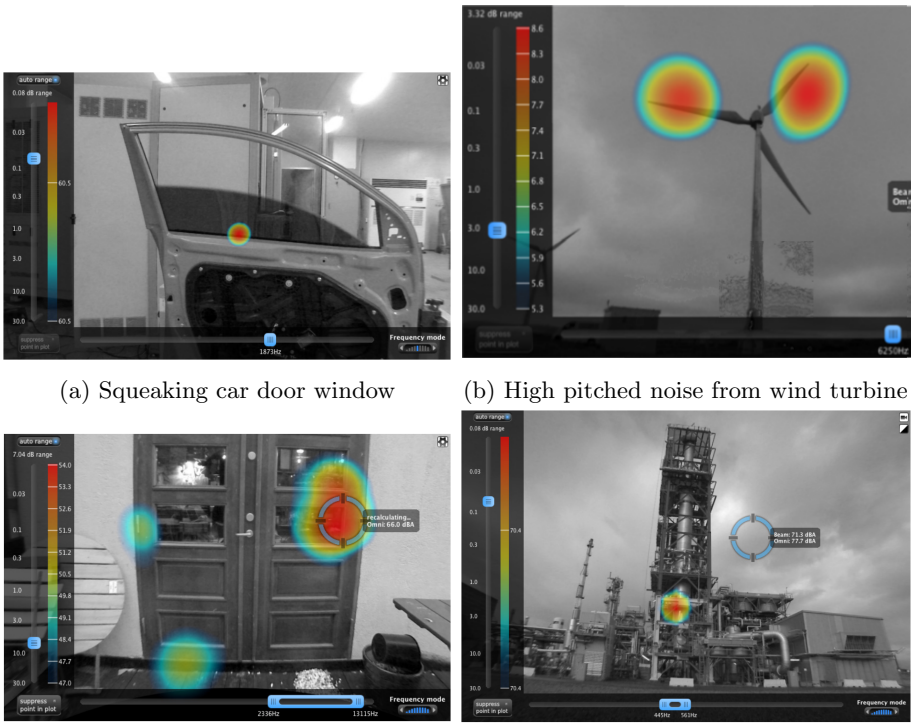



Figure 1.3: Overlaying an audio image over the camera to pinpoint the location of a high-frequency noise source.

Examples of use cases for this are illustrated in figure 1.4 including undesired noises from mechanical devices, such as a car door emitting a loud squeeking noise when the window is operated (1.4a), and a wind turbine causing high pitched noises due to wind interaction with its blades (1.4b). Other uses include identifying undesired noise leakage, such as sound leaking through the doors of a nightclub, despite the venue recently having renovated to avoid sound leaking (1.4c), and high pressure pipes emitting loud noise near a residential area, violating local noise ordinances (1.4d). In all of these cases, properly locating the noise source is vital in order to correctly apply repairs or changes.



(a) Squeaking car door window

(b) High pitched noise from wind turbine

(c) Sound leakage from nightclub

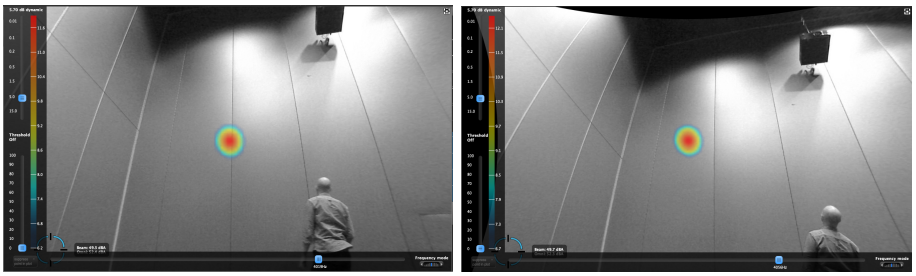
(d) Sound leakage from pipes

Figure 1.4: Different scenarios where locating the source of a noise is of high interest.

## 1.2 Sound source localisation

Microphone arrays used for acoustic imaging applications demand highly precise localization of the sound sources. Most commercial systems superimpose an acoustic image on top of a video image, and a spatial alignment of the images is decisive for the correct assignment of the sound level to the proper sound source location.

The required precision in the alignment of the camera and audio image is set by the resolution of the camera, and by the properties of the applied beamformer and the microphone array. Large microphone arrays have typically a higher localization precision, and a mismatch between the sound and image planes will therefore have serious impact on accuracy. Ultimately, it can lead to false positioning of sound sources. False positioning could mean large amounts of time spent solving the problem in the wrong location. Figure 1.5 shows an image of a recording performed to locate sound leakage from what is supposed to be an acoustic insulating separator wall between conference rooms. A displacement in image would lead to applying additional insulation to the wrong parts of the wall, wasting time and resources.



(a) No error induced (b) Acoustic image displaced

Figure 1.5: Searching for audio leakage in a separating wall. Displacement between the image, and audio layer can give a misleading impression of the location of a sound.

### 1.2.1 Audio image alignment

The alignment of the audio and image planes is normally done by a manual calibration of the camera, done as the final manufacturing step. The audio image is then mapped directly onto the camera image, assuming the camera is properly positioned and calibrated. However, changes in the physical placement of the camera relative to the microphone array can occur, for example due to field usage and transportation. The changes in camera positioning could be to the entire camera module, displacing and rotating the camera, or changes to the imaging sensor's position in relation the camera lens may occur. Moving the sensor displaces the image, and changes the effect of lens distortion on the image. Thus a small change to the alignment between image sensor and lens, can result in a large effect on the alignment between the camera image and the audio plane.

This rises a need for an automatized procedure for detection and correction of the misalignment between the sound and the image plane. For practical purposes, the solution should not pose a requirement of a very precise calibration rig, nor a special environment, such as anechoic chamber. Ideally the calibration should be doable in the field, using as little equipment as possible. The motivation is to relatively easily be able to test an acoustic imaging system for this type of error, and to algorithmically compensate and correct the detected deviations.

## 1.3 Calibration

In order to detect and measure alignment errors between the camera image and the sound plane, positional information is needed from both. The solution would be to have an object be detected using image processing to get a position in the camera image, and signal processing to get a position in the audio plane. Such an object would be an audio source playing a known signal, that can be visually detected in the image. Thus arises the first two challenges; find a technique to detect the audio source visually, using the camera, and acoustically using the microphones, with as few equipment constraints as possible.

Further, a technique will be needed to estimate and correct for any misalignments detected between the video and the audio. Such a technique will involve re-defining the mapping of the audio image onto the camera image, by minimising the estimated error. Thus the complete calibration requires a setup enabling detection as well as three computational steps::

- Camera image positioning
- Audio plane positioning
- Estimating error and remapping the audio onto the camera image.

# Chapter 2

# Sensors

This chapter contains a short description of the microphone array used in all experiments contained in this thesis. All measurements in this thesis were done using the "Hextile - Acoustic Camera" microphone array (Shown in Figure2.1) produced by Squarehead Technology and sold by Norsonic. The array designed for industrial measurements using the accompanying "Norsonic Acoustic Camera" software to record audio and video data. The recorded data are then split up into separate image frames and audio channels using Squareheads internal python-API, and processed using python.

Figure 2.1: The Hextile Acoustic Camera microphone array used in experiments

The array can also be combined with two other identical arrays to create a larger multitile aperture, enabling lower frequency operations, as shown in figure 2.2.

Figure 2.2: Different multitile configurations allowing low- and very low-frequency operation.

## 2.1 Microphone array

The microphone array itself is a planar array composed of 128 MEMS (MicroElectrical-Mechanical System) microphones mounted on an underlying PCB (Printed Circuit Board). The array is laid out in a spiral pattern forming approximately a hexagon along its edges, as shown in Figure 2.3. The microphones each have a sample rate of 44.1 kHz. The average spacing between Microphones is about 2.8 cm, and the outer diameter is approximately 42 cm. Table 2.1 lists some of the basic properties of the microphone array, as well as the properties of its microphones. Figure 2.4 shows the Half-Power beam-width for the different array configurations. This thesis only involves the single array configuration, and is thus limited to using frequencies above 410 Hz, preferably above at least 1 kHz to have achieve good accuracy. All single frequency and narrow bandwidth recordings done in this thesis containing a single audio source are focused around 4 kHz. The beampattern of the microphone array at 4 kHz is shown in figure 2.5

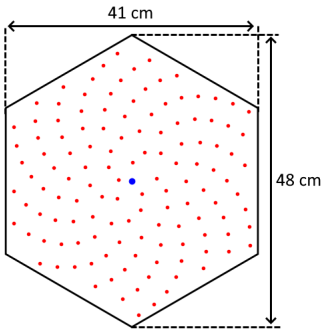



Figure 2.3: Front illustration of the Hextile microphone array, showing microphone positions in red, and camera in blue.

| | |
|---|---|
| Microphones | 128 MEMS microphones |
| Sampling rate | 44.1 kHz |
| Max sound level | 128 dBA |
| Min sound level (system) | 9 dBA |
| SNR per microphone | 65 dBA |
| SNR array | 82 dBA |
| Dimensions(width,height, max radius) | 41 cm x 48 cm, Ø 48 cm |
| Communication interface | USB |

Table 2.1: Specifications of the Hextile microphone array. ("Acoustic Camera, Norsonic", n.d.)

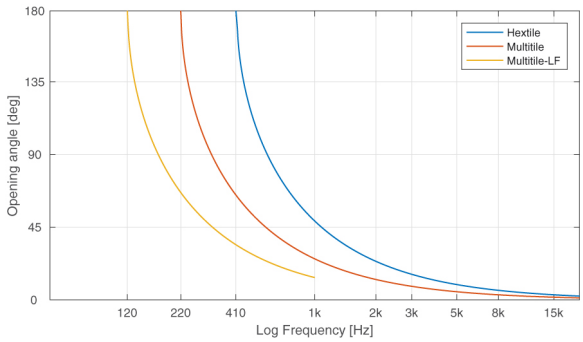



Figure 2.4: Approximate Half-Power beam-width at different frequencies, for different array configurations.
The blue line (Hextile) represents the characteristics of a single array

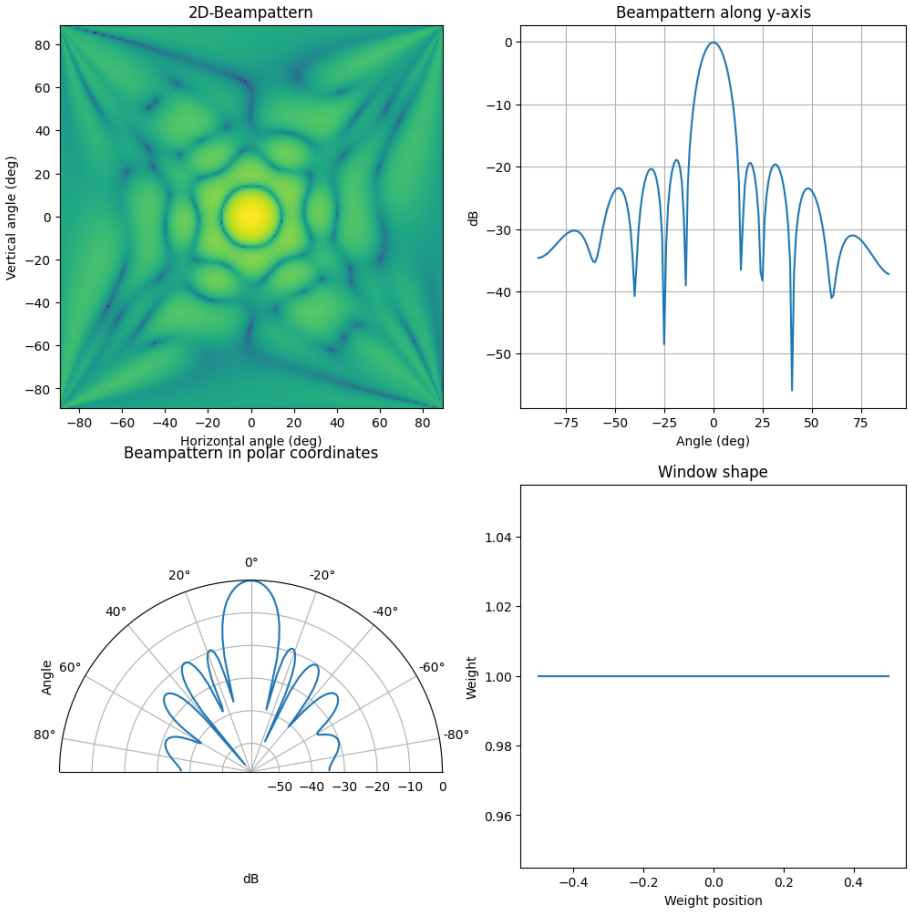



Figure 2.5: Beampattern for Hextile at 4 kHz with no Apodization

## 2.2 Camera

The camera module features a 2592 x 1944 sensor with an ultra-wide-angle or fisheye lens, with a horizontal viewing angle of 105°. It records video at 15 FPS in a Motion JPEG format in which each frame is a single JPEG compressed image. Table 2.2 contains a summary of the key parameters of the camera. Figure 2.6 show a schematic view of the lens used for the camera.

| | |
|---|---|
| Sensor Module | Omnivision OV5640 |
| Resolution | 2592 x 1944 |
| Pixel size | 1.4 $\mu$m x 1.4 $\mu$m |
| Sensor image area | 3673.6 $\mu$m x 2738.4 $\mu$m |
| Frame rate | 15 FPS |
| Video format | Motion JPEG |
| Horizontal opening angle | 105° |
| Lens size | 6.35 mm (1/4 ") |
| Focal length | 2.15 mm |
| F Number | 2.4 |

Table 2.2: Camera specifications

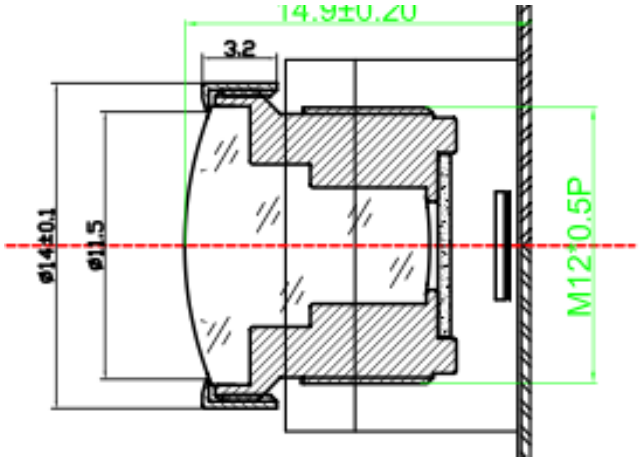



Figure 2.6: Schematic view of the lens for the camera in the Hextile microphone array

# Chapter 3

# Theory

This chapter gives a brief theoretic introduction to various topics needed to understand the methods described in this thesis. The chapter is divided into image processing and signal processing.

In the image processing segment an overview of the process of correcting for lens distortion is described, as well as a description of the sensor offset parameter used to describe the imaging sensors relative position to the lens. Further discussed is the background for the object detection and positioning in the camera image, using the HSV (Hue Saturation Value) colour-space and segmentation to separate an object from the background. Positioning an object is explained through raw image moments and the centroid of the object. Finally a short description of the affine and perspective transforms that may be used to re-map audio data onto the image.

In the audio processing there is a short description of beamforming as well as some central terminology and considerations such as; sampling criteria, resolution, far/near-field, windowing, and a short description of some common beamformers.

## 3.1 Geometric coordinate transforms

### 3.1.1 Rotation matrices

A Rotation matrix is a matrix with the property that any vector multiplied with it is rotated about an axis with a certain angle. Equations 3.1, 3.2, and 3.3 show the rotation matrices for rotating around the x, y, and z axis respectively. All these matrices can be combined through multiplication into a single rotation matrix performing all rotations, as shown in equation 3.4.

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{3.1}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{3.2}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

$$R = R_x(\alpha)R_y(\beta)R_z(\gamma) = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma - \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{3.4}$$

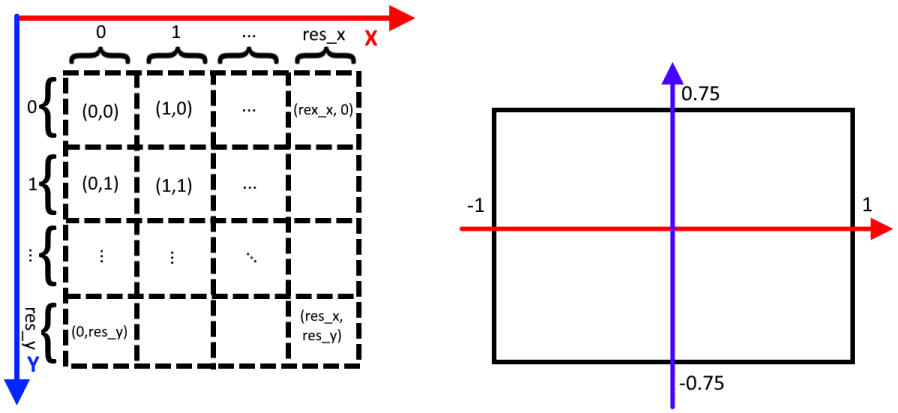
## 3.2 Image processing

### 3.2.1 Image coordinate systems

The most common coordinate system used in digital images is pixel coordinates. In such a system all pixel values are arranged in a grid of columns and rows, and each pixel has a single integer coordinate associated with it. The origin is placed in the upper left hand corner of the image with the x-axis pointing right and the y-axis pointing down. This is due to the convention of having the z-axis point along the cameras optical axis in digital imaging, and thus to conserve a right handed coordinate system the y axis is inverted, pointing down instead of up. Figure 3.1a contains an illustration of how matrix/pixel coordinates work.

An alternative is a normalized image coordinate system, in which coordinates lie between -1 and 1. The origin is located at the centre of the image and 1 is defined as the largest extent the image has from the centre. For a landscape image the largest extent is horizontal. As an example, an image with a size of 100x75 pixels will range from -1 to 1 along the x-axis and range from -0.75 to 0.75 in the y axis. Figure 3.1b shows an illustration of a normalized camera coordinate system.

The final coordinate system relevant for this thesis is to project the image into the world coordinate system and measure using world units instead of image units. This allows us to discuss the positions and distances between objects in terms of meters or centimetres instead of pixels.




(a) Coordinate system of a digital image (b) Normalized image coordinate system with all positions between -1 and 1

Figure 3.1

### 3.2.2 Distortion correction of wide-angled lens

When a camera captures an image it turns what it sees into a 2d representation of the world in front of it. In order to associate an objects location in 2d to its 3d world position we need to know the cameras intrinsic and extrinsic parameters. The extrinsic parameters describe the cameras position and rotation relative to the world coordinate system. The intrinsic parameters describe the fixed internal construction of the camera, such as it's focal length, image sensor size, and image sensor location.

The cameras extrinsic parameters can be described by a rotation matrix $R$, (like the one in equation 3.4) rotating from the world coordinate system to the cameras coordinate system. This is then followed by a translation vector $T$, translating in the camera coordinate system to the cameras position relative to the world origin. The resulting extrinsic matrix $M_{ext}$ is shown in equation 3.5. Multiplying a vector with the extrinsic matrix gives us the same vector in the cameras coordinate system.

$$M_{ext} = \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & T_1 \\ r_{21} & r_{22} & r_{23} & T_2 \\ r_{31} & r_{32} & r_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

For our purposes the cameras coordinate system and the arrays coordinate system share the same origin, however the orientation might be a source of alignment error between the audio image and camera image.

The intrinsic parameters for a pinhole camera can be represented by the matrix in equation 3.6, where $f$ is the focal distance of the lens, and $s_x$ and $s_y$ is the pixel density along the x and y axes.

$$M_{int} = \begin{bmatrix} f/s_x & 0 & c_x \\ 0 & f/s_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

**Sensor offset**

Here $c_x$ and $c_y$ defines the principal point, that is to say where the optical axis intersects the imaging sensor. Ideally the principal point is located in the centre of the imaging sensor, but due to imprecisions in manufacturing and impacts during transportation and handling, the sensor might have a not insignificant offset from its intended location. As such the microphone arrays used in this thesis includes a sensor offset to compensate for the sensors true location behind the lens. This offset is normally estimated using a manual calibration step during the manufacturing of the array.

**Radial barrel distortion**

Due to the wide angled nature of the lenses used in acoustic imaging application the images produced contain a significant amount of barrel- or radial distortion. Radial distortion magnifies the centre of the image while demagnifying the edges. Thus we need to expand the cameras intrinsic parameters with equations 3.8 and 3.9, to compensate for radial distortion, where x and y are normalized pixel coordinates relative to the optical centre. The intrinsic parameters $k_1$, $k_2$, and $k_3$ need to be estimated through a calibration involving multiple views of a known plane such as a chessboard. Figure 3.2 shows the effect of barrel distortion on a grid.

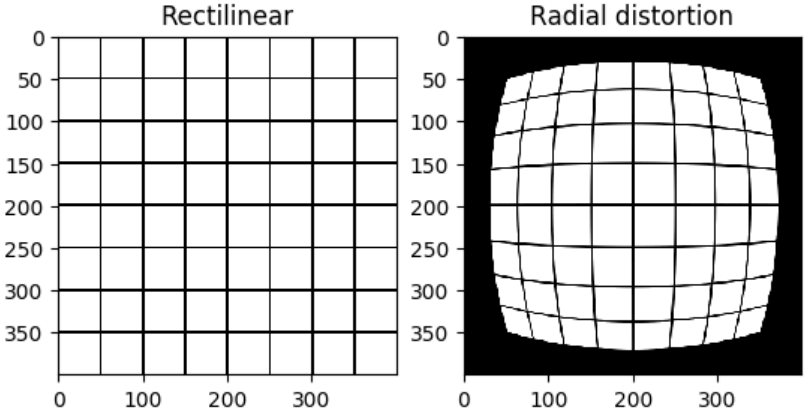



Figure 3.2: Image with and without radial distortion

$$r^2 = x^2 + y^2 \tag{3.7}$$

$$x_{distorted} = x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{3.8}$$

$$y_{distorted} = y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \tag{3.9}$$

#### Fisheye OpenCV model

For ultra-wide wide angle lenses, also known as fisheye lenses, the model described above does not sufficiently describe the lens distortion. OpenCV also provides a fisheye lens calibration model based on (Kannala & Brandt, 2006). This model is the one used for correcting the lens distortion in this thesis.

**Note** For the purposes of this thesis it is assumed that all camera intrinsic parameters have been estimated and are known. The exception being the centre offset/location of the optical axis, which is assumed to be a possible source of error for sound and image alignment. For more in-depth reading on camera calibration see (Kaehler & Bradski, 2016, ch. 18) and (Kannala & Brandt, 2006)

### 3.2.3 HSV colour space

To simplify segmentation by colour it is useful to convert our images into a colour space where it is more easy to describe a specific colour and to define what counts as that colour for purposes of segmentation. HSV (hue, saturation, value) redefines the RGB colour space to be instead of 3 basis vectors describing relative value of a specific colour (red, green, blue) to instead describing a colour by its hue in a circular axis with red at 0°, green at 120°, blue at 240°, and red again at 360°. [1] The saturation describes the purity of the colour with respect to its distance from grey, and the value its separation from pure black.

#### Conversion from RGB to HSV

(From OpenCVs documentation "OpenCV: Color Conversions", n.d.)

Equations 3.10, 3.11, and 3.12 describe the process to compute value, saturation and hue respectively, using RGB values normalized between 0 and 1.

$$V = max(R, G, B) \tag{3.10}$$

$$S = \begin{cases} V - \min(R, G, B) & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

$$H = \begin{cases} \frac{60(G-B)}{V-\min(R,G,B)} & \text{if } V = R \\ 120 + \frac{60(B-R)}{V-\min(R,G,B)} & \text{if } V = G \\ 240 + \frac{60(R-G)}{V-\min(R,G,B)} & \text{if } V = B \end{cases} \tag{3.12}$$

### 3.2.4 Segmentation by colour

Segmentation is essential in detecting mono-coloured objects, allowing the creation of binary masks that can be used for further processing. To segment an image, specify some boundaries within the current colour space and mark any pixels with values within the bounds as 1 and otherwise 0. To detect coloured

[1] In openCV the hue axis is halved to fit within one byte going instead from 0° to 180°.

objects in the HSV colour space, we set the boundaries in the hue axis to be the desired colour ± some tolerance, and for the saturation and value we set it to 100% minus some tolerance. The tolerances need to be adjusted based on the lighting of the scene in the image as well as possible colour inaccuracies in the camera and the colour of the object. Figure 3.3 shows an image of a red object before and after a colour based segmentation is applied.

Figure 3.3: Image before and after applying segmentation.

### 3.2.5 Morphological image processing

(Based on Gonzalez, 2008, pp. 628 - 639)

Morphological image processing involves modifying binary images, often binary masks defining an object. Morphological transforms can be applied to shrink or expand objects in binary images. The following pages contain a short description of four morphological operations relevant to the image processing performed in this thesis.

#### Structuring element

All morphological operations require a basic pre-defined binary shape called the structuring element. Typical examples of this is a 3x3 box of 1s or a cross/plus. Typically, the origin of the structuring element is the centre.

#### Erosion

In erosion the structuring element is convolved over the image and all pixels where the structuring element is completely encompassed in the shape are marked. This means that all positions for B in the image where B is a subset of A, are kept marked. Equation 3.13 describes erosion, where $z$ is a point in the image plane, and $B_z$ is the structuring element B displaced so that its origin is at $z$. Figure 3.4 shows a binary shape before and after a 3x3 structuring element is used to erode it.

$$A \ominus B = \{z|(B_z) \subseteq A\} \tag{3.13}$$

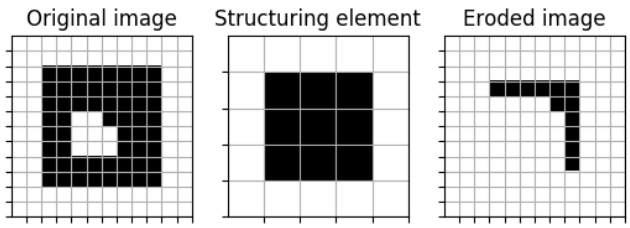



Figure 3.4: Example of an eroded image

### Dilation

In dilation the structuring element is convolved over the image and all pixels where the structuring element is partially or fully encompassed in the shape are marked. This means that all positions for B where B partially or fully intersects A, are marked. Equation 3.14 describes erosion, where $z$ is a point in the image plane, and $B_z$ is the structuring element B displaced so that its origin is at $z$. Figure 3.5 shows a binary shape before and after a 3x3 structuring element is used to dilate it.

$$A \oplus B = \{z | B_z \cap A \neq \varnothing\} \tag{3.14}$$

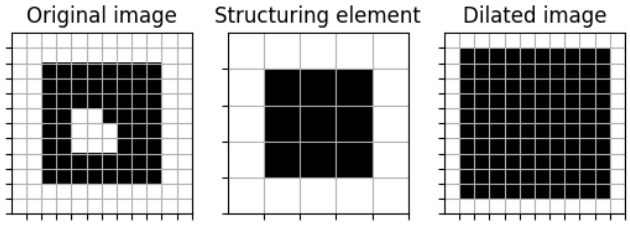



Figure 3.5: Example of a dilated image

### Opening

Opening consists of erosion of A by B followed by a dilation by B. Opening generally smooths the contours of objects and breaks thin bridges and removes objects smaller than the structure element while maintaining the size of other objects. Equation 3.15 shows openings relation to erosion and dilation, and figure 3.6 shows the effect of opening applied on the same binary shape as earlier.

$$A \circ B = (A \oplus B) \ominus B \tag{3.15}$$

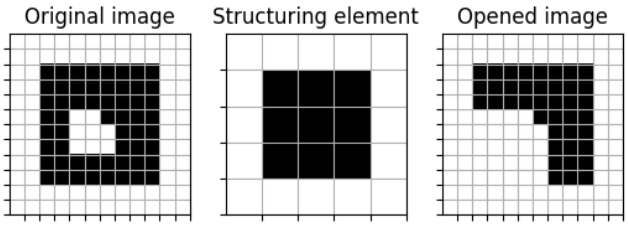



Figure 3.6: Example of an opened image

**Closing**

Closing consist of dilation followed by erosion. Closing, like opening also generally smooths objects, however it also closes small holes and bridges nearby shapes fusing them together. Equation 3.16 shows closings relation to erosion and dilation, and figure 3.7 shows the effect of closing applied on the same binary shape as earlier.

$$A \bullet B = (A \ominus B) \oplus B \tag{3.16}$$

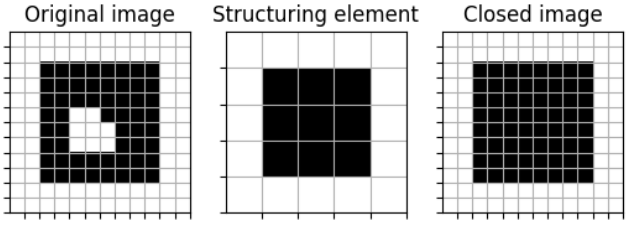



Figure 3.7: Example of a closed image

### 3.2.6 Raw image moments

An image moment is the weighted average over the images intensity values. It can be used to estimate and objects position and orientation. Equation 3.17 describes how to compute a raw image moment $M_{ij}$ of the i-th and j-th degree.

$$M_{ij} = \sum_{x} \sum_{y} x^i y^j I(x, y) \tag{3.17}$$

Where $I(x, y)$ is the intensity value at position $(x, y)$ (Gonzalez, 2008, p. 839) .

### 3.2.7 Centroid of shape

The centroid of a shape is the geometric centre of the shape, the mean of all the points in the shape. So, if a shape is composed of $n$ distinct points/pixels $x_1 \dots x_n$, then the centroid $x_c$ can be described as in equation 3.18

$$x_c = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.18}$$

Alternatively, we can find the centroid using image moments as shown in equation 3.19 where $C_x$ and $C_y$ are the x and y coordinates of the centroid, respectively, and $M_{ij}$ is the image moment given by equation 3.17

$$\begin{aligned} C_x &= \frac{M_{10}}{M_{00}} \\ C_y &= \frac{M_{01}}{M_{00}} \end{aligned} \tag{3.19}$$

### 3.2.8 Geometric image transforms

A geometric image transform is a transform that changes the location of data in an image without changing the data itself. The simplest example of a geometric transformation is a resize. A resize either expands or shrinks the image so that it has more or less pixels, in other words a resampling of the image. As with resampling, a resize involves mapping data with integer coordinates into potentially float coordinates. This is solved through interpolation. Typical interpolation alternatives when remapping pixels are: nearest neighbour, bilinear, and bicubic interpolation.

For our purposes, we wish to move and rotate the audio map onto the camera image. The simplest transform for this is an affine transform. The affine transform can move, rotate and shear an image, based on three origin and destination point pairs. Another more robust transform is a perspective transform, that does the same as the affine transform, but also has the ability to rotate the image in the third dimension so as to change its perceived perspective. Table 3.1 gives a short summary of the two transforms.

| Transform | Req. points | Ability |
|---|---|---|
| Affine | 3 | Move, Rotate, Shear |
| Perspective | 4 | Move, Rotate, Shear, Rotate perspective |

Table 3.1: Overview of two geometric image transforms, their requirements and the operations available with the transform.

As illustrated in figure 3.8, the affine transform can change any square into any parallelogram, while the perspective transform can turn a square into any trapezoid. Thus, the possibilities of the affine transform are a subset of those of the perspective transform.

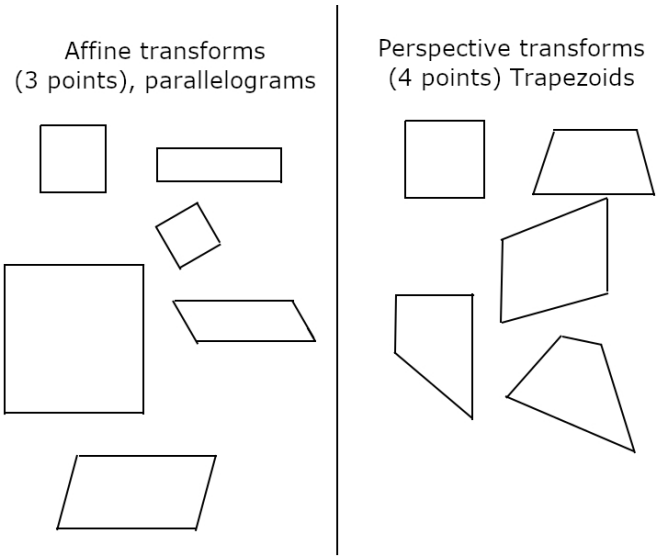



Figure 3.8: Examples of resulting shapes when applying the affine and perspective transforms to a rectangle. All affine transforms are a subset of perspective transforms.

## 3.3 Audio processing

### 3.3.1 Arrays

In the same way that most digital systems store signals over time by intermittently sampling sensors at specific time intervals, arrays sample wavefields at separate points in space. In this way, arrays sample wavefields in both space and time as opposed to how a single acoustic sensor is only able to sample in time. Spatial sampling of wavefields allows for analysis of time-frequency parameters as well as spatial parameters such as the direction a wave travels and the position of the signal source's location in space.

### 3.3.2 Beampattern

It is important to note that while an array might be able to separate signals in space, it does not do so perfectly. How much a signal arriving from a certain angle affects the beamformer output can be graphed as the beampattern. Figure 3.9 shows a typical beampattern for a linear array. The largest peak in the middle of the plot is the mainlobe, it is the direction where the array is focused. All other peaks around the mainlobe are the sidelobes, these are directions where the array is not focused yet signals are still being picked up. Typically, we can describe the features of a certain beamforming configuration using measurements of its beampattern. An array's resolution will depend on the width of the mainlobe, and its ability to attenuate signals from other directions is dependent on the height of its sidelobes.

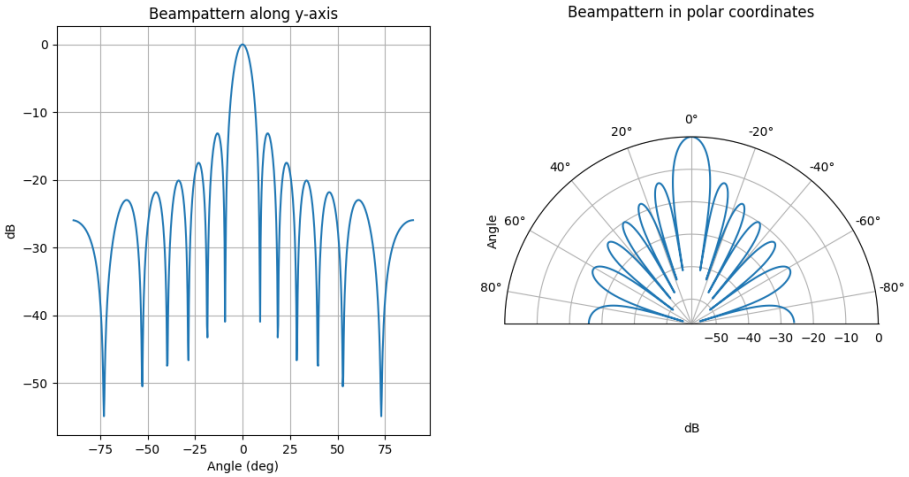



Figure 3.9: Beampattern of a well sampled linear microphone array at 4kHz

### 3.3.3 Spatial sampling criteria

The Nyquist-Shannon sampling theorem states that, in order for a sequence of discrete samples of a continuous bandwidth-limited signal to capture all the information necessary to describe the signal, the sampling frequency needs to be at least twice the highest frequency component present in the continuous signal (Shannon, 1949). That is, to capture a signal with maximum frequency $f_{max}$, the sampling frequency $f_s$ is restricted as shown in equation 3.20.

$$f_s > 2 \cdot f_{max} \tag{3.20}$$

The time difference between samples $\Delta t$ can then be described as in equation 3.21:

$$\Delta t < \frac{1}{2f_{max}} \tag{3.21}$$

The same principle applies to spatial sampling. The maximum distance between regularly spaced sensors depends on the shortest wavelength present in the wavefield. The maximum prescribed sensor spacing or pitch $d$ is then as shown in equation 3.22:

$$d < \frac{\lambda_{min}}{2} \tag{3.22}$$

where $\lambda_{min}$ is given by equation 3.23, where c is the wave propagation speed.

$$\lambda_{min} = \frac{c}{f_{max}} \tag{3.23}$$

In temporal sampling, frequencies above half the sampling frequency will appear as lower frequencies as aliasing. Aliased signals appear as ghost signals/frequencies where there are none. In the case of spatial sampling with an array, the aliasing appears in the form of grating lobes at the edges of the arrays beampattern. The array will then have additional main lobes at the edge of its field of view, similair to side lobes but with the full magnitude of a main lobe. Thus, ghost duplicate signals may appear at certain angles. An example of the beampattern of an array with grating lobes can be seen in figure 3.10.

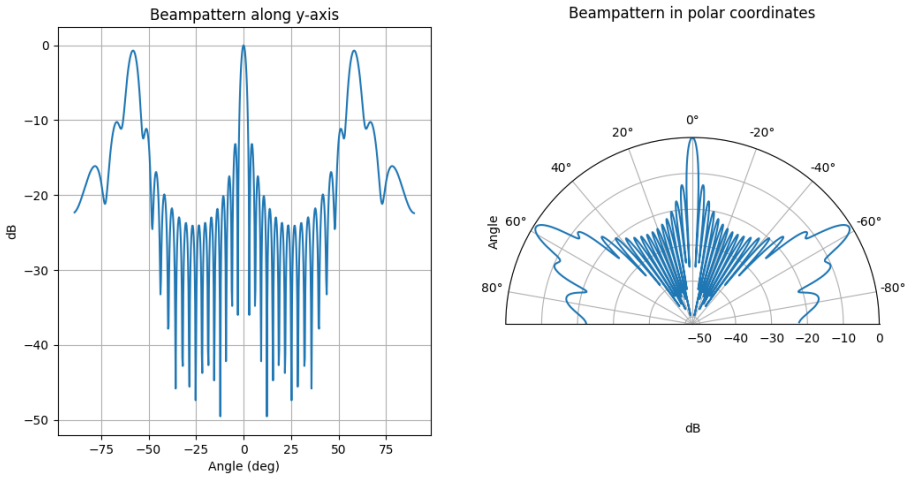



Figure 3.10: Beampattern of the same array as 3.9 at 16kHz, showing the effect of grating lobes on the beampattern.

### 3.3.4 Resolution of apertures and arrays

A sensor array can in general have angular and ranging resolution. Distance ranging requires a three-dimensional array. In our case, the array is a planar array, and has only the angular resolution capability. The angular resolution or resolving power of an array is defined as the closest angle between two sources, where they can still be distinguished from each other and still appear as two distinct peaks at the output of the beamformer. The array's geometry, its size, applied weighting and the type of beamformer used, all affects the arrays angular resolving power.

Generally speaking, the minimum angular separation required between two signals to separate them is inversely proportional to the aperture size and proportional to wavelength, equation 3.24 describes a rule of thumb relationship between this angular separation, wavelength and array size (Grythe, 2016).

$$\theta \propto \frac{\lambda}{D} \tag{3.24}$$

A common definition used in optics is the Rayleigh criterion which states that two plane waves can be resolved if the peak of the mainlobe of one wave falls on the first zero of the other. In other words, the separation between to plane waves

needs to be greater than the distance between mainlobe peak and first zero, for a given wavelength. For a circular aperture, this works out to equation 3.25, where $\theta$ is the minimum angle between two plane waves, measured in radians.

$$\theta \approx 1.22\frac{\lambda}{D} \tag{3.25}$$

### 3.3.5 Far-field/near-field

When imaging wavefields, it is relevant to consider whether we are operating in the near-field or far-field, as the waves radiating from a sound source will appear and behave differently based on their distance to the array. A wrong assumption about the distance to the sound source can lead to positioning errors. In the near-field, the signal source appears as a point in space with the waves travelling radially away from the source, in a spherical shape. With a near-field source, it is possible to estimate the source position. When the signal source is far enough from the array so that the wave propagation direction at each sensor is roughly the same, and the incoming wave can be approximated as a planar wave, the signal source is considered to be in the far-field. The boundary can be defined as a maximum phase-error over the array, typically measured in fractions of the wavelength. One such, is the Fraunhofer distance, commonly used in geometrical optics (equation 3.26) which yields a maximum phase-error of $\lambda/16$ (Holm, 2020).

$$d = \frac{2D^2}{\lambda} \tag{3.26}$$

Using equation 3.26 we can compute the boundary $d$ for the array used in this thesis. Using the inner diameter of the array from table 2.1, 41 cm, a frequency of 4 kHz, and a sound speed of 340 m/s we find that the far/near-field boundary is approximately 4 meters. Similarly, using the outer diameter of the array we get a boundary at about 5.4 meter. Since this thesis primarily uses signals around 4 kHz and the primary experiment is at the range 5.1 m, we are at the boundary between far- and near-field. This implies the distance to the audio source is important for proper positioning, and as such this thesis uses beamformers with near-field modelling.

### 3.3.6 Beamforming

Beamforming is analogous to filtering in space. In much the same way that frequency filtering enables distinguishing signals that are separated in frequency, spatial filtering distinguishes signals separated in space. Some of the many beamforming techniques for this are discussed later in this chapter. In general, the basic idea of a beamformer is to steer/focus an array of sensors in such a way that signals incoming from a desired direction add constructively, and signals from other directions are attenuated through destructive interference.

### Delay-and-sum

The DAS (Delay-And-Sum) beamformer is the simplest beamformer to implement. It computes the travel distance between a desired focus and each sensor location, and converts the distance to travel time using the wave propagation speed $c$. The sensor data is then delayed for each sensor based on the travel time from the focus to each respective sensor. Thus a wave coming from the focus arrives, effectively, at the same time for all sensors (Johnson, 1993, p. 113). Equations 3.27 and 3.28 describe the process for computing delay, and summing up the delayed sensor data. Given M microphones, we compute the time delay $\tau_m$ for each microphone $m$ using the focus points distance to the array $r^0$ and the focus points distance to each microphone $r_m^0$. Using the computed delay we delay each microphones output $x_m$ and apply a weighting $w_m$ before summing to the beamformer output $y(t)$. Figure 3.11 shows an illustration of how delays and weighting is applied to each channel before summing.

$$\tau_m = \frac{r^0 - r_m^0}{c} \tag{3.27}$$

$$y(t) = \sum_{m=0}^{M-1} w_m x_m(t - \tau_m) \tag{3.28}$$

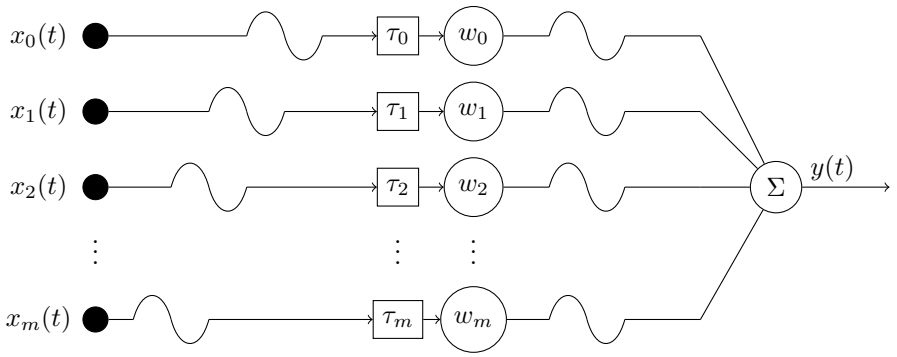



Figure 3.11: Basic illustration of Delay-and-Sum

### Weighting

Apodization or weighting of the array using windowing functions can reduce sidelobe levels, at the cost of increasing mainlobe width, reducing resolution. The windows are applied by weighting the individual sensors outputs using a windowing function before summation. The main purpose of weighting is to compensate for the hard cut-off at the edge of the aperture by tapering it towards almost zero. The cost of this is that the data towards the edge of the array plays a smaller role in the output of the beamformer, effectively reducing the size of the aperture and increasing the mainlobe size. Some common windows can be seen in table 3.2 and figures 3.12, 3.13, 3.14, 3.15, and 3.16.

| Window | Formula | FWHM | Max sidelobe level (dB) |
|---|---|---|---|
| Rectangular | 1 | $^{7.60}/_{D}$ | -13 |
| Hamming | $0.54 - 0.46\cos\frac{2\pi n}{D}$ | $^{11.4}/_{D}$ | -43 |
| Dolph-Chebyshev | eq. 3.29 using $\alpha = 2.5$ | $^{11.6}/_{D}$ | -50 |
| Kaiser | $I_0\left[\pi\alpha\sqrt{1-\left(\frac{n-D/2}{D/2}\right)^2}\right]/I_0(\pi\alpha)$ | $^{12.5}/_{D}$ | -46 |
| Hann | $\frac{1}{2}(1-\cos\frac{2\pi n}{D})$ | $^{12.6}/_{D}$ | -32 |

Table 3.2: Some common window functions over the interval $0 \leq n < D$. (Table from Johnson, 1993, p. 325) Note that $\alpha$ in Kaiser scales mainlobe size at the cost of sidelobe level.

$$
\begin{aligned}
&W(k) \propto \begin{cases} \cos(D\cos^{-1}[\theta(k)]) & \text{if } |\theta(k) \leq 1| \\ \cosh(D\cosh^{-1}[\theta(k)]) & \text{if } |\theta(k) > 1| \end{cases} \\
&\text{where } \theta(k) = \beta\cos\pi k/D \\
&\text{and } \beta = \cosh[(\cosh^{-1}10^{\alpha})/D]
\end{aligned}
\tag{3.29}
$$

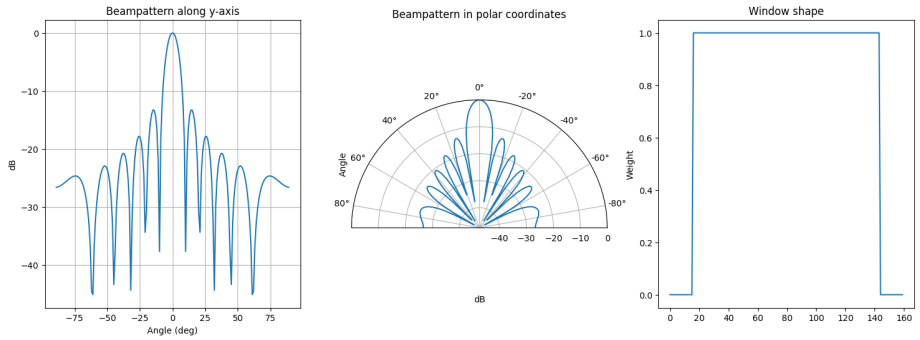



Figure 3.12: Beampattern of a linear array with a rectangular window

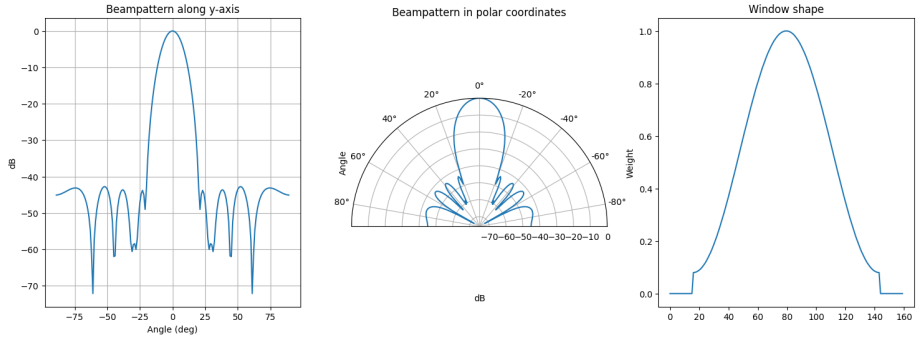



Figure 3.13: Beampattern of a linear array with a Hamming window

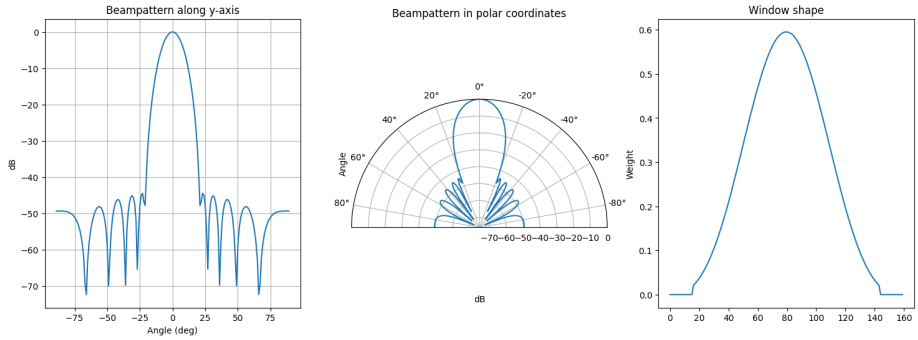



Figure 3.14: Beampattern of a linear array with a Dolph-Chebyshev window

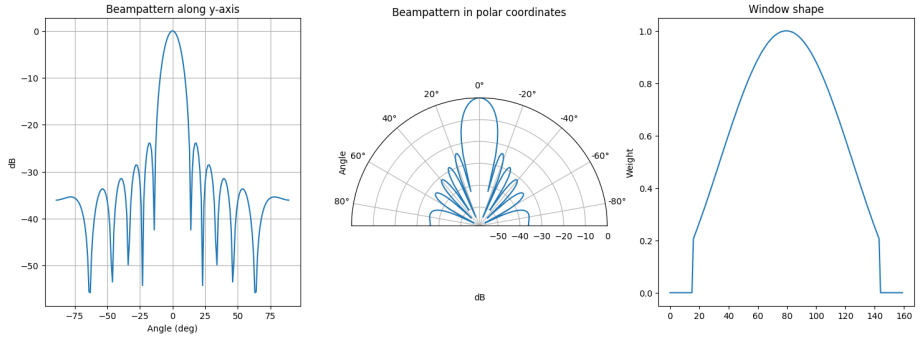



Figure 3.15: Beampattern of a linear array with a Kaiser window

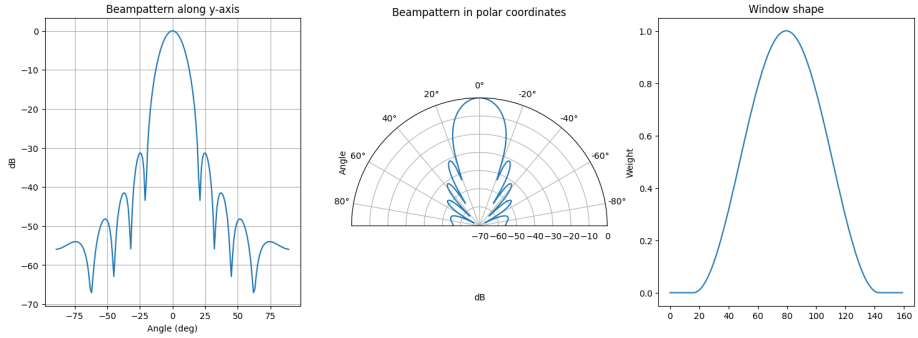



Figure 3.16: Beampattern of a linear array with a Hann window

**Minimum-Variance Distortionless-response adaptive beamformer**

The Minimum-Variance Distortionless-response beamformer also known as the Capon beamformer is an adaptive beamformer. This means that the response at the output is dependent on its input. MVDR seeks to optimize on two criteria minimize undesired signals incoming off-axis from steered direction (minimum variance), while keeping signal from the steered direction unaltered (distortionless response).

Consider a sensor array of n microphones. Let $\boldsymbol{e}(\theta)$ be the arrays response to a plane wave arriving from angle $\theta$. We assume that a monochromatic signal $s(t)$ is arriving from this angle with additive white noise $v(t)$ being present from other directions. The array output can then be described as in equation 3.30.

$$\boldsymbol{y}(t) = \boldsymbol{e}(\theta)s(t) + v(t) \tag{3.30}$$

We can then modify the arrays output by weighting the signal as in equation 3.31.

$$\boldsymbol{y_w}(t) = \boldsymbol{w}^T\boldsymbol{y}(t) = \boldsymbol{w}^T\boldsymbol{e}(\theta)s(t) + \boldsymbol{w}^T v(t) \tag{3.31}$$

The goal is then to minimize the total output of the beamformer $\boldsymbol{w}^T\boldsymbol{y}(t)$ while keeping the signal unaltered in the output $\boldsymbol{w}^T\boldsymbol{e} = 1$. We can formalize this as the two constraints in equations 3.32 and 3.33.

$$\min_w \varepsilon\left(\left|\boldsymbol{w}^T\boldsymbol{y}\right|^2\right) \tag{3.32}$$

$$\boldsymbol{w}^T\boldsymbol{e} = 1 \tag{3.33}$$

Equation 3.34 shows the solution to these constraints, where $\boldsymbol{R}_y$ is an estimated covariance matrix for the array output $\boldsymbol{y}(t)$.

$$\mathrm{w}_{\mathrm{MVDR}} = \frac{\boldsymbol{R}_y^{-1}\boldsymbol{e}}{\boldsymbol{e}^T\boldsymbol{R}_y^{-1}\boldsymbol{e}} \tag{3.34}$$

Equation 3.35 shows the power output of the MVDR beamformer for a direction specified by $\boldsymbol{e}$.

$$\mathcal{P}^{MVDR}(\boldsymbol{e}) = \frac{1}{\boldsymbol{e}^H\boldsymbol{R}^{-1}\boldsymbol{e}} \tag{3.35}$$

For further reading on MVDR see (Lorenz & Boyd, 2005).

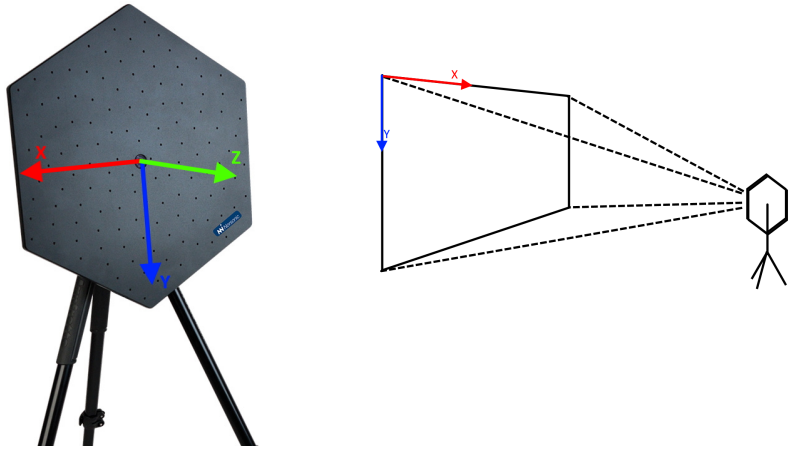
# Chapter 4

# Methods

The following chapter describes a recording setup from which to acquire calibration data, and multiple methods applied to the data to extract positional and calibration information. The setup involves a planar microphone array with a camera and an audio source. The methods describe the computational steps used to estimate positional information from audio and video, and the relative error between them. Finally, different methods to correct for the estimated error are described.

## 4.1 Recording setup

To measure the relative error between the sound and image planes, this thesis assumes the following setup: an acoustic microphone array combined with a digital camera, facing a white wall. The microphones on the array form a single 2d microphone plane, and are pointed broadside (perpendicular to the plane) at an audio source. The digital camera is located in the same plane as the microphones, and is the centre of the array's coordinate system. The camera is also pointed broadside, with the cameras optical axis perpendicular to the array. The array's local coordinate system is defined such that the front of the array points along the z-axis, to preserve a right-hand coordinate system the y-axis points down. This is a common definition used in camera imaging systems. The world and image coordinate systems are illustrated in figure 4.1. The resulting images from both the camera and microphone array have their origin in the upper left corner.




(a) Local array coordinate system. (b) Image coordinate system.

Figure 4.1: Array's local coordinate system, and image coordinate system.

The setup further consists of an audio source. The audio source is placed on a plane/wall parallel to the array's microphone plane and in front of the array. The distance between the array's microphone plane and the wall/plane on which the sound source is located is measured, and is used as the focus distance when beamforming. A background wall is not necessary but is preferred here to ensure that the sound source is always located along the same plane for all recordings made with a given setup. In addition, a wall works as a neutral background for the camera. With the audio source in front of the array during recording, it should be within the field of view of both the array and the camera so that its position may be estimated using both. Several audio source positions are recorded and grouped as a set from which to compute calibration data.

Assuming the setup described above, the challenge of this thesis can then be divided into five steps:

- Perform several recordings with different audio source positions in front of the array.
- Compute the 2d-position of the sound source in the camera image.
- Compute the position of the sound source in space relative to the microphone array
- Estimate the difference/error between video and audio when mapping the audio data onto the image.
- Re-compute the mapping of the audio image using the estimated error, either by changing a parameter such as sensor offset or by adding an image transform on the audio image.

The process is also summarized in the flowchart in Figure 4.2.

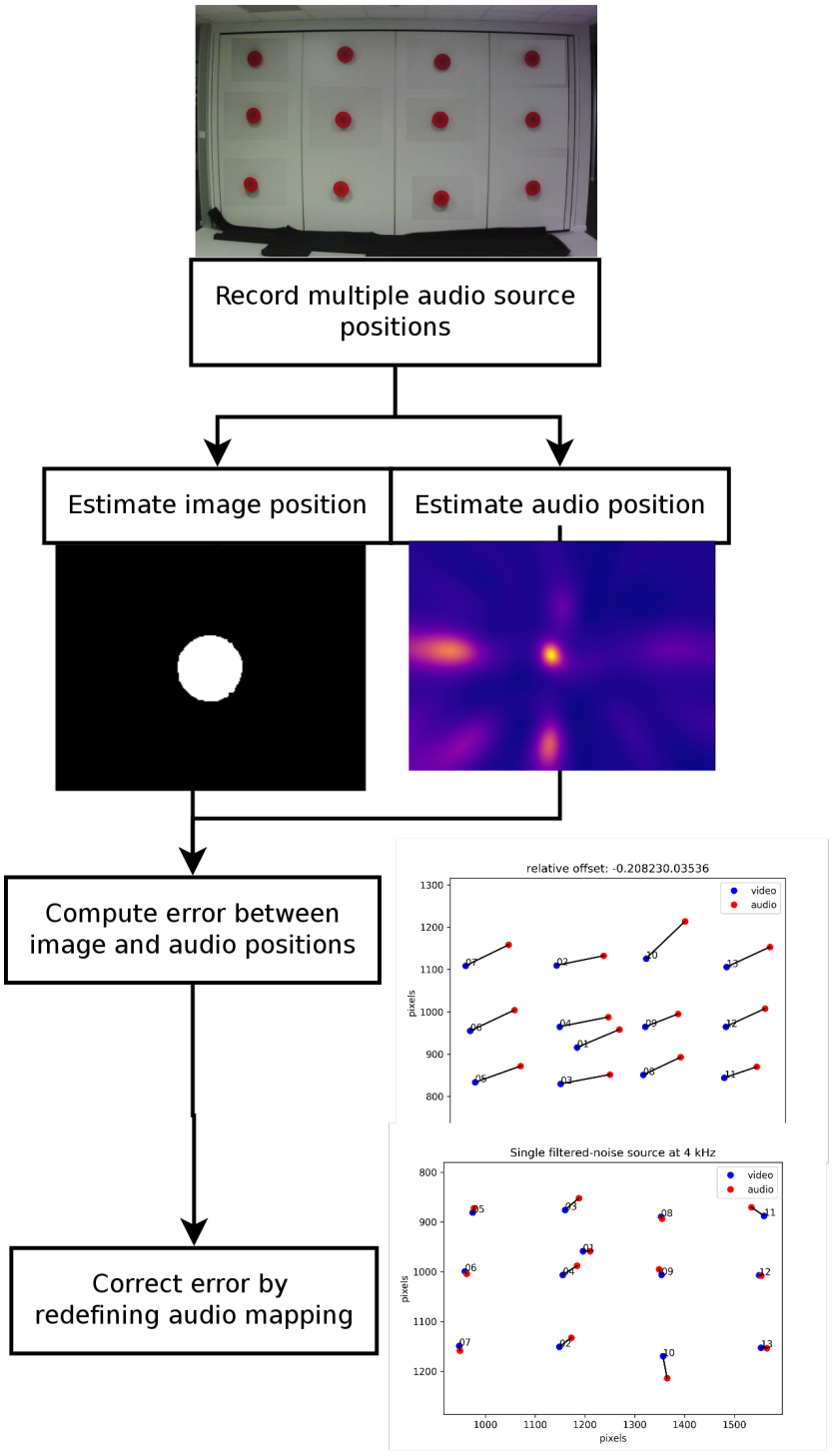



Figure 4.2: Flowchart illustrating the proposed calibration process

## 4.2 Image detection

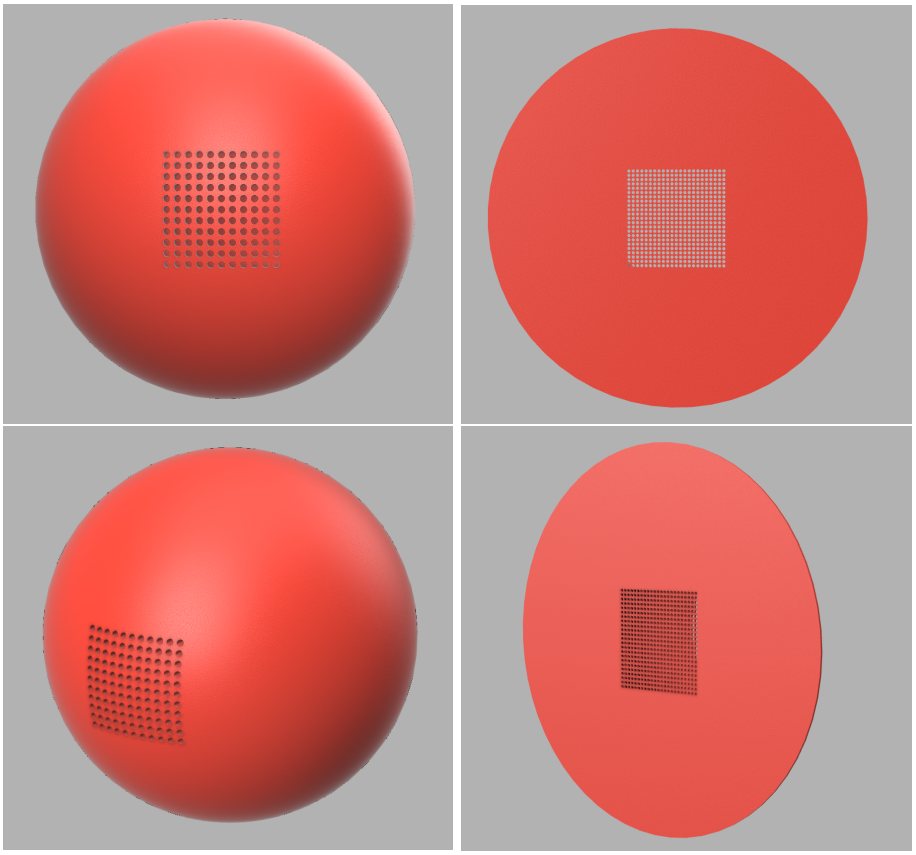
To perform image detection, we need to estimate the sound source location in the image. This thesis will investigate the following options:

### 4.2.1 Estimating position of a coloured object

This is the primary technique used in this thesis for estimating sound source location in the image. By giving the object a known colour, image filtering can be applied to highlight all instances of the colour. To be more specific, the filtering process is applied on a specific hue of colour. Assuming a background with a neutral colour, only the audio source should be highlighted by the colour filter. In addition to giving the audio source a bright, and clear colour, we also make it circular in shape. The circular shape defines a clear centre, avoiding undesired biases when estimating its position. The centre does not change when viewed from a perspective, and the rotation of the sound source does not affect its appearance relative to he camera. The centre of the circular object can be estimated by estimating the centroid, or centre of mass of the object see chapter, 3.2.7.

#### Sound source geometric shape

In order for the sound source to be easily detected, it will need to be brightly coloured as well as large enough for the camera to pick it up. The appearance of the sound source should make it possible to not only locate it but also locate where on the visible part of the sound source the sound is coming from. Purely visually, a coloured sphere seems ideal. Its centre is well defined, and it remains of constant size and shape, regardless of rotation and viewing angle. However, the actual location of the audio source on the sphere would not coincide with the visual centre if viewed from an angle like illustrated in Figure 4.3a. A flat circle however, has similar properties in that its centre is clearly defined and unaffected by rotation, but viewed from an angle, the perspective changes it into an ellipse. The centre of the ellipse is the same point as the circle when viewed from straight on, and thus even though the shape of the circle changes, its apparent centre does not, as shown in Figure 4.3b.



(a) Sphere viewed from front and from angle.

(b) Circle viewed from front and from angle.

Figure 4.3: Arrays local coordinate system, and image coordinate system.

### Implementation

We start by correcting for the wide-lens distortion on the image, the effect of which can be seen in Figure 4.4. The distortion correction is done using already known parameters of the lens, and a sensor offset (See chapter 3.2.2) if it has been calibrated before. If it has not, we assume the sensor offset is zero and the principal point is in the centre of the image.

Figure 4.4: Before (left) and after (right) distortion correction is applied to the raw image from the camera.

The image is then converted into HSV using OpenCV (chapter 3.2.3), and a colour based segmentation is applied (Equation 4.1), creating a binary mask separating the image into object and background. If no object is detected using the current threshold, the threshold for saturation and value is lowered, and the segmentation is reapplied. This is repeated until an object is detected. The mask is smoothed and holes filled through morphological closing, and detected objects only a couple of pixels wide are removed through morphological opening (Chapter 3.2.5). Figure 4.5 shows a flowchart of this process as well as the result of the different steps applied to the image of a loudspeaker covered with a bright red lid.

$$y_{ij} = \begin{cases} 1 & \text{if } h - 20° < x_{ij}^{Hue} < h + 20° \\ & \text{and } x_{ij}^{Saturation} > 150 \\ & \text{and } x_{ij}^{Value} > 100 \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

The resulting binary mask may contain several objects depending on the presence of other coloured objects in the scene. Assuming a relatively neutral background, the most centred target is then selected. The source position is then defined as the centroid of the object, and found using equation 3.19 and raw image moments (equation 3.17).

## 4.3 Audio detection

For Audio detection the recorded audio is used to estimate the sound source location;

### 4.3.1 Audio image

Assuming the sound source is on a plane/wall at a known distance from the array, and that the relative volume of the sound source is higher than the surrounding background noise, we can then focus the array using a beamformer to estimate sound power at a specific point on the plane. Re-using the audio

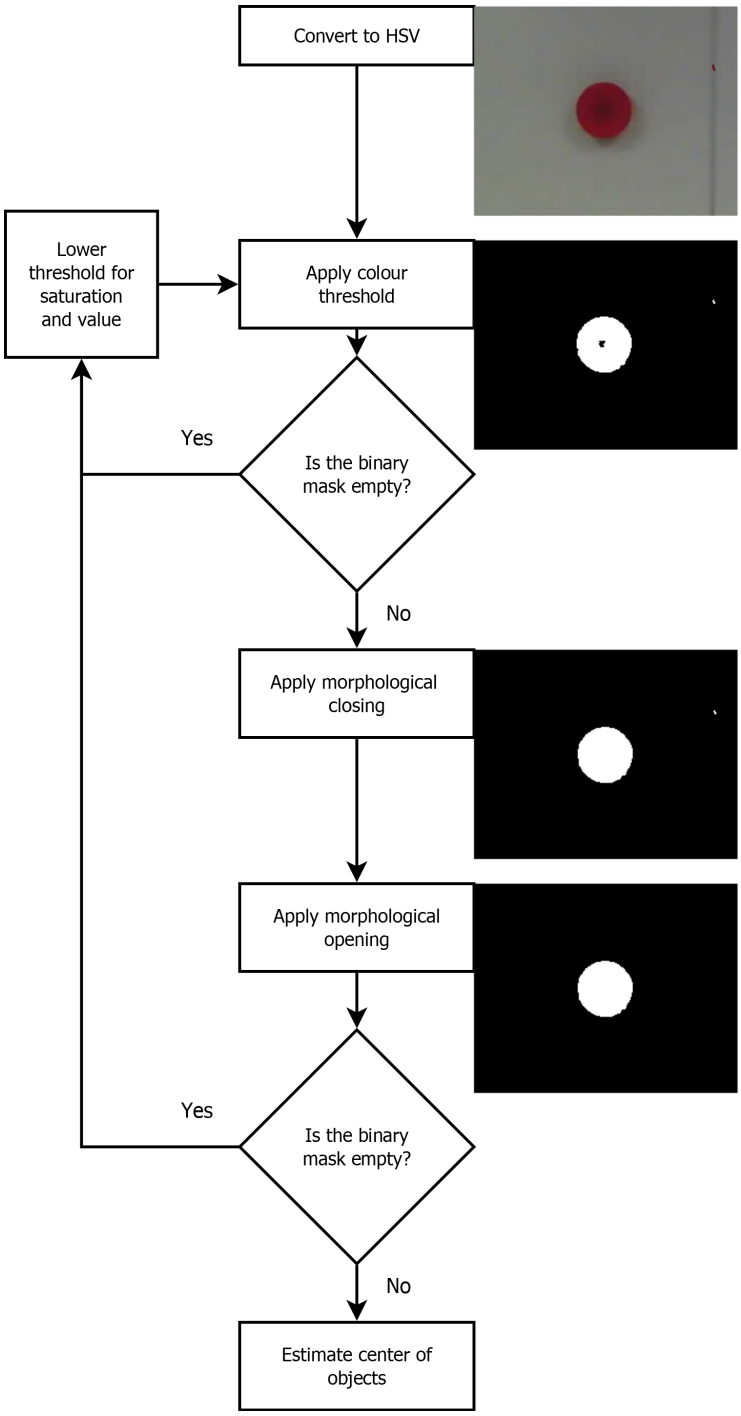



Figure 4.5: Flowchart illustrating image object detection through colour based segmentation. A red dot has been added in the upper right corner of the example to show the effect of the morphological opening step.

data to sample the plane multiple times along a grid we can generate an image containing an estimation of acoustic power along the plane. The grid is defined to have the same aspect ratio as the camera image, and its horizontal and vertical extent is set to match the angle of the cameras field-of-view. Using this data we can find the peak/highest amplitude on the plane, and thus estimate the position of the sound source in the plane.

Here it is possible to generate the audio image with various beamformers, including DAS, FAS and the MVDR beamformer.

### Generating the audio image

To generate the audio image, the 128 audio channels are focused on a point in the plane using the selected beamformer. The summed audio-channel is then used to generate an amplitude estimate for each point on the plane. The plane we focus on is defined by the cameras field of view as well as the distance to the wall where we are expecting the audio source. Assuming a world coordinate system with its origin at the centre of the array we can compute the corners of the plane using equation 4.2, where $d$ is the distance to the wall, and $\theta_h, \theta_v$ are the cameras horizontal and vertical field of view.

$$\begin{aligned} h_{\text{offset}} &= d \cdot \tan\left(\frac{\theta_h}{2}\right) \\ v_{\text{offset}} &= d \cdot \tan\left(\frac{\theta_v}{2}\right) \end{aligned} \tag{4.2}$$

We can then define the four corners of the plane in world coordinates as: $(h_{\text{offset}}, v_{\text{offset}}, d)$, $(-h_{\text{offset}}, v_{\text{offset}}, d)$, $(h_{\text{offset}}, -v_{\text{offset}}, d)$, $(-h_{\text{offset}}, -v_{\text{offset}}, d)$. We then sample this plane using our beamformer, and for our purposes we have chosen 324 samples in the width, which gives a corresponding 243 samples in height. The samples are then put into a greyscale image with 324 x 243 pixels resolution, which is exactly one eight of the cameras full resolution of 2592 x 1944. The image generated is then upscaled to match the resolution of the camera, so that it can be overlaid.

### Localising audio sources

To find the local maxima in the audio image, a maximum filter is applied, setting each pixel to the maximum intensity within a local neighbourhood of 100x100 pixels. A large neighbourhood is chosen to avoid classifying the same peak multiple times. If a pixel has the same value in both the filtered and the original image, it is considered a local maximum. To remove detections in flat areas, we apply a minimum filter and remove any detected maxima where the difference between the local maximum and the local minimum is relatively small. Listing 4.1 shows this implemented in Python, and Figure 4.6 shows a flowchart with examples images of each step.

Listing 4.1: Python code for finding local maximum

```
def find_local_maxima(img, filter_size=100, threshold=20):
    data_max = maximum_filter(img, filter_size)
    maxima = (img == data_max).astype(np.uint8)
```

```
    data_min = minimum_filter(img, filter_size)
    diff = ((data_max - data_min) > threshold)
    maxima[diff == 0] = 0

    return np.array(find_centers(maxima))
```

## 4.4 Error estimation

Using the estimated positions from both audio and camera data, the error is estimated by mapping audio data onto the camera image. The error is then the distance between the image coordinates and audio coordinates pair. By recording several audio source positions, we can achieve a more robust estimate of the error. The error can then be either expressed as a vector of the differences between all the image/audio point pairs, or as an averaged single scalar error. Listing 4.2 shows a python implementation of computing mean scalar error and vector error.

Listing 4.2: Python code to compute error between image and audio coordinates in the same coordinate system.

```
def calc_error(video_coords, audio_coords):
    """ video_coords, audio_coords:
            arrays containing N 2d-coordinates. Shape: (N, 2)
        Returns:
            Mean distance between video/audio point pairs.
    """
    diff = np.array(video_coords) - np.array(audio_coords)
    norm = np.linalg.norm(diff, axis=-1)
    mean = np.mean(norm, axis=-1)
    return mean


def calc_error_vec(video_coords, audio_coords):
    """ video_coords, audio_coords:
            arrays containing N 2d-coordinates. Shape: (N, 2)
        Returns:
            1d array with the difference between all coordinates.
    """
    diff = np.array(video_coords) - np.array(audio_coords)
    flat_diff = diff.flatten()
    return flat_diff
```

## 4.5 Error correction

### 4.5.1 Sensor offset

Assuming the most common source of alignment error comes from image sensor position being either poorly calibrated or changed due to events during transportation and handling, the easiest fix is estimate a new offset in the distortion correction of the camera. This could be accomplished by a gradient descent algorithm searching for the sensor offset with the least alignment error between the camera and audio image.

A least squares estimate can be performed by using a vector function $f(o_x, o_y) \rightarrow \mathbb{R}^{2N}$, where $(o_x, o_y)$ defines a value for the centre offset, and $N$ denotes the num-

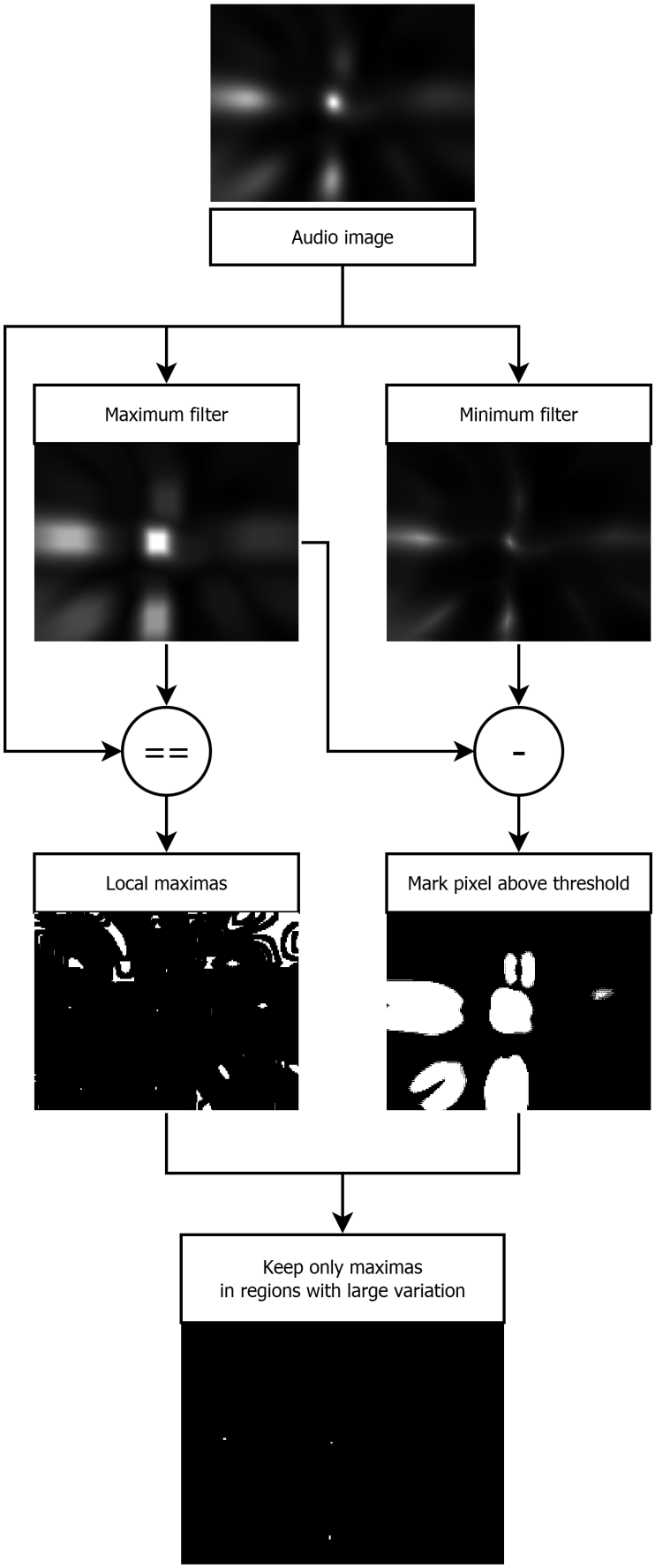



Figure 4.6: Flowchart describing the process of locating local maximas

ber of image/audio coordinate pairs. As such we have an error function giving the error as a vector with all measured differences. The python package SciPy provides a function `scipy.optimize.least_squares`, to minimize nonlinear least-squares problems. This also provides multiple methods for solving the optimization problem. In this thesis the Levenberg-Marquardt algorithm is the one used.

### 4.5.2 Camera rotation

We can correct for alignment errors stemming from camera rotation by applying the rotation matrices described in 3.1.1. We define a function $f(\alpha, \beta, \gamma) \rightarrow \mathbb{R}^{2N}$ that returns the error between $N$ point pairs by rotating the image coordinates about the x, y and z axes by angles $\alpha$, $\beta$ and $\gamma$ respectively. The rotation is performed by transforming the points to world coordinates with an inverted camera matrix, applying the rotation matrices and transforming back to image coordinates using the camera matrix.

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = M_{int} R_z R_y R_x M_{int}^{-1} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{4.3}$$

### 4.5.3 Affine and Perspective transforms

Another alternative would be using simple pre made geometric image transforms to re-map either the sound or image planes to best fit the other. A geometric image transform can be described as a function $f(\mathbb{R}^2) \rightarrow \mathbb{R}^2$ repositioning the pixels of an image based on their locations. The transforms examined in this thesis are the affine, and perspective transforms described in Chapter 3.2.8. The OpenCV image processing library implements several convenient functions for both affine and perspective transforms. A summary of the relevant functions can be found in table 4.1

The function `cv2.getAffineTransform(src, dst)` computes an affine transform matrix perfectly mapping a set of 3 source points to a set of 3 destination points. So for any 3 point pairs, this function computes an exact mapping between them. The function `cv2.getPerpsectiveTransform(src, dst)` performs the same operation creating an exact perspective transform using 4 point pairs instead.

The function `cv2.warpAffine(src, m)` can then be used to warp an image using the resulting affine transform matrix m, and `cv2.transform(src, m)` can equivalently be used to re-map a set of pixel coordinates instead of an entire image. The functions `cv2.warpPerspective(src, m)`, and `cv2.perspectiveTransform(src, m)` serve the equivalent purposes for perspective transforms.

However, because the points estimated in the camera image and audio image do not have perfect accuracy it is desirable to include as many points as possible when computing the image transform. The functions `cv2.estimateAffine2D(src, dst)`, and `cv2.findHomography(src, dst)` provide this functionality. They

compute the transformation matrix based on a least-squares method computing the transform with the least error.

| OpenCV function | Description |
|---|---|
| getAffineTransform | Computes an exact affine transform based on 3 point pairs |
| getPerpsectiveTransform | Computes an exact perspective transform based on 4 point pairs |
| warpAffine | Performs an affine transform on an image |
| transform | Performs an affine transform on set of points |
| warpPerspective | Performs a perspective transform on an image |
| perspectiveTransform | Performs a perspective transform on a set of points |
| estimateAffine2D | Estimates an affine transform using a least squares method and at least 3 point pairs |
| findHomography | Estimates a perspective transform using a least squares method and at least 4 point pairs |

Table 4.1: Description of functions provided by OpenCV for geometric image transforms.

## 4.6 The Experiment

This chapter gives a description of the experimental setup used in this thesis as well as an overview of the equipment employed. All experiments were carried out in the Squarehead Offices in Oslo, Norway. All recordings used in this thesis were recorded outside of office hours in a mostly empty office.

### 4.6.1 Equipment

The experiment involved one Hextile planar microphone array as described in Chapter 2 mounted to the tripod in Figure 4.7b using a standard VESA mount. The array was positioned pointing towards a wall so that the arrays microphone plan was parallel to the wall, the array remained stationary during the entire experiment. A standard measuring tape was used to measure the distance between the wall and the array. Two different loudspeakers were recorded, the Anker SoundCore mini wireless loudspeaker, and the SONY SRS-XB12 wireless loudspeaker, both which can be seen in Figure 4.7a. There were three loudspeakers of each type, the Anker loudspeakers were all black, while the Sony loudspeakers were coloured red, green and blue respectively. Also used were 3D-printed lids in red, green and blue respectively, the design of which are described further down in section 4.6.1. The lids were used to simplify the visual detection by acting as a large brightly coloured target for the image segmentation to latch onto.


(a) SONY SRS-XB12 (left) and Anker SoundCore mini (right) wireless loudspeakers.

(b) Tripod used to mount microphone array.

Figure 4.7: Equipment employed during the experiments.

To play audio through the loudspeakers a laptop was connected using Bluetooth toas playback device. The loudspeakers were light enough to be stuck to the wall with double-sided tape to hold them in position during recording.

#### Visual detection

To detect the audio source visually it needs an easily detectable appearance. Solutions using some coloured fabric were considered, assuming the fabric did not attenuate the signal too much and could be mounted in front of the loudspeaker. This would require the construction of some sort of rig to mount the fabric onto. However since Squarehead had recently acquired a 3D-printer, a fully 3D printed lid would be a relatively simple option, the 3D-model could be

provided when calibration is needed, or a lid could be printed and sent at a very low cost.

Due to the cylindrical shape of both types of loudspeakers seen in Figure 4.7a, the lid was designed with a cylindrical base in which to insert the loudspeaker. The front of the lid is filled with a grid of holes in the centre to let sound waves through, small enough to not be too visible by the camera, yet large enough for the 3D-printer to resolve them. The front face of the model is a large circle to give the model as large a visual extent as possible when viewed from the front. The size of the front facing circle was limited by the 3D-printers printable area, and was printed with a 17 cm diameter. In order to strengthen the contact point between the cylinder and the circular front, small edges were added to increase contact area. The finished model can be seen in Figure 4.8

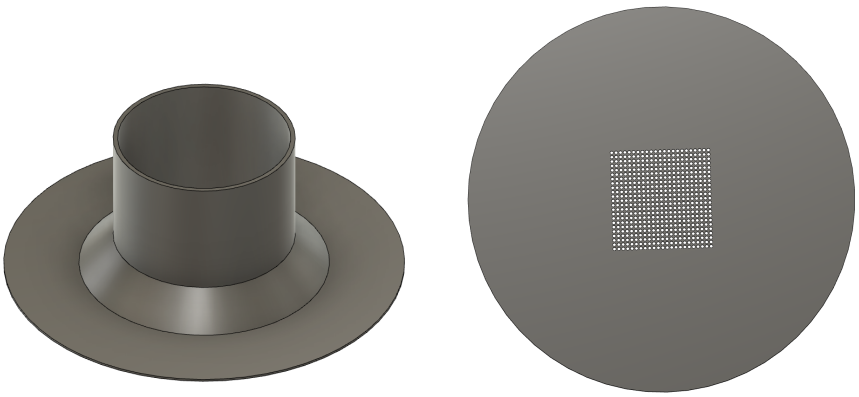


Figure 4.8: 3d-model used to print the lids

The finished set of 3D-printed lids is shown in Figure 4.9. The model's cylinder-radius and -height matches the size of the two loudspeaker types. Rather than colouring the finished printed model, the lids were printed using coloured material to ensure an even and saturated colour.

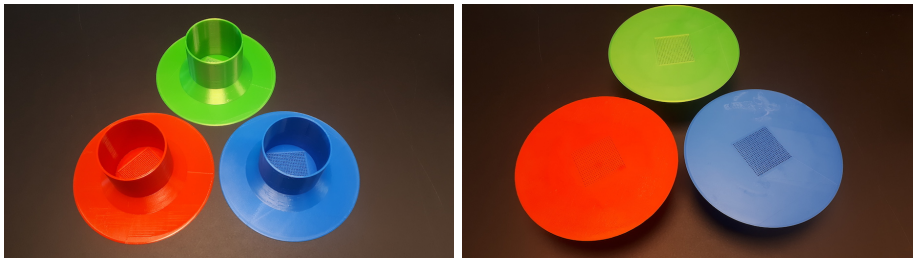


Figure 4.9: 3D-printed lids used to simplify visual detection of audio sources

### 4.6.2 Setup

**Single static audio source**

A single audio source is placed on a plane/wall with a known distance from and perpendicular to the array. The audio source is covered by a coloured 3D-printed circular lid to allow for visual detection. Several recordings are made with the audio source moving to different positions on the wall to map the potential differences between sound and image plane.

The audio source was tested with 4 different variations:

- Anker loudspeaker without 3D-printed lid
- Anker loudspeaker with 3D-printed lid
- Sony loudspeaker without 3D-printed lid
- Sony loudspeaker with 3D-printed lid

The loudspeakers were recorded playing multiple signal types:

- Monochromatic signal at 4 kHz
- Narrow-band white-noise 4 kHz $\pm$ 500 Hz
- Wide-band white-noise 20 Hz - 20 kHz

### 4.6.3 Execution

The array was placed perpendicularly facing a wall, and the distance to the wall was measured at 5.11 m as shown in Figure 4.10. A Bluetooth loudspeaker acting as sound source was stuck to the wall using double-sided tape. A single recording consisted of a loudspeaker/lid combo playing a signal at a wall position. All wall positions for a specific loudspeaker/lid/signal combination forms a recording set. Most recording sets contained 12 positions on the wall with exception to the Anker loudspeaker without lid combination, which contained 4 positions. Figure 4.11 shows an image containing 12 loudspeaker positions from 12 separate recordings edited into one single image.

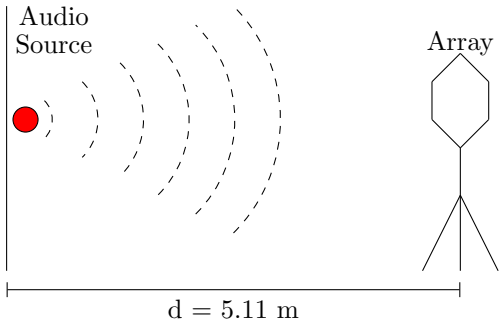



Figure 4.10: Basic experiment setup

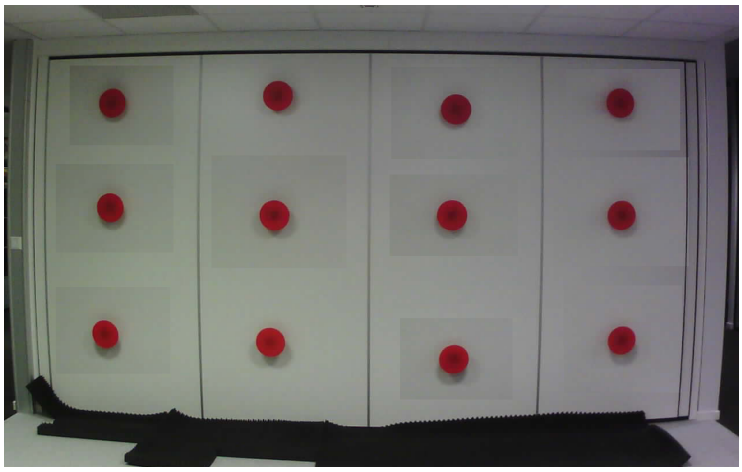


Figure 4.11: Example of a set of audio source positions in a recording set.

# Chapter 5

# Results and Discussion

This chapter describes the results of applying the colour segmentation, DAS, FAS and MVDR methods described in Chapters 4.2.1 and 4.3.1, to the results of an experiment with the setup described in Chapter 4.6. We begin by testing the methods used to detect the image and audio position of the audio source. The errors, and visualizations are made with the factory-calibrated sensor offset, and should as such be considered to have no significant error from mechanical misalignment. A comparison of the different types of audio signals are made, as well as the different kinds of loudspeaker appearances.

Following this, we investigate the performance of different error correction techniques described in chapter 4.5. We do this by introducing an error using either the centre offset, or by applying a rotation to the camera coordinates on some chosen recording sets with relatively good results, and avoiding the recording sets with poor results.

During the experiment, recordings were grouped into sets, where each set is a specific loudspeaker/lid/signal type combo. So a set might be for example all recordings using the Sony loudspeaker with a 3d-printed lid, playing white noise. Each set of recordings with a single stationary loudspeaker contains 12 recordings, each with a different loudspeaker position.

## 5.1 Sound-source positioning

For each recording we locate the audio source in the image using the colour segmentation technique described in section 4.2.1, and locate the audio source with the microphone array by creating an audio image and locating local maximas as described in section 4.3.1.

We can then generate a visualization of a recording set by plotting all point pairs in the set onto the same coordinate system. Figure 5.1 shows a combined plot for a recording set of the Sony loudspeaker with a 3d-printed lid, playing white noise, with audio positions estimated using the DAS beamformer. Worth noting in this plot is that the coordinate system origin is outside the boundary

of the plot, and as such this is a zoomed in version of the real extent of the coordinate system.

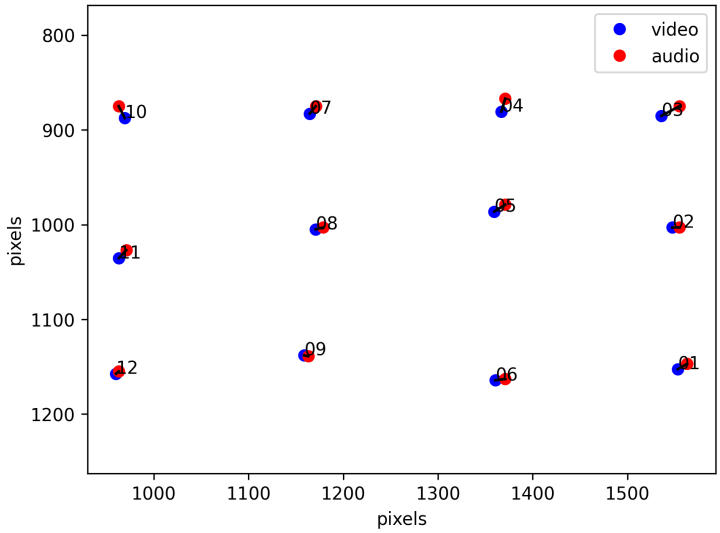
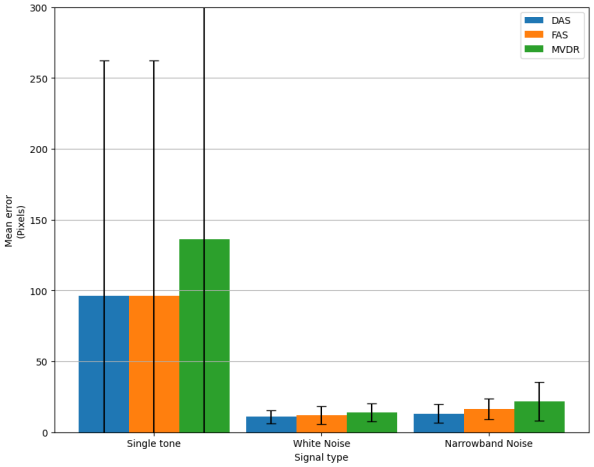


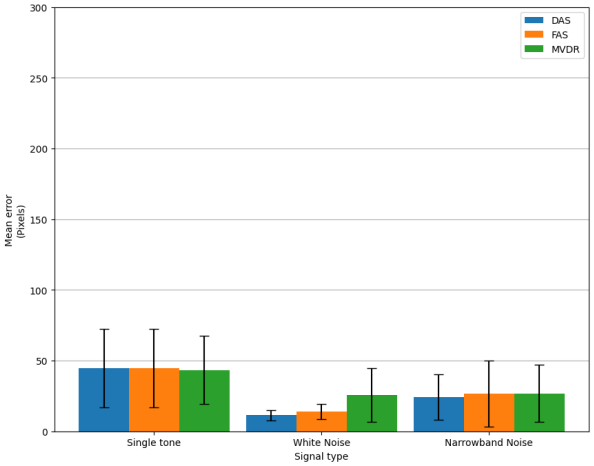

Figure 5.1: Comparison of audio and image positions for the Sony loudspeaker with 3d-printed lid playing white noise. Each point pair connected by a line represents a single recording with a unique sound source location. The estimated audio positions are marked in red, and the estimated image positions are marked in blue. Each point pair is connected by a black line, the length of this line is considered the error for that point pair.

For the recording set in Figure 5.1 the distance between each pair of points is relatively low, considering the resolution of 324 x 243 used to obtain the audio image, which implies one audio pixel is 8 x 8 camera pixels.

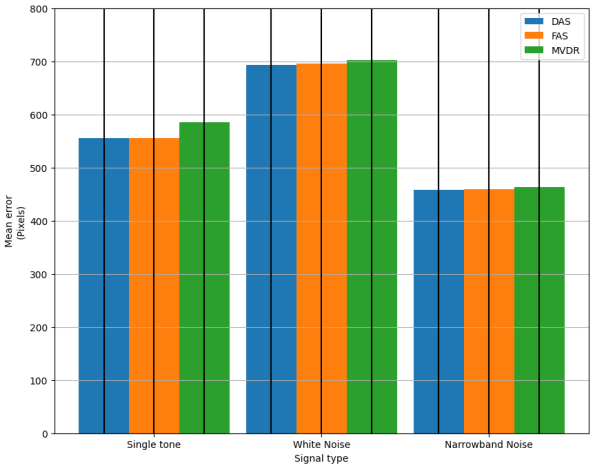
By measuring the distance between the audio position and the image position for each recording in a set, we can compute the mean error in the recording set and compare the different signals and sound source appearances, and beamformers. Figure 5.2 shows a comparison of the mean error for all recording sets, as well as the standard deviation.




(a) Sony loudspeaker with lid




(b) Anker loudspeaker with lid




(c) Sony loudspeaker without lid

Figure 5.2: Comparison of mean error for each set of recordings. Note the change of the y-axis in (c)

### 5.1.1 Wrong target detection in image

Most of the recording sets show a relatively low degree of error, except the recordings in Figure 5.2c, involving the Sony loudspeaker without a 3d-printed lid. We can imagine that this is related to the audio source being harder to detect without the help of the coloured lid placed on it. Figure 5.3 shows the estimated image and audio coordinates for the Sony loudspeaker without a lid playing white noise, using the DAS beamformer.

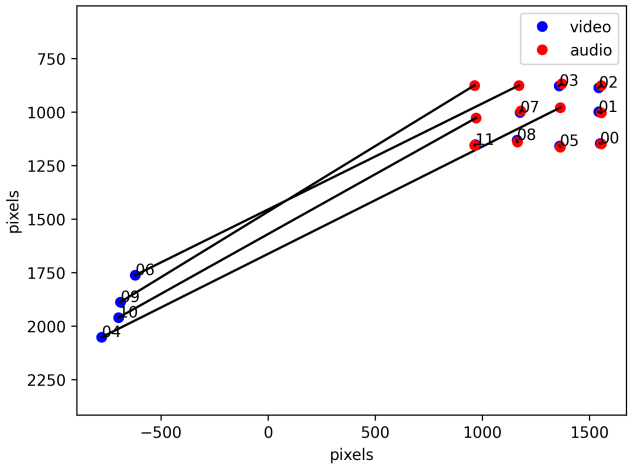



Figure 5.3: Estimated image and audio coordinates for the Sony loudspeaker without a lid playing white noise, using the DAS beamformer.

It is apparent that some sort of object in the bottom left of the image is being detected instead of the intended loudspeaker. Figure 5.4 shows the raw image output from the camera for one of the outlier recordings. Looking at the bottom left, an arm is present in the image. Remember that the target we are looking for here is red, and that the algorithm lowers its colour threshold until some object has been detected. In the case for this recording the redness of the skin from the person on the left is detected before the loudspeaker in the background.

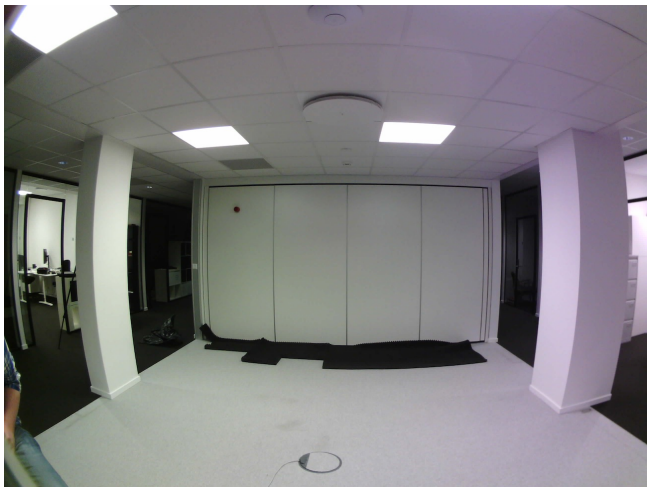


Figure 5.4: Raw image from recording 9 in the recording set for Sony loudspeaker without a lid playing white noise. An arm is present in the lower left, causing wrong detection of the image position of the audio source.

Similar issues are present in the other recording sets without a lid, shown in Figure 5.2c. This might tell us that red is a non ideal colour to target when trying to find an object in the image, as if the intended target is not bright enough humans present in the edges of the image might be wrongly identified as the target. Future tests might include testing with other colours but in the interest of time we apply a crop to the image when trying to detect the image position.

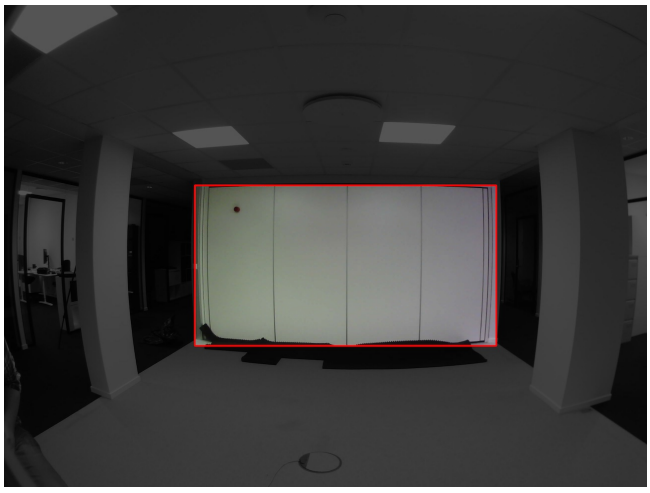


Figure 5.5: Same image as Figure 5.4, but with a crop outline. The outline defines the boundary of where objects can be detected in the image.

By limiting the area on the image where we accept detection we can ignore unintended detection outside the wall we are recording against. Figure5.5 shows

the outline of a rectangle only encompassing the wall on which the loudspeakers are put. By only accepting objects within this rectangle we can to a certain degree guarantee that we are detecting the correct object. Figure 5.6 shows the improvement of this crop on the same recording set as Figure 5.3. The crop has limited the detection of the colour red outside the intended area and the detection now has correctly chosen the loudspeaker as the target.

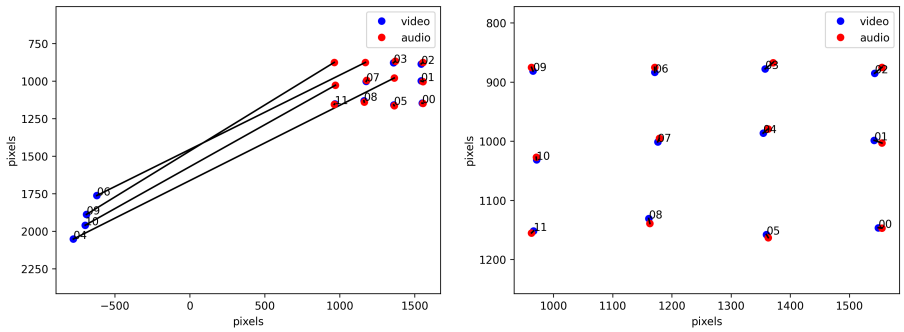



Figure 5.6: Estimated image and audio coordinates for the Sony loudspeaker without a lid playing white noise, before (left) and after (right) a crop to the image detection has been applied. Now most points seem to be correctly on target.

Since all recordings in the experiment were performed without moving the microphone array, we can apply the same crop to all of them to limit the same type of error in all recordings. Figure 5.7 shows the result of applying the crop to all recordings of the Sony loudspeaker without a lid. We can now se that the error on the recordings has been reduced significantly.

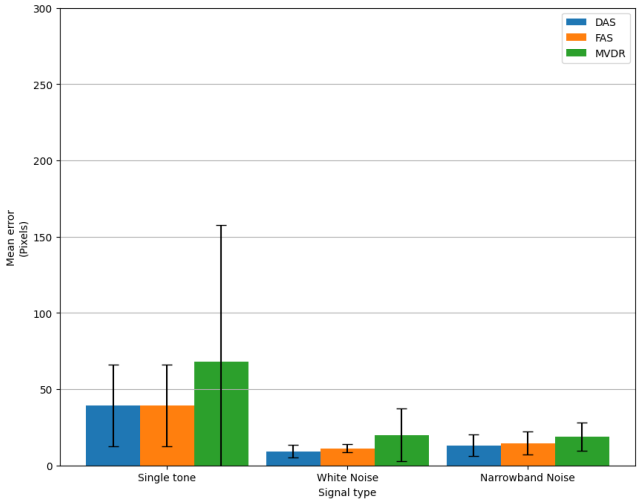



Figure 5.7: Comparison of mean error for the Sony loudspeaker without a lid, after a crop has been applied to the image detection. The scale of the y-axis is here the same as in Figure 5.2 a and b.

### 5.1.2 Single tone - Sony with lid

With the crop implemented, the single tone, Sony with lid recording set as seen in Figure 5.2a has a significantly higher error rate than any of the other recording sets. Looking more closely at the position plot for the recording set in Figure 5.8 we see that the audio coordinates are being placed in strange locations. In this instance the two middle upper source positions seem to be the largest outliers.

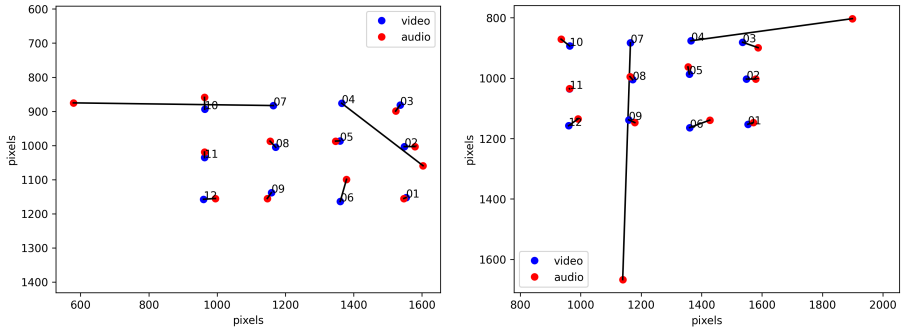



Figure 5.8: Estimated image and audio coordinates for the Sony loudspeaker with a lid playing a single sine wave, using the DAS (left) and MVDR (right) beamformers.

To further investigate we also examine the corresponding audio image for the upper middle left recordings overlaid over the camera image in Figure 5.9, where we can see that there are no spikes in the audio at the location of the audio source. Thus the only detected peak is the reflections in the floor and column.

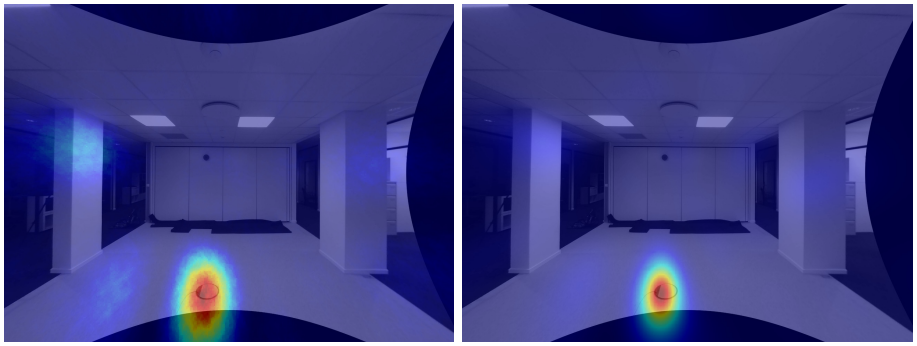


Figure 5.9: Overlaid audio images for recording number 7, using DAS (left) and MVDR (right)

In general a single sine wave as the audio source appears to be the least reliable. This can be explained by the multiple paths the sound is able to take before reaching the array. As such some areas on the array may result in varying degrees of constructive and destructive interference. It is worth mentioning

that these results may have been improved if the phase of the sine wave was accounted for when generating audio images.

### 5.1.3 Beamformer and signal type performance

For this part of the experiment we are looking to examine how the choice of beamformer and signal affects the result. Looking at the comparison of errors in figures 5.2 and 5.7 we can see that the beamformer has had little impact on how closely the estimated coordinate match the camera. Although MVDR offers narrow peaks and great cancelling of off axis interference, these are properties that do not directly benefit in localising the sound source. In almost all cases the DAS results are the closest. Since we are recording in a relatively quiet environment, and are merely picking the peak in the audio image, the beamformers sidelobe level and ability to suppress other noise sources does not in fact play a large role in positioning of the sound source.

With regards to choice of signal we can see that a single monochromatic sine wave gives the poorest results. The single tone is especially prone to error from reflections in the room interfering with detection. Narrow and wide band white noise performs better than the single tone, but the differences between them is small. However of the two, wide band white noise gives the best results. The increased bandwidth of the signal gives a more robust positioning.

From this we can assume that the best combination of signal and beamformer is wide band white noise with the DAS beamformer.

## 5.2 Inducing an image error

Analysing the results from the recordings the microphone array/camera rig used reveals that there are no significant alignment issues. In order to test methods for correcting alignment issues we can artificially induce alignment errors of the kind we expect we might see.

### 5.2.1 Sensor offset error

By changing the sensor offset used in distortion correcting the raw camera image we can simulate an error in the sensors position relative to the lens. Using this we can methodically induce an error between the image and sound planes. The effect of this on a recording set can be seen in figure 5.10.

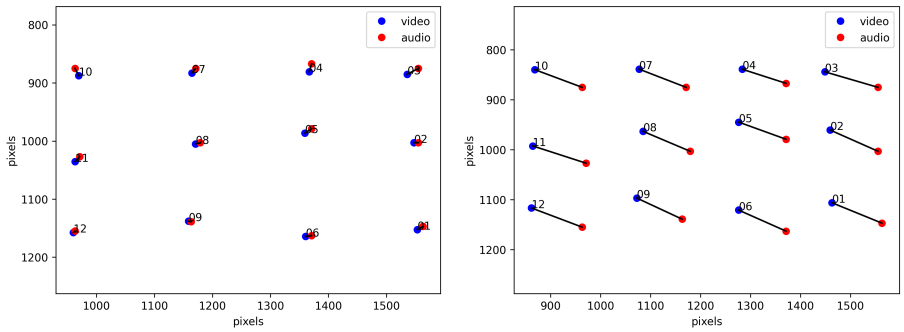
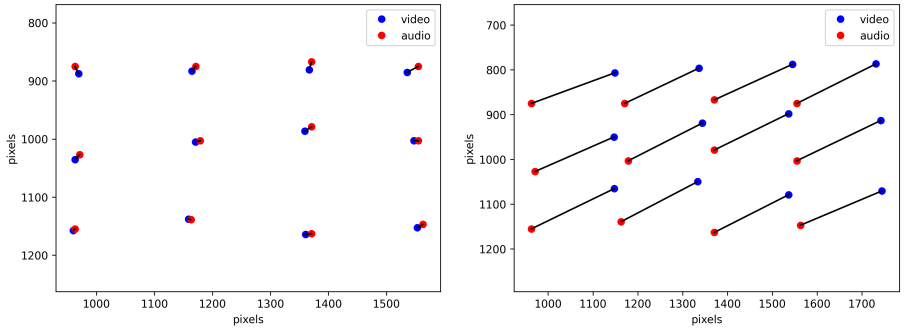



Figure 5.10: Recording set before (left) and after (right) changing the sensor offset.

### 5.2.2 Camera rotation error

By taking pixel coordinates, transforming them to world coordinates and rotating their location along one or several of the cameras axes using the methods described in Chapter 3.1.1 we can simulate alignment errors stemming from a misaligned/rotated camera module, as shown in figure 5.11.




(a) No error induced

(b) Camera rotated

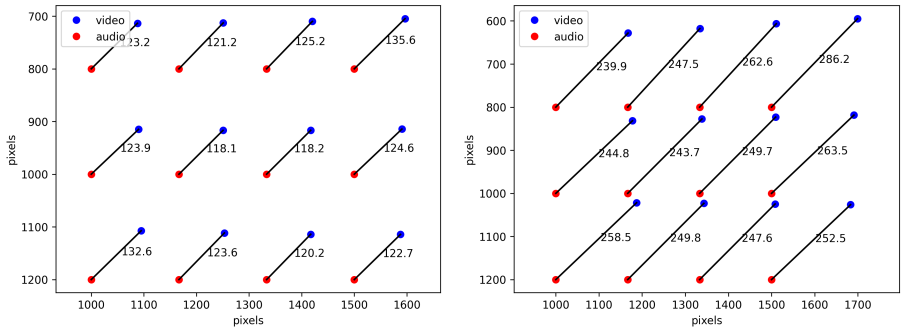
Figure 5.11: Recording set before and after applying a rotation around the cameras x and y axes.

## 5.3 Correcting induced image error

In order to revert the induced error, the same transforms were applied. However in order to estimate the parameters of the sensor offset, and camera rotation an iterative least squares solver was applied. The python package scipy provides a least_squares solver with the Levenberg–Marquardt algorithm as one of the possible solvers. In addition the built in geometric transforms in openCV were tested as well with their own built in iterative methods.

The different methods and correction models were tested using both ideal, and experimental data. Ideal data in this case means that the misaligned coordinates are derived directly from their desired positions using either a modified sensor

offset or a rotation. As such there exists an ideal solution that is the inverse of the applied transform with 0 error. In the case for the experimental data, coordinates from the camera are modified using either a rotation of sensor offset, and the audio coordinates are used as the target location. In the latter event there is no 0 error solution, and as such we can expect the minimum error solution found by the least squares solver to be somewhat different from just the inverse of the applied error.




(a) Modified sensor offset.

(b) Rotation about cameras x and y axes

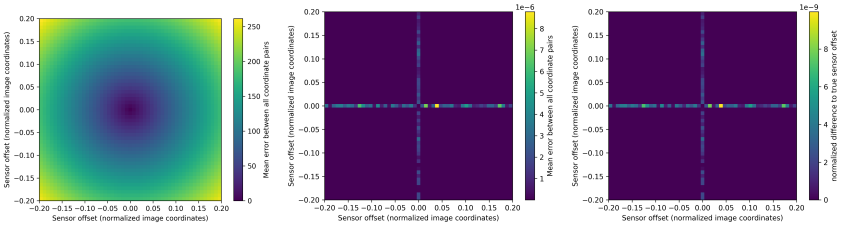
Figure 5.12: Synthetic set of coordinates with transform to image coordinates. The numbers indicate the error in pixels between each coordinate pair

Figure 5.12 shows a set of ideal coordinates that map exactly onto each other but transforms have been applied to the image coordinates. worth noting in the figure is that even though both transforms appear to just be a translation of the coordinates the errors are not the same for all coordinate pairs.

### 5.3.1 Estimating sensor offset

Estimating sensor offset is relatively trivial when done by itself, the offset is merely a translation of the raw image before the distortion correction is applied. As such the most efficient way of correctly estimating the sensor offset is instead of re-applying the distortion correction to the entire image with a different offset we can instead distort our audio coordinates. With the audio coordinates in the same warped coordinate system as the raw camera image we can find the translation that maps the image coordinates best onto the audio coordinates.

Using the least squares method in scipy with the Levenberg–Marquardt algorithm we run estimation attempt on both synthetic ideal data, and using one of the coordinate sets from the experiments. We apply a sensor offset to the image coordinates and look at how closely the least squares solver is able to follow different offsets.




(a) Mean error for an ideal coordinate set with different sensor offsets

(b) Mean error after estimating and applying a new sensor offset

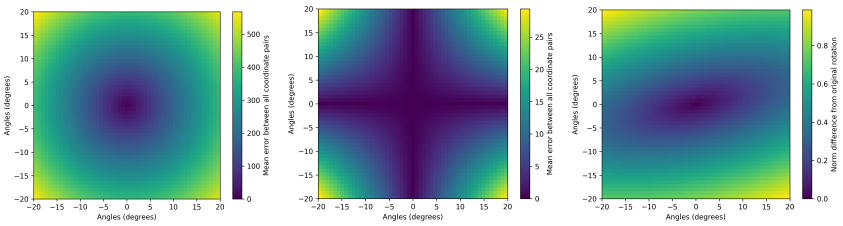
(c) Difference between estimated and true sensor offset

Figure 5.13: Estimating sensor offset for different modified sensor offsets

Figure 5.13 shows the result of applying a sensor offset to an ideal set of coordinate pairs and then estimating the offset using a least squares algorithm. We can see that for all possible offsets we are able to correctly estimate the offset to a large degree of accuracy regardless of the offset.

### 5.3.2 Estimating camera rotation

We apply the same ideas to the camera rotation, by applying a small rotation along the x and y axes we are able to create an alignment error between the point pairs. We use a least squares solver to find the rotation counteracting the induced error above using the same rotation matrix described in equation 3.4.




(a) Mean error for an ideal coordinate set for different camera rotations

(b) Mean error after estimating and applying a camera rotation

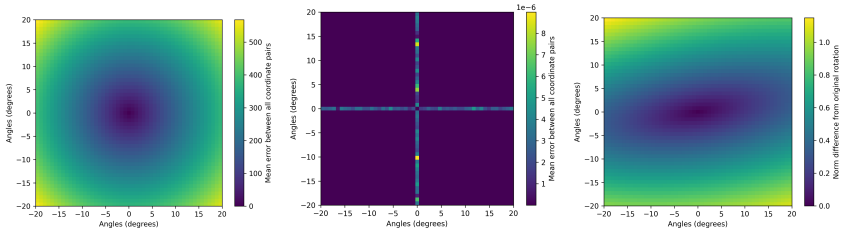
(c) Difference between estimated and true camera rotation

Figure 5.14: Estimating camera rotation using only two axes of rotation.

Figure 5.14 shows that we are not able to completely counteract the induced error by just applying the same two rotations. This can be explained in that the original induced error is rotated around the x and y axis in that order, which also ends up rotating the set of coordinates around the z axis slightly. To counter this we would have to apply the rotations inversely, y then x. But we want to be able to correct for a camera tilted in any direction, so we add another degree of freedom by also rotating around the cameras optical axis, the z axis.

### Adding another axis of rotation

By also rotating around the z axis we should be able to compensate for all possible camera rotations. Figure 5.15 shows that we are indeed able to the error between the coordinate points to nearly zero, even if the estimated rotations themselves dont match up perfectly to the original rotations.




(a) Mean error for an ideal coordinate set for different camera rotations

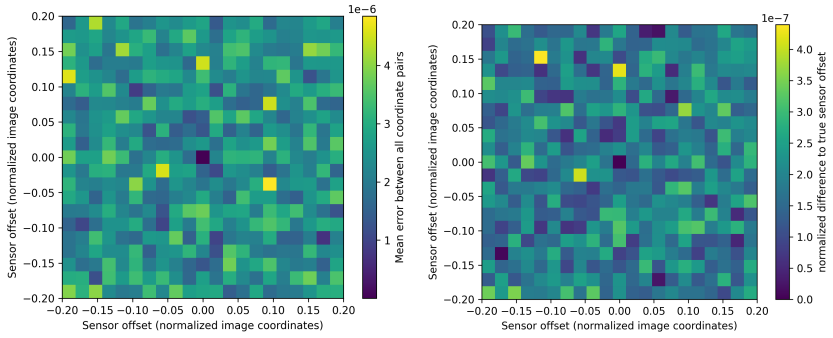
(b) Mean error after estimating and applying a camera rotation allowing rotation around the optical axis

(c) Difference between estimated and true camera rotation

Figure 5.15: Estimating camera rotation using three axes of rotation.

## 5.3.3 Combining sensor offset with camera rotation

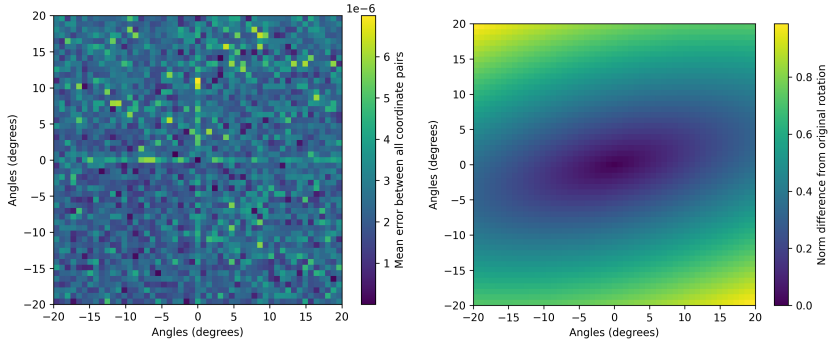
Of course it is certainly possible for both of these types of errors to occur to an acoustic imaging system, either both at once or simply one or the other. Both the cameras sensor alignment an the cameras orientation might be changed by impacts to the system during use and transportation. We define the transform for correcting for both types of errors by combining the two transforms by simply applying first one then the other. We give the least squares solver all 5 parameters to estimate, two being sensor offset and three being the different camera rotations.




(a) Mean error in pixels after estimating and applying a new sensor offset.

(b) Difference between estimated and true sensor offset.

Figure 5.16: Estimating sensor offset for different modified offsets with both sensor offset and rotation as parameters.




(a) Mean error after estimating and applying a camera rotation allowing rotation around the optical axis

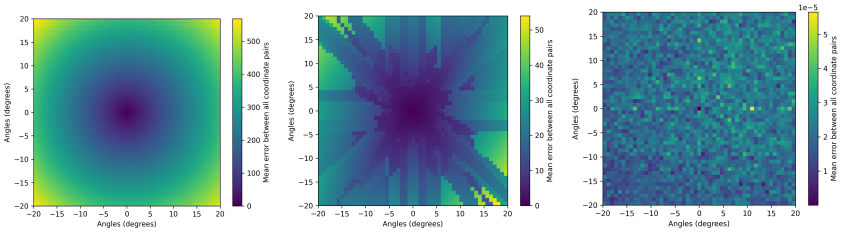
(b) Difference between estimated and true camera rotation

Figure 5.17: Estimating sensor offset for different modified sensor offsets

Figures 5.16 and 5.17 show that with all 5 parameters combined, the least squares solver still ends up correcting using only the appropriate parameters, as opposed to a combination of all parameters. This indicates that there is enough of a difference between the translation from a rotation and the translation from an offset sensor.

### 5.3.4 Image transforms

A general image transform is able to move, scale, and rotate the audio image to best fit the image data. The OpenCV library has multiple options in this regard, the functions cv2.getAffineTransform() and cv2.getPrespectiveTranform() both generate a transformation matrix exactly mapping one set of points to another. They both however have the disadvantage that they only accept an exact amount of coordinates. 3 in the case of the affine transform and 4 in the case of the perspective transform. Since we might have access to more than 3/4 points, it would be preferred to use as much information as possible when estimating an image transform. The OpenCV library also provides the cv2.estimateAffine2D() and cv2.findHomography() functions. The functions take an arbitrary amount of input and output point pairs and estimates the transformation matrix that remaps the points with the least total error.




(a) Mean error for different camera rotations.

(b) Affine transform.

(c) Perspective transform.

Figure 5.18: Estimating camera rotation using three axes of rotation.

Figure 5.18 shows the resulting error after applying the affine and perspective transforms to different rotations. It is clear that a rotated camera can not be corrected by just translating and rotating the image with an affine transform, especially for larger rotations. The perspective transform performs just as good as just estimating rotation with least squares.

### 5.3.5 Inducing error on data from the experiment

The only difference found when inducing error on data from the experiment was that the parameters estimated were off by a small constant error for each recording set. This means that the minimum error for the data is affected by the variation in the data. The results for inducing error on experimental data were in essence identical to the results for ideal data with the addition of the small constant offset, and as such does not show us anything more than what is already explored above.

# Chapter 6

# Conclusion and Outlook

The goal of this thesis was to develop a way to measure and compensate for misalignments between the sound and image plane in an acoustic imaging system. Several alternatives were explored with regards to image and audio detection schemes, as well as different possible error models and a way of correcting them.

### Image positioning

The primary and most robust image detection method was to dress the object up with a coloured circle to detect. The dedicated 3d-printed lids showed good results in this aspect, giving robust and consistent detection in all instances where this was tested. 3D-printing a lid is both low cost and quickly adaptable to almost any sort of loudspeaker.

There were also tests involving coloured loudspeakers, where the loudspeaker itself has a high contrast colour such as the Sony SRS-XB12 which is available in different colours. This configuration gave less reliable results but still a surprisingly good detection rate considering the low complexity of this setup. Difficulties in detecting a loudspeaker that is smaller in size and less bright in colour can be overcome by inserting information into the detection such as limiting where in the image the detection algorithm searches for the sound source, and using a neutral background avoiding colours similar to that of the target.

### Audio positioning

The primary approach used to estimate the audio position was the audio imaging functionality developed by Squarehead for the exact purpose of mapping audio amplitude onto an image. It then follows that this would be the primary metric to measure how well aligned the audio is. Some challenges were encountered using this method however, primarily in the form of false detections from the audio reflecting in the room. This was mostly a problem for monochromatic signals with the problem being less pronounced with broad band white noise signals. In the case where multiple possible sound sources are found the most centred one was chosen, assuming calibration will be performed in the front of the array.

#### Calibration setup

The calibration setup described in the thesis involved placing a sound source emitting a known signal in front of the array. The sound source was coloured such that it would be more easily recognized by the camera. Although it was found that a lid in front the sound source is preferable, resulting in more reliable results, having the sound source simply be a coloured loudspeaker still gave usable results, assuming the background behind the loudspeaker is of a neutral contrasting colour, such as a blank wall. Thus calibration of audio image alignment can be performed only using a loudspeaker on a high contrast background, given enough distance and avoiding reflections from the room.

#### Correcting alignment error

Although no alignment error was found during experiments, we induced alignment error though two methods; Changing the sensor offset in the raw image by moving the centre before compensating for lens distortion, and simulating a rotated camera.

The alignment error was corrected by estimating the parameters of the error such as the displacement of the raw camera image, or the cameras rotation about its axes. The parameters were estimated using a numeric least squares algorithm. With the parameters estimated the transforms were applied and the resulting alignment measured for error. Using idealized coordinate pairs the least squares estimate of the parameters correctly re-aligned the audio and camera images for both errors induced through modified sensor offset and camera rotation.

When experimental data was used to induce and then correct error, it was found that the least squares estimate does not find the exact matching parameters used to induce error. As such we conclude that the accuracy of the calibration is more dependent on the accuracy of the detection of the audio and image targets rather than the source of the error.

### 6.0.1 Future possibilities

There is still much to be investigated with regards to audio image and camera image alignment. Several ideas were explored that have not been covered in this thesis, these include the possible use of multiple sound sources to condense the calibration procedure into a single recording. Another alternative would be to move the sound source around during recording to cover more area.

There is also the need for a technique to handle outliers, RANSAC (Random sample consensus) was looked into as a possible solution, but not tested.

In order to accurately test the performance of correcting for rotation it would be appropriate to apply the techniques discussed to an array with the camera physically rotated.

It would also be greatly beneficial to perform experiments on other microphone arrays, both of same design to look for deviations in manufacturing, but also testing arrays of different shapes and sizes to look into how well the techniques discussed in this thesis apply in general cases.

# References


*Acoustic camera, norsonic* [accessed May 2020]. (n.d.). https://web2.norsonic.com/product_single/acoustic-camera/

Gonzalez, R. C. (2008). *Digital image processing* (3rd ed.). Upper Saddle River, N.J, Pearson/Prentice Hall.

Grythe, J. (2016). Evaluating array resolution [Last accessed 12 may 2020]. https://web2.norsonic.com/wp-content/uploads/2016/10/TN-array_resolution.pdf

Holm, S. (2020). From diffraction to the fourier transform [Last accessed 07 February 2020]. http://folk.uio.no/sverre/lectureNotes/2020-DiffractionSignalProcessing-ver1-2.pdf

Johnson, D. H. (1993). *Array signal processing : Concepts and techniques*. Englewood Cliffs, NJ, PTR Prentice Hall.

Kaehler, A., & Bradski, G. (2016). *Learning opencv 3* (1st ed.). O'Reilly Media, Inc.

Kannala, J., & Brandt, S. S. (2006). A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(8), 1335–1340.

Lorenz, R. G., & Boyd, S. P. (2005). Robust minimum variance beamforming. *IEEE Transactions on Signal Processing*, *53*(5), 1684–1696.

*Opencv: Color conversions* [accessed 21-01-2020]. (n.d.). https://docs.opencv.org/master/de/d25/imgproc_color_conversions.html#color_convert_rgb_hsv

Shannon, C. (1949). Communication in the presence of noise. *Proceedings of the IRE*, *37*(1), 10–21.