Facing The Music

Binaural Sound Localisation using Machine Learning

Presented by:
Kevin Murning

Prepared for:
Jarryd Son
Dept. of Electrical and Electronics Engineering
University of Cape Town

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Electrical and Computer Engineering

November 7, 2019

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed, and has been cited and referenced.
3. This report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof.

Signature:.....
K. M. Murning

Date:.....

Acknowledgments

Firstly, I would like to thank my supervisor Jarryd Son, who provided much needed guidance through this project.

In a personal capacity, I would like to thank the following people:

My mother Helene, who has been an unwavering pillar of support throughout my life, thank you for teaching me how to think.

My father Mark, who has been unrelenting in his belief of my potential. Thank you for teaching me perseverance and kindness.

Faith, whose individuality teaches me lessons in how to be strong.

Misha, my closest friend, who told me to study engineering.

Terms of Reference

ID:	JS19-03
SUPERVISOR:	J. Son
TITLE:	Facing the Music: Binaural Sound Localization using Machine Learning
DESCRIPTION:	Hearing is a sensing modality that is rarely used in robotics beyond speech recognition, however, it has the useful capability of providing information outside of a visual field view. This can be useful in many scenarios, for example, a search and rescue scenario might involve a person crying out for help from behind obstructions or under rubble. Relying solely on visual sensors would not be sufficient to locate them, however it might be able to locate them using sound. This project aims to develop a system that is capable of predicting the direction of a sound source using machine learning algorithms. The system should be controllable so that it can rotate towards the predicted sound source. While rotating it may be possible to create more accurate predictions.
DELIVERABLES:	The student will be expected to complete the following: • Refine the technical requirements of the proposed project and develop acceptance test procedures • Identify appropriate hardware and software solutions • Design and build a device to detect sound and rotate • Develop and implement algorithms to localize the sound source and control the rotation of the device. • Perform test procedures on sub-systems and integrated solution • Design and perform experiments to evaluate the performance of the sound localizing device.
SKILLS/REQUIREMENTS:	Mechatronic design, embedded systems
ELO1: Problem solving: *Identify, formulate, analyse and solve complex* engineering problems creatively and innovatively*	Localization of a sound source is a complex problem as real-world environments are noisy and sound may be reflected of various surfaces. The student would need to find some solution to these problems using a combination of signal processing and machine learning techniques which may not necessarily be taught at undergraduate level. A high level of creativity and innovation will be required to design and build the device, and develop the sound localization algorithms, while meeting the technical requirements of the project and within the available budget.
ELO 4**: Investigations, experiments and analysis: *Demonstrate competence to design and conduct investigations and experiments.*	• Design: Refinement of technical requirements, development of acceptance test procedures, justifiable selection of hardware, software and algorithm solutions • Investigations: Research appropriate sound localization techniques and control schemes for rotating the device. • Experiments: Plan and execute experiments to evaluate the performance of the sound localization algorithms, such as measuring the accuracy.
EXTRA INFORMATION:	
AREA:	Signal Processing, Machine Learning, Mechatronics
Project suitable for ME/ ECE/ EE?	ECE, ME

Abstract

This work intends to document the implementation of a binaural sound source localisation system using machine learning, with ultimate ends of deployment on a rotating robotic platform. The goal in this project was to use machine learning to perform sound source localisation that is robust to both noise and reverberation. Design and implementation of this system was a multi-layered process. A data synthesis methodology was designed and implemented. This method was used to generate large synthetic datasets for use in machine learning. Classical signal processing techniques were used to act as a comparative baseline for the machine learning techniques utilized herein. Three deep learning models were designed, trained and evaluated. These were used in conjunction with a rotation algorithm implemented on the recording apparatus. The use of a rotation algorithm, in which predictions are updated with each movement, allows 360° localisation to take place. Experimentation showed that even though a useful model of binaural sound source localisation was developed, its performance with respect to noise and reverberation rejection did not show a significant improvement upon the existing signal processing method. This being said, the deep learning model was successfully deployed on a hardware system and was able to perform direction of arrival estimation with promising accuracy in a controlled environment. This work presents a starting point for further research in pursuit of a robust, end-to-end, data-driven solution to binaural sound source localisation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Binaural sound source localisation (SSL) is the process of determining the position of a sound source relative to an observer through the use of a pair of auditory sensors. This process is one with which almost all humans are intimately acquainted. We have evolved to use our ears in conjunction with head movements to localise sound sources with remarkable resolution. Given the right conditions, humans are capable of determining the direction of arrival of an incoming sound with up to one degree of resolution [**?**]. For us, such a capability is indispensable. Without it, we would not be able to separate voices in conversation from background noise, nor would we be able to tell the presence of oncoming traffic around a blind corner when crossing the street.

In an engineering context, one could imagine such capabilities having application in multiple fields of robotics and human-computer interaction. One such use case could be in the design of a search and rescue robot that utilises auditory perception to localise disaster victims it cannot see. Another implementation could augment the scene analysis performed by autonomous vehicles, allowing for the perception of other vehicles and pedestrians not within the field of view of the cameras and lidar scanners already onboard.

This comparison of sound localisation to computer vision as a means of perception in autonomous systems begs an interesting question: Why only use two auditory sensors? Modern autonomous vehicles utilise multiple cameras to provide highly accurate semantic representations of the real world, the usefulness of which would surely experience degradation if these systems were limited to only two sensors. Given the success of using multiple

sensors in computer vision, why constrain the task of sound localisation to only two sensors? The argument I will present is one of parsimony.

The extraordinary performance of the human auditory system in sound source localisation is evidence that more than two sensors might not be necessary in order to perform robust auditory scene analysis. Austerity in sensor utilisation has the advantage of reducing both financial and computational costs. However, in the case of binaural SSL, this pursuit presents the challenge of converting a relatively sparse collection of information into a complex semantic representation of an auditory scene. To some extent, using physical models of acoustics in conjunction with careful signal processing has proven successful in this task. However, the pursuit of a generalisable model, robust to noise, reverberation and other perturbations, presents an opportunity to employ a data-driven approach.

Deep learning has revolutionised computer vision tasks in recent years, which urges us to consider whether this technique could prove equally useful in solving the problems of auditory perception. With these considerations in mind, this study poses the following question: can deep learning be used to develop a robust and generalisable model of binaural sound source localisation? Furthermore, considering the possible use case of a search and rescue robot, can this model be deployed on a mechatronic system in order to locate a sound source and turn to face it? The following section will present these objectives in further detail.

1.2 Objectives of this study

1.2.1 Problems to be investigated

In order to develop a model for binaural SSL using deep learning, several subproblems need to be explored. Recent history has shown that the success of deep learning methods is influenced by the quality and size of the dataset used for training. Furthermore, in many fields of deep learning research, large datasets exist that can be used for the unambiguous evaluation of the performance of a model. In the case of spatial audio, both concerning training and evaluation, no such benchmark datasets exist.

The absence of such a dataset presents two problems: First, in order to train a model, a labelled dataset needs to either be recorded or synthesized. Second, an evaluation methodology of the performance of the model needs to be created, such that comparative

benchmarks can be set. Another subproblem relating to deep learning is the design of the model architecture. Many possible architectures exist, and experimentation and theoretical development need to be performed to design an architecture for this use case.

In terms of deployment of the model on a mechatronic system, the most immediate concern is adapting notoriously computationally expensive deep learning algorithms to perform well on a resource constrained embedded system. Furthermore, appropriate hardware needs to be designed and implemented, and an algorithm needs to be developed in order to rotate such hardware with reference to model predictions. Lastly, a classical approach to SSL needs to be implemented in both simulation and hardware, in order to provide a performance baseline to which deep learning models can be compared.

1.2.2 Purpose of the study

As alluded to in the previous section, the purpose of this study is to utilize deep learning to develop a model of binaural sound source localisation that is both robust to noise and reverberation and capable of generalisation to multiple previously unseen environments. The reason this is being undertaken is in an attempt to use a data-driven approach to alleviate the design of SSL systems from a dependence on assumptions about unseen acoustic environments and simplified physical models. Furthermore, the use of deep learning may be able to overcome the traditional constraints of using only two sensors, as feature extraction performed by a deep learning model may expose useful features not previously considered by classical empirical techniques. Finally, as a proof of concept, the model will be deployed on an embedded system which will be able to rotate to face a localised sound. The purpose of this is to demonstrate that this model is useful in the context of real world robotics applications.

1.3 Scope and Limitations

Many dimensions of the SSL problem exist. These will be discussed in detail in Chapter 2. Due to the time constraints of the project, the scope will be limited to stationary, single source, one-dimensional direction of arrival estimation. The direction of arrival will be limited to a 5° resolution, and will apply to a full circle of rotation. These constraints exclude the estimation of the source distance as well as localisation of more than one source. Tracking of the position of the source will not be considered. Direction of arrival will be limited to the horizontal plane and will not address the vertical azimuth.

1.4 Plan of development

First, a review of the relevant literature is presented in Chapter 2. This will start with a general review of sound source localisation, followed by a more specific look at recent work applying machine learning to the problem. Following this, Chapter 3 will present a thorough definition of the problem at hand. From this, requirements analysis will be performed that will inform subsystem identification. A problem approach trade off study is presented at the end of this Chapter, in which different ways of tackling this problem are compared with reference to resource and feasibility constraints. Next, Chapter 4 will present a theoretical development of the techniques decided upon in the problem approach trade off study. This will cover binaural audio, data synthesis, classical SSL techniques and the application of deep learning for SSL. Chapter 5, will present the design process of the subsystems outlined in Chapter 3. The following chapter will detail the implementation and integration of these subsystems. Chapter 7, will present the designs of tests for system verification and evaluation. Chapter 8 will present the results of these tests, which will be discussed and analysed in Chapter 9. Finally recommendations for further research will be presented in Chapter 10.

Chapter 2

Literature Review

2.1 Sound Source Localisation

Sound source localisation (SSL) is the problem of determining the position of sound sources relative to an observation point. Traditionally this problem is broken up into two separate subproblems: direction of arrival (DOA) estimation and distance estimation. DOA estimation corresponds to identifying the angle of arrival of a sound source at a sensor. Distance estimation entails determining how the far the source is from the sensor. Other degrees of freedom are essential to consider, such as whether it is necessary to localise more than one source, and whether localisation is required to span 1 or 2 dimensions. Humans are naturally adept at obtaining the source location of a sound and filtering the relevant information in numerous challenging environments [**?**]. This challenge is commonly referred to as the cocktail party problem - the task of localising and separating sound sources in noisy, reverberant conditions [**?**]. It is desirable to develop this capability in machines as it has many practical applications in numerous sub-fields of robotics and speech recognition, most notably in human-robot interaction. A fundamental challenge in accurate sound source localisation is the generalisation of methods to be robust to distortion caused by the noise and room reverberation present in different environments [**?**]. This challenge and other problems, such as dealing with multiple speakers and detecting short, low-energy verbal cues, have continued to push research forward in this already mature field. The search for a robust solution to the problems outlined above has ushered a move from a pure signal processing approach to one that leverages machine learning to construct models that are less dependent on assumptions about the signal, noise and environment [**?**]. Biological and psychological models of the human auditory system, as developed by the field of psychoacoustics, have

been instrumental in informing these assumptions. An observation that has been key to the development of SSL is the identification of how the human auditory system uses inter-aural difference cues to localise sound. These cues are the differences between the phase and the pressure levels observed by the ears during perception of an incoming auditory signal [**?**]. Many researchers in SSL have used some variation of these cues to form a biomimetic approach to solving the problem. Surprisingly, even some recent deep learning approaches have made use of these ideas to some success [**?**]. Although the human auditory system has been referenced extensively throughout the literature, most studies in SSL have made use of multi-microphone arrays as the listening system [**?**, **?**, **?**]. The performance of these multi-microphone systems depends on the array geometry and the number of microphones used. Accuracy in these systems generally improves with the microphone count [**?**]. The fact that humans can perform robust localisation tasks using only two sensors implies that in principle, such a possibility should be achievable with machine analysis of a binaural signal [**?**]. The following section will present an investigation into binaural signals and their properties.

2.1.1 The Sound Source Localisation Problem

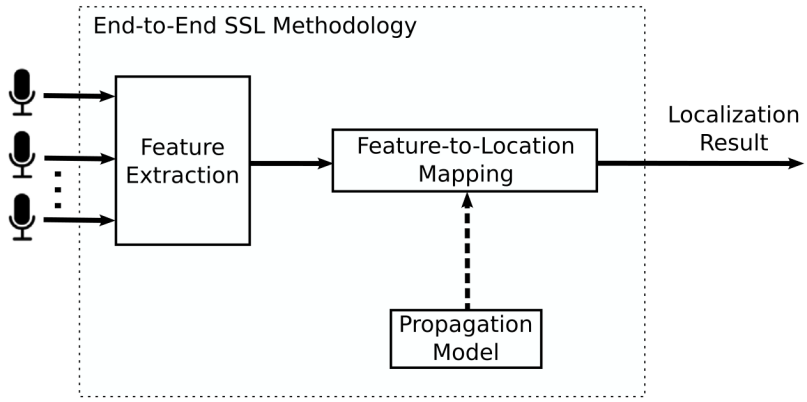

Figure 2.1: A typical end-to-end SSL Methodology [**?**]

Fig. 2.1 shows a classical end-to-end SSL methodology. Unprocessed audio signals are mapped to a DOA azimuth estimation by a chain of processing stages. Initially, feature extraction from the raw audio signal takes place. Following this, a system informed by a propagation model performs a feature-to-location mapping.

Throughout the literature, several different types of features representations exist. The most popular representations usually utilize some combination of the time difference of arrival (TDOA) and the inter-microphone intensity difference (IID). The TDOA is the time difference between two signals when captured by two microphones. Section 2.2.1

presents a detailed analysis of this feature. The IID is the energy difference between two signals at a given time. A frequency-domain equivalent to this is the difference spectrum between two short time Fourier transformed signals, known as the inter-microphone level difference (ILD). Many other features exist, such as the MUSIC spectrum and steered response vectors. However, these features are most used in the case of multiple speaker localisation, and are thus beyond the scope of this work. In binaural localisation studies, binaural cues describe the use of a model of the human pinnae in conjunction with ILD and TDOA features, resulting in corresponding features referred to as the inter-aural level difference (ILD) and inter-aural time difference (ITD).

Microphone configuration and acoustic environment inform the choice of propagation model. The most frequently used propagation model in classical literature is the free-field/far-field model [**?**]. This model assumes that the signal originating at the source reaches the sensor uninterrupted via a direct path. This assumption implies a reflection-free environment, where there are no objects between the microphone and the source. A second assumption is that the inter-microphone distance, and the distance between the microphone array and sound source, is such that the sound wave is planar [**?**]. The recent use of machine learning in SSL aims to use data to learn a propagation model free from these assumptions.

Mapping procedures for simple features such as TDOA are typically carried out by the direct application of a propagation model. However, some features require the optimisation of an SSL solution space. This technique is frequently used in multiple source localisation techniques such as MUSIC and is carried out in the form of a grid-search, entailing the application of the mapping procedure throughout the solution space, in order to produce a solution spectrum which has local maximums at SSL solutions. Machine learning approaches aim to simplify this approach, by implicitly encoding the propagation model into the mapping procedure. Furthermore, if the learned algorithm negates the necessity for handcrafted feature extraction, a fully end-to-end machine learning SSL system is plausible.

2.1.2 Dimensions of the SSL Problem

Multiple dimensions to the SSL problem exist, with different works focusing on different facets of the problem. It is essential to take several sound source characteristics into consideration. These characteristics include the type, number and distance of the sources. In terms of source type, most works in the classical SSL literature do not impose any

strict constraints [**?**]. However, some deep learning localisation studies [**?**] aim to classify the source type (e.g. speech or music) alongside the DOA. Many machine learning methodologies use spatial audio synthesised from existing speech recognition datasets for training, which could lead the model to have a sensitivity towards speech signals.

The number of sources is another principal consideration. The majority of early works aim to localise only one source. However, some recent works are focused on the localisation of multiple sources. The classical signal processing techniques for multiple source localisation such as beamforming and subspace methods are far more intricate and complex than those used for single source localisation. Many deep learning studies have successfully implemented multiple source localisation [**?**, **?**, **?**]. The mobility of sources is a primary concern of some studies. For example, [**?**] emphasises the localisation and subsequent tracking of sound sources through techniques such as Kalman filtering and particle filtering. The distance of the source is of concern to studies in distance and DOA estimation. The majority of works locate sound sources within a 5-meter radius of the microphone array [**?**], corresponding to a small room or office space.

Multiple environmental considerations exist, such as noise and reverberation. Noise can be considered any random data captured by the microphones. Noise robustness is the focus of several recent studies [**?**, **?**, **?**]. Reverberation occurs when the source reflects off elements in the environment, the accumulation of which results in a series of delayed, low energy replicas of the source signal. These signals can propagate towards the sensors from any direction. For these reasons, some SSL systems display sensitivity to reverberation. Most works focused on noise robustness are also focused on mitigating the effect of reverberation.

The type of hardware used is another dimension of the SSL problem. Studies that primarily focus on performance often utilise large multi-microphone arrays. Works focused on binaural localisation are either motivated by austerity or biomimicry. In the name of austerity, some aim to keep the microphone count as low as possible, with some deep learning studies even utilising monaural signals [**?**]. This approach is desirable as it has the potential to reduce both the financial and computational cost of systems. On the other hand, works focused on the development of humanoid robots utilise binaural sensing in order to mimic the human auditory system.

2.2 Classical Techniques

As mentioned in section 2.1, classical approaches to DOA estimation can usually be broken down into three areas of research. The first of these is one-dimensional single source DOA estimation, which entails the estimation of the DOA of a single sound source in the azimuth plane. An extension of this problem is two-dimensional single source DOA estimation, which takes the elevation angle of the DOA into account. Finally, multiple source DOA estimation attempts to estimate the DOA of more than one source.

2.2.1 One Dimensional Single Source DOA Estimation

The calculation of the TDOA of a signal when observed by a pair of microphones is a common approach to DOA estimation in the azimuth plane. This technique is especially relevant in the case of binaural localisation. Calculating a cross-correlation vector between two signals is the most common way of estimating the TDOA [**?**].Cross-correlation can be calculated using the Pearson correlation factor shown in equation 2.1.

$$CCV[\tau] = \frac{\sum_t (x_1[t] - \overline{x_1})(x_2[t-\tau] - \overline{x_2})}{\sqrt{\sum_t (x_1[t] - \overline{x_1})^2}\sqrt{\sum_t (x_2[t-\tau] - \overline{x_2})^2}} \tag{2.1}$$

Here, the two discrete signals subject to correlation are x_1 and x_2. The point at which x_2 is linearly shifted and correlation calculated is τ. The TDOA of the signal (τ_0) is the τ value that maximizes CCV. Using the free-field/far-field propagation model, a feature to location mapping from τ_0 to DOA θ_0 can be calculated with equation 2.2.

$$\theta_0 = \arcsin\left(\frac{V_{sound} \cdot \tau_0}{f_{sample} \cdot d}\right) \tag{2.2}$$

V_{sound} is the speed of sound ($\approx 343 m/s$), f_{sample} is the sample rate in Hz and d is the distance between microphones in metres.

Estimating TDOA from cross-correlation is very sensitive to noise and reverberation [**?**]. A technique to counter this is to use the generalized cross-correlation (GCC). This is a frequency-domain cross-correlator CC_F with a weighting function $\psi[f]$ shown in equation 2.3.

$$GCC_F[f] = \psi[f]X_1[f]X_2[f]^* \tag{2.3}$$

If $\psi[f] = 1$, $\mathcal{F}^{-1}(GCC_F)$ equates to $\mathcal{F}^{-1}(CC_F)$, which is subject to the same noise and reverberation sensitivity as CCV shown in equation 2.1. Normalising the magnitudes in GCC_F by setting the weighting function to Eq. 2.4 is a common technique used to counter this problem. This is known as the phase transform (PHAT), developed by [**?**] as a purely ad hoc technique shown to cause the GCC to be robust to noise, whilst keeping phase information intact.

$$\psi[f] = \frac{1}{|X_1[f]X_2[f]^*|} \tag{2.4}$$

The generalised cross-correlation with phase transform (GCC-PHAT) is shown in Eq. 2.5.

$$GCCPHAT_F[f] = \frac{X_1[f]X_2[f]^*}{|X_1[f]X_2[f]^*|} \tag{2.5}$$

GCC-PHAT is the most commonly used TDOA estimation method for one-dimensional single source localisation because of its robustness and ease of implementation [**?**]. However, in the case of binaural localisation, this approach is limited to the frontal hemifield as there is an inherent ambiguity present in any ITD that occurs in a specific region, which has been dubbed the "cone of confusion" [**?**].

2.2.2 Two Dimensional Single Source DOA Estimation

If the microphone array has 3-dimensional geometry, it is possible to calculate the elevation angle using TDOA as in the 1D localisation case. However, if the array is only 2D, such as in the binaural case, more sophisticated techniques need to be employed. One such technique is the use of a head-related transfer function (HRTF). Often in binaural localisation studies, a head-shaped object is placed between the two microphones in order to emulate the humanoid listening apparatus. This experimental set-up causes filtering of the source signal through reflections of the signal off the body. This filtering is dependent on the azimuth and elevation of the DOA. Thus, the HRTF is a filter that, given a two-dimensional DOA, modifies the sound source so as to emulate the effect of a body

between the two microphones. HRTFs are generated by using a dummy head with two microphones at ear position by recording a set of impulse responses in an anechoic room. To perform DOA estimation, a set of HRTFs are measured with their corresponding DOA. This HRTF-to-DOA association is then used to calculate a database of inverse HRTF filters. Incoming signals are filtered with the inverse HRTF following a grid-search mapping. The DOA associated with the inverse HRTF that maximizes the correlation between the output signals is selected as the estimated DOA [**?**].

2.2.3 Multiple Source DOA Estimation

Approaches to multiple source DOA estimation are generally based on beamforming or spectral estimation methods. Steered-Response Power with Phase Transform (SRP-PHAT) is the most widely utilised algorithm for multiple speaker localisation based on beamforming. In terms of spectral estimation, Multiple Signal Classification (MUSIC) is the most commonly used. Multiple speaker localisation is beyond the scope of this study, however it is important to mention these techniques, as they are frequently referenced in the deep learning studies reviewed in the following section.

2.3 Deep Learning Techniques

2.3.1 Deep Learning and Sound Source Localisation

The techniques for sound source localisation discussed in section 2.2 all rely on strong assumptions and hand-crafted algorithms. These have been shown to be successful only in clean, distortion-free conditions [**?**]. Furthermore, these algorithms have not been successful in generalising to new acoustic environments where the aforementioned assumptions no longer hold. For these reasons, supervised learning methods offer a great advantage as they have been shown to be robust to noise and changes in acoustic environments, without making strong assumptions [**?**]. In recent years, major breakthroughs have been made in numerous signal processing fields, such as speech recognition and computer vision, through the use of deep learning. This is attributable to the fact that these areas entail complex relationships between signals and the underlying processes that generate them, ushering in the need for a data-based approach to aid the development non-linear mappings from inputs to outputs [**?**]. Deep learning approaches are particularly adept at solving these problems due to their data-driven nature. The use of neural

networks to solve problems relating to SSL has existed in the literature since the 1990s [?]. However, limitations in available data and computing power limited the scope of these investigations [?]. Many impressive results in recent SSL research can be attributed to harnessing old ideas efficiently through greater computational capacity. As a result, a number of recent deep learning methods have shown promising SSL performance. These methods differ in terms of network architectures, data, input feature representations, sensor configurations, network outputs and experimental set up. The different studies all have different goals in mind, not all of which are relevant to this research project.

2.3.2 Input Feature Representations

The use of deep learning to localise sound entails the utilisation of a learned mapping of an input representation of audio data, to an output determination of the location of the sound. The choice of input to the network represents a key dispute throughout the recent literature. Several works use various signal processing techniques to preprocess the raw audio data to derive input features from the physical properties of the environment [?, ?, ?, ?, ?]. These techniques are often informed by classic SSL methods such as GCC-PHAT and MUSIC as discussed in section 2.2. On the other hand, some works utilise the raw audio data in some form as network input [?, ?, ?, ?, ?]. These techniques put the responsibility on the network to extract relevant features through training, which could lead to improved performance due to the relaxation of initial assumptions imposed by hand-crafted feature design. In the case of binaural localisation techniques, to the best of the author's knowledge, none of the recent literature makes use of this more generalised approach, with most work [?, ?, ?] seeming to favour biomimetic feature extraction techniques based on the inter-aural cues discussed in section 2.1.1. The feature extraction process of all three of the aforementioned binaural localisation studies is very similar. Each begins by calculating the GCC between the two signals at different frequency bands as separated by a gammatone filter. Following this, in [?], the ILD is calculated and the GCC is averaged across each frequency band to generate two vectors to be used as input. A similar approach is used in [?], however, the GCC at each frequency band and the ILD are combined to create a 38-dimensional feature vector. In both of these approaches, the ITD is not directly calculated, but rather the entire GCC is used as an input feature. This is due to the calculation of the ITD from the GCC to be susceptible to the influence of noise and reverberation [?]. In [?], only the ITD and the ILD for each time frame is used at the network input. This is arguably the input representation that leaves the least responsibility for the learning process to extract useful features.

The approach of handcrafting the inputs to networks is not limited to the binaural localisation case. Multiple studies that utilize microphone arrays also use various data preprocessing techniques to create input vectors. Both [?] and [?] create input features derived from GCC coefficients, whereas [?] makes use of the GCC coefficients directly. In terms of studies where the raw audio data is utilised as input features, there are a few differing approaches. Both [?] and [?] directly use the power information from several frequency-domain frames of a multiple-channel audio stream as input features. Another approach utilised by [?, ?, ?] is to generate spectrograms of each audio stream and concatenate them to create a feature vector. As opposed to these frequency-domain methods, [?] uses the raw multichannel audio samples as network inputs. This is a step towards an end-to-end solution to the acoustic localisation problem.

2.3.3 Network Architecture

Neural network based approaches to SSL range from very simple, shallow networks [?] to networks comprising of complex multi-layer architectures [?]. Both deep neural networks (DNN) and convolutional neural networks (CNN) have been widely utilized throughout the recent literature. It should be mentioned that a number of other network types have been used, ranging from conventional speech processing approaches such as long short term memory networks (LSTM) [?] to more niche network topologies, such as complex-valued neural networks (CVNN) [?] and spiking neural networks (SNN) [?].

Deep Neural Networks

A deep neural network is an artificial neural network with multiple hidden layers. DNNs have the ability to self-organize sensory features from extensive training data, allowing the modelling of complex nonlinear relationships between inputs and outputs. Early studies that employed neural network based machine learning systems for SSL [?, ?] used simple multi-layer perceptrons (MLP) with one hidden layer and a small number of nodes. Multiple different approaches to the use of DNNS in SSL exist. In [?] a different DNN was trained for each frequency channel, each utilising an input layer, 8 hidden layers and an output layer. A more simple approach was taken in [?], training a single DNN with only 4 hidden layers. In their study, they compared the performance of this DNN trained with GCC-PHAT input features to a CNN trained with GCC-PHAT features separated with a mel-scale filter bank. It was found that for single speaker localisation the DNN outperformed the CNN.

Convolutional Neural Networks

CNNs are the most widely used architecture in the recent deep-learning SSL literature. These networks are a variant of standard feed-forward networks in which neuron activations are computed through shared weights over small local areas of the input. These weights are filter coefficients that are learned through training and applied to inputs to generate a feature map at the output of the convolutional layer. Each filter is applied across the whole input space, which leads to fewer trainable parameters compared to fully-connected networks. This is known as weight-sharing. Instead of a standard fully-connected DNN, a CNN architecture usually consists of a number of convolutional layers followed by fully-connected layers at the output. These convolutional layers usually consist of convolution and pooling operations. Pooling is the process of reducing the feature map resolution through the combination of filter activations throughout a specified region [**?**]. CNNs have been found to be successful in a number of audio-related fields, such as acoustic event detection, speech recognition and acoustic modelling [**?**]. The usage of CNNs and the justification thereof in the current SSL literature is detailed below.

An early usage of the architecture in SSL appears in [**?**], where it is utilized in order to enable a more generic input preprocessing chain that can better adapt to different situations. In this study, pooling layers were not used, as the authors found that they were not useful for audio because of the lack of positional pattern shifts occurring in the whitened spectrograms. The study in [**?**] found that the use of convolutional layers could lead to better robustness against noise, as the SNR across the audio spectrum is not constant, allowing the filters to detect phase structures from high SNR regions to compensate for the information lost in low SNR regions. Note that both of the aforementioned studies use spectrograms as network inputs. For studies that utilize other high-dimensional input features such as [**?**], the abilities of CNNS to learn local features with a reduced amount of local parameters lead to a reduction in overfitting and computational expense. Many other studies [**?**, **?**, **?**, **?**] have utilized CNNs with promising results.

Deep Learning Architectures for Learning Acoustic Models

Outside the specific application of SSL, promising designs of network architectures for learning acoustic models have been presented in [**?**]. This work proposes techniques for designing deep convolutional networks for learning acoustic models directly from minimally preprocessed time domain waveforms. The paper presents 5 model architectures

of varying depth and complexity. The notable techniques used in the design of these models are Batch Normalisation and Residual Learning. The models presented in [**?**] are in contrast to other methods that similarly rely on deep networks for joint feature extraction and classification in that they use time-domain input vectors as opposed to their frequency-domain counterparts. The results in [**?**] suggest that using these time-domain techniques match the performance of frequency-domain inputs in audio classification.

2.4 Data

The datasets presented in the literature have largely been synthesized. This entails the convolution an audio signal, usually speech, with an impulse response to impart positional information on the signal. This is typically achieved using a collection of room impulse responses (RIR) containing several impulse response recordings for each microphone at each azimuth interval. These impulse responses can either be synthesized themselves using software such as [**?**] or obtained from existing datasets such as [**?**]. Works that focus on SSL for specific robotics implementations [**?**, **?**] use real data, recorded using the robots in question. This ensures that the acoustic characteristics of the robot, such as microphone configuration, ego-noise and reflections of sound from the robot body, are included in the dataset. Some works focused on binaural localisation use an impulse response library recorded using a dummy head [**?**,**?**]. This is useful for the implementation of humanoid robotic systems with binaural listening, as the acoustic characteristics of the human head are recorded into the impulse response. Studies that synthesize the impulse responses using software [**?**, **?**] utilize the image source method [**?**], a robust model for simulating small room acoustics. This allows great flexibility in choosing sensor configuration and room characteristics when generating an impulse response, meaning that a dataset with multiple different conditions could easily be generated. This is desirable as it could allow for the generation of extensive training data. On the other hand, a weakness in this approach could be a lack of sensitivity to the uncertainty introduced in a real-world experimental setup. For example, a microphone capsule might have a diaphragm pattern that is unaccounted for in the simulation. For this reason, some studies [**?**, **?**] use both a synthetic dataset for initial training and a real one, recorded to reflect their experimental setup, for fine tuning. In the pursuit of robustness against noise and reverberation, many studies include noise in their synthetic datasets. In [**?**], the results show that training the network with noise significantly improves robustness. This is further exemplified in [**?**] where noise was the only signal used for training.

2.5 Evaluation Methodologies

Several types of SSL evaluation methodology are identified in [**?**]. For the purposes of this study two are particularly relevant. The first type of evaluation aims to quantify the performance of the SSL system for a specific facet. This entails a measurement of the precision of the estimated positions given changes in variables such as the position of the source, the SNR or the environmental conditions. The second type of evaluation aims to establish a system of comparing methodologies. This evaluation will develop a common ground upon which to evaluate the performance of method A against method B, given the same inputs. Several metrics exist to quantify these methodologies. *Average Error* is the average difference between the true source azimuth and an estimated one for a set of measurements. *Correct Detection* measures how many times an estimation is correct in a set of measurements. An estimation being correct is determined by it being within a given range of error from the true source. *Precision and Recall* are metrics based upon the correct detection metric. Precision quantifies how well the system predicts true positives. Recall quantifies the occurrence of false negatives. *Front-Back Confusions* is a metric commonly reported in studies of binaural localisation. This is the percentage of times within a set of measurements that the system identifies a source in the incorrect hemifield.

2.6 The State of The Art

Using the terminology outlined in the preceding sections, we can classify the SSL problem in this paper as that of binaural, one-dimensional, single source DOA estimation using deep learning. This section will explore the state of the art in this sub-field, and by making reference to the techniques developed in the greater SSL literature, propose possible improvements to the current methods of tackling this problem. It must be noted that in the field of SSL no standardised dataset exists, such as ImageNet for computer vision. Thus, some interpretation may be required to reliably compare the results reported by different papers.

To the best of the author's knowledge, all of the works that tackle this specific research question utilise some form of inter-aural difference cues as network inputs. In [**?**] a feature vector is constructed using 15 ITDs and ILDs, one pair for each frequency band. Following this, a new vector is constructed by filtering these cues. This is done in an attempt to mitigate the effects of reverberation. A similar approach with slightly less pre-processing

is seen in [?], where a feature vector is constructed using the entire GCC-PHAT vector concatenated with the ILD. One of these vectors is created for each of the frequency bands. A similar approach is used in [?]. Each one of these works utilize gammatone filterbanks to separate the audio into frequency bands. This filtering process is a widely used model of the human auditory system.

In terms of network architecture, the study in [?] utilized a simple fully-connected network with a single hidden layer. This is in stark contrast to [?], in which a DNN with 8 hidden layers was utilized. A different approach is taken in [?], which used a cascade-correlation network with one hidden layer. Both [?] and [?] utilized a dataset generated with a collection of Head Related Impulse Responses (HRIR) recorded using the KEMAR 45BC dummy head. In [?], the image source method was used to synthesize a relatively small dataset of 4000 points. Another technique employed by both [?] and [?] is the utilization of head movements. This is the process of rotating the experimental setup during the localisation process to prevent front-back confusions. This has been successful in achieving greater accuracy in 360 degree localisation.

The most promising results are presented in [?], which reported a 94% localisation accuracy and less than 1% front-back errors. This is testament to the power of DNNs, as this system vastly outperforms the other two systems reviewed in this section. One weakness of this system, as well as the others presented in this section, is the reliance on biomimetic assumptions, evident in the use of inter-aural difference cues and gammatone filtering. One step to improve this could be to strive towards an end-to-end approach with more generic preprocessing as seen in [?, ?, ?, ?, ?]. Another possible improvement could be made with the utilization of CNNs as seen in [?, ?, ?, ?]. This, in combination with the powerful head movement techniques, could yield impressive results.

Chapter 3

Methodology

This section will present and formalise the methodology of this project, both in terms of design and testing.

3.1 Design Methodology

The design methodology followed in this project is the V-diagram shown in figure 3.1. This diagram will act as a roadmap for the following chapters. Sections outlined in the diagram are hyperlinked for easier navigation.

3.2 Formulation of Research Question

Before performing the requirements analysis, it is important to formalise the research question of this project. This has been informed by the project brief and examination of the literature. The project brief specifies the following:

> "This project aims to develop a system that is capable of predicting the direction of a sound source using machine learning algorithms. The system should be controllable so that it can rotate towards the predicted sound source. While rotating it may be possible to create more accurate predictions."

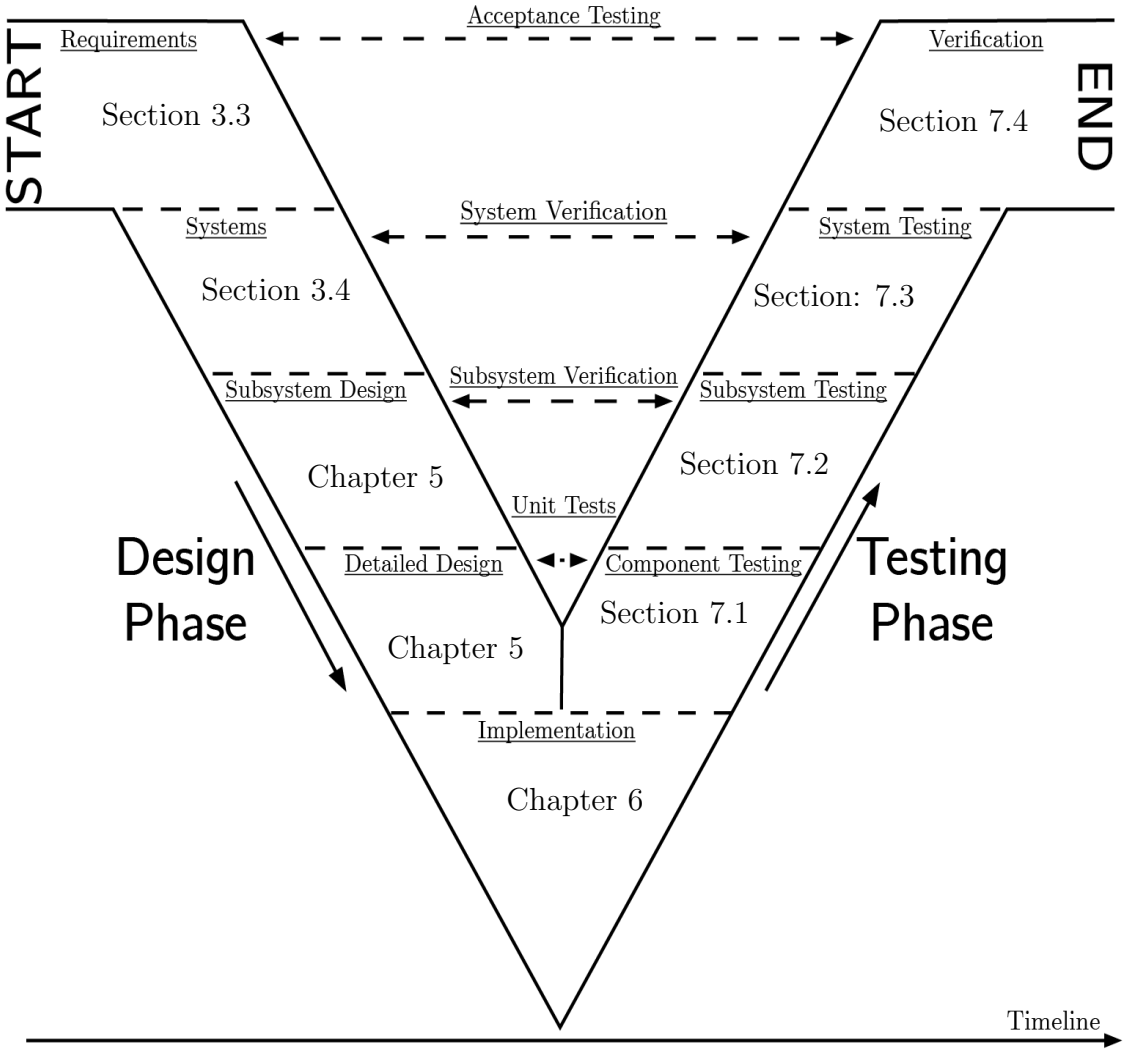

Figure 3.1: V Diagram

As discussed in Chapter 2, many dimensions of the sound source localisation problem exist. Thus it is important to constrain the meaning of sound source localisation appropriately. For this project, sound source localisation will pertain to estimating the direction of arrival of a single source in a single dimension. This localisation will be performed with a pair of microphones, hence the term binaural localisation.

The project brief also specifies that the SSL problem must be solved using machine learning algorithms. Numerous subfields of machine learning exist, and thus for the purpose of this project it is important to refine the specification. A review of the machine learning literature with respect to SSL shows that recent studies almost exclusively utilize deep learning techniques. Thus in order to refine the scope of the project, only deep learning techniques will be used. Furthermore, the project brief specifies that the SSL algorithm must be deployed on a system that is capable of rotating towards the predicted sound source. Thus part of the research question entails determining if rotations and iterative predictions enable more precise SSL.

Two core challenges of SSL identified in the literature review are robustness to noise and reverberation. In this instance noise refers to sound sources that are not the source to be localised. Reverberation refers to reflections of sound waves off objects in the environment. Robustness specifies the ability to reject these sources of sound and locate the source of interest. Much of the classical binaural SSL techniques have proven to be accurate but sensitive to noise and reverberation. One such technique is GCC-PHAT TDOA outlined in Chapter 2. A challenge of the utilisation of machine learning is to develop a system that is at least as adept at SSL as classical signal processing techniques whilst attaining greater ability to reject noise and reverberation.

Given these considerations the research question of this project is presented as follows:

> In comparison to using classical signal processing techniques, can deep learning be used in the task of binaural single source sound localisation in order to develop a system that is robust to noise and reverberation? Furthermore, can the deployment of this system on a rotating robotic platform improve localisation accuracy?

3.3 Requirements Analysis

3.3.1 User Requirements Analysis

The user requirements analysis is presented in table 3.1. This details the high-level requirements of the system. Each requirement has an associated technical requirement and acceptance test procedure. These will be described in the following sections.

	User Requirement
UR1	Predict the DOA of a single sound source in the azimuth plane
UR2	Perform localisation in 360 degrees
UR3	Localisation must be robust to noise
UR4	Localisation must be robust to reverberation
UR5	System must be deployed on a robotic platform in order to rotate to face the sound source once localised.
UR6	The prediction model must be robust in multiple different environments.

Table 3.1: User Requirements Analysis

3.3.2 Technical Requirements Analysis

Technical requirements analysis is a means of quantifying the user requirements into a set of technical specifications required to be met in order successfully conclude the project. The technical requirements are shown in table 3.2. A set of corresponding acceptance test procedures are introduced in section 3.5.2.

	Technical Requirement
TR1	Predict the DOA of a single sound source in the azimuth plane with 5 degrees of resolution. An incorrect prediction within the correct quadrant will be considered the maximum acceptable error.
TR2	Localisation must accurately predict the DOA of sound sources in both the frontal and rear hemifields.
TR3	The localisation algorithm must be able to reject sound sources that do not correspond to the source to be localised
TR4	The localisation algorithm must be able to reject the reflections of the sound source off objects in the environment
TR5	A robotic system must be able to rotate microphones on a stepper motor to face the localised sound source
TR6	The deep learning model must not overfit to the training data and must exhibit a sufficient degree of generalisation

Table 3.2: Technical Requirements Analysis

3.4 Subsystem Identification

Subsystem identification entails identifying layers of subsystems that integrate to form the full system. Given the research question formalised in section 3.2, we can define the full system as follows:

> A robotic system with the capability of rotating to face sound sources using a deep learning based sound localisation algorithm.

The full system described entails multiple subsystems and components. The subsystems are formulated in figure 3.2. Subsystem level 1 separates the hardware and software components of the system. Subsystem level 2 further divides these subsystems into their constituents. The methodology for vertical integration of these subsystems will be outlined in Chapter 6.

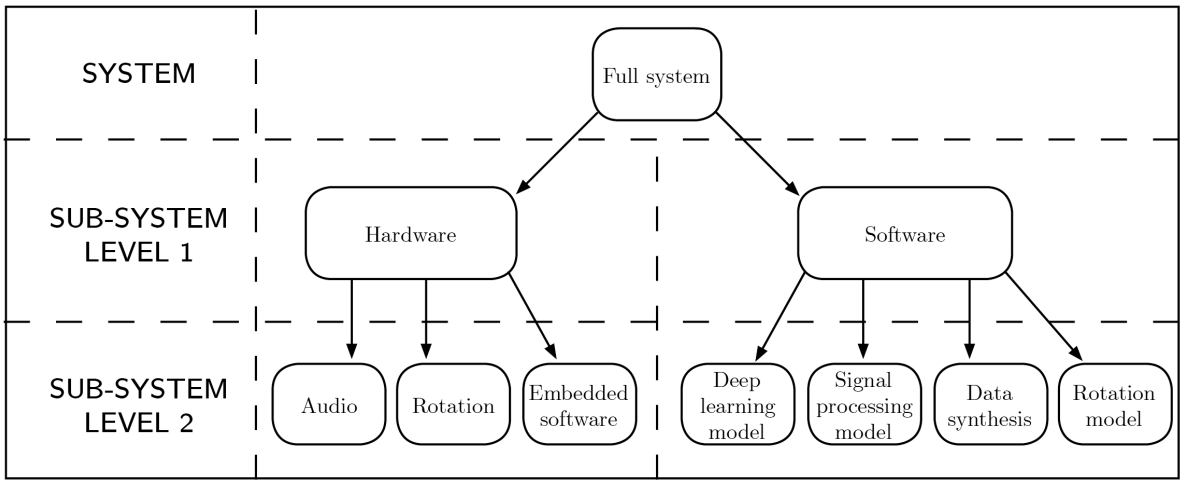

Figure 3.2: Subsystem Identification

3.5 Evaluation Methodology

3.5.1 Evaluation Metrics

Common evaluation metrics in the binaural SSL literature have already been described in section 2.5. Applying these approaches to the research question at hand outlines two primary goals of evaluation:

1. Measure the precision of a prediction model given changes in environmental variables such as room size, SNR level, reverberation level and loudspeaker orientation.
2. Compare the performance a deep learning SSL implementation to a signal processing implementation using *average error* and *correct detection*, where
   (a) *Average error* is the average difference between the true source azimuth and an estimated one for a set of measurements.
   (b) *Correct detection* measures how many times an estimation is within the correct quadrant for a set of measurements.

3.5.2 Acceptance Test Procedures

Acceptance testing is a means of validating that the user and technical requirements have been met. These tests incorporate the use of the evaluation metrics defined in the

previous section. The tests designed in Chapter 5 will implement the acceptance test procedures defined in table 3.3.

	Acceptance Test Procedures
AT1	Run model in constant conditions to form an average error. If the average error keeps the prediction within the correct quadrant for every DOA, the test is passed.
AT2	Run a test to ensure that front-back confusions do not occur. If no front-back confusions occur the test is passed.
AT3	Run a test where multiple conflicting noise sources are present. If the model is able to accurately separate the noise from the signal in each case, the test is passed.
AT4	Test the model in a highly reverberant environment. If the average error does not increase the test is passed.
AT5	Test the correspondance between model prediction and actual rotation performed. If the rotation corresponds to the model prediction then the test is passed.
AT6	Test the model in a completely unseen environment. If the average error does not increase then the test is passed.

Table 3.3: Acceptance Test Procedures

3.6 Problem Approach Trade-off Study

For many of the subsystems identified in figure 3.2, multiple different approaches for implementation have been identified in the literature review in Chapter 2. This section will present a trade-off study for implementation of each subsystem. This will address which techniques have and have not been attempted in the literature, the complexity of their implementation, and the constraints associated with their use.

3.6.1 Data

As previously mentioned, no datasets that reflect the needs of the current research problem have been published. The requirements for a dataset for this project are the following:

- The dataset must represent audio as a stereo binaural image. That is, the spatial information of the position of the source must be present in the data.
- Each data point must be labelled with the corresponding source position.
- The dataset must be sufficiently large as to provide enough training data to be used in deep learning
- The dataset must contain numerous room configurations, levels of noise and levels of reverb in the pursuit of a model robust to these influences

Three different techniques for acquiring data exist in the literature as discussed in section 2.4. The most straightforward method of obtaining a dataset is by recording audio on the hardware used for experimentation. This method has the advantage of creating training data that strongly reflects the environment in which the model will be deployed. However, the accuracy of deep learning models is strongly dependent on the size of dataset used in training. Creating a large dataset by manually recording thousands of permutations of experimental configurations would be an extremely arduous undertaking. Given the time constraints of the project, this approach is not feasible.

Another commonly used technique is synthesizing data using a library of Head Related Transfer Functions (HRTFs). As mentioned in section 2.4, these are collections of impulse response recordings made using an artificial dummy head with a pair of microphones placed in each ear. An example of such a dummy head can be seen in figure 3.3 [**?**] These libraries contain impulse responses of various room configurations with different levels of noise and reverberation. A training dataset is created by convolving another non spatial audio dataset with the dataset of HRTFs, in order to create an arbitrarily large collection of stereo binaural audio files. This technique is appealing as it is relatively easy to implement and is capable of creating large datasets. However, the use of a dummy head to record a dataset introduces multiple complicating factors. The introduction of a head shaped object into the sound wave propagation path introduces filtering not desired in this particular use case. Furthermore, many of these dummys make use of physical filters in the "ears" of the dummy. All of these influences dramatically alter the frequency response of the audio in a way undesirable for this research project.

The final technique for obtaining a dataset is by means of synthesis. Using the Image Source Method (ISM) described in section 2.4, an arbitrary number of sources, microphones and noise sources can be artificially placed in a room of any defined dimension. Precise control over the SNR, reverb time and source distance is possible. Once these variables have been defined an impulse response can be synthesized at each microphone. Following

Figure 3.3: Kemar Dummy Head

this, a stereo audio file can be generated given any arbitrary sound source by means of convolution. This technique has the advantage of flexibility and scalability. A large dataset of many complex configurations can be implemented. Furthermore, a synthetic environment can be created that precisely reflects the real experimental set up to be used. The disadvantage of this method is the complexity of its implementation and the computational expense of generating the data. Given these considerations, the ISM data synthesis technique will be used as it presents the best trade-off of accuracy and flexibility under the project time constraints.

Finally, it is important to specify that all data synthesis to be used in training and evaluation corresponds to a standardized room size. This size is modelled after a $3 \times 3 \times 2.5$ meter room, in which baseline evaluation of the hardware will take place.

3.6.2 Classical Signal Processing Model

The problem of binaural SSL has been tackled with multiple different signal processing techniques. For this trade-off study, MUSIC and GCC-PHAT TDOA will be considered. Both of these techniques have been introduced in section 2.2. GCC-PHAT has the advantage of simplicity and proven accuracy in binaural SSL. MUSIC has been shown to be more robust to noise and reverberation, but is more complex and better suited to the localisation of multiple sources. Considering that the signal processing model used in this project will be for the purpose of providing a comparative baseline, GCC-PHAT is the most appropriate choice due to its ease of implementation.

3.6.3 Deep Learning

Network Architectures

Alongside the size and quality of the dataset used for deep learning, the choice of network architecture is a fundamental factor in determining the success of a model. Multiple differing network architectures are used throughout the literature. These are described in section 2.3.3. The majority of recent studies in SSL have utilized CNNs to some success. Others opt for more complex architectures such a ResNets and LSTMs. This portion of the project will require a degree of experimentation to determine the most appropriate architecture for the task at hand. The advantages and disadvantages of each architecture will be hard to anticipate without experimentation. Thus the prescription of this trade-off analysis is to experiment with CNNs, Fully Connected DNNs, and a ResNet, in order to determine which is best suited to the task. The starting point for design of network architectures will be informed by the work in [**?**]. The implementation of this will be discussed in section 5.2.

Input Feature Representations

Of the techniques reviewed in the SSL machine learning literature, input feature representations are the most frequently contested topic. Multiple different methods have been implemented, ranging from the use of intricate handcrafted preprocessing algorithms, to simply applying normalisation to audio files and feeding them directly into a network. The time constraints of this study prevent the exploration of all the techniques presented in section 2.3.2. Two methods will be used in this study to preprocess and craft input features. These have been chosen as they represent competing philosophies apparent in the literature. Considering that GCC-PHAT will be implemented as the signal processing model described above, and taking inspiration from Ma et al. in [**?**], a set of models will be trained using a GCC-PHAT vector as an input feature. However, unlike Ma et al., the audio will not be split into multiple frequency bands and fed through separate networks. In stark comparison, another set of models will be trained only on the raw unprocessed audio, similar to the techniques used in [**?**, **?**]. The purpose of training models with both of these techniques is to determine if the reliance on manual feature extraction is necessary to create a robust sound localisation model. Achieving promising results using only the raw audio files could be a step towards developing an end-to-end deep learning model of sound localisation.

3.6.4 Hardware

The project specification leaves the description of the hardware relatively open-ended. The primary requirement of the hardware is the ability to rotate to face a sound source once it has been localised. This means that the deep learning model does not need to be run on the embedded system itself. For the purposes of comparison, two possible designs have been identified:

1. Run the deep learning model on a separate computer and control the robot via a serial connection to a microcontroller.
2. Deploy the deep learning model on an embedded linux platform such as the Raspberry Pi and control the actuation of the robot via the GPIO pins onboard.

The primary constraint in the design of the hardware is the computational expense of deploying deep learning models. However, many deep learning frameworks now include tools to help the deployment of large models on constrained hardware such as mobile phones. Thus in pursuit of developing a self contained SSL robot system, the SSL model and actuation system will all be deployed on a Raspberry Pi as described in technique 2 above.

In terms of actuation, the rotation of the microphones requires a motor with precise control over rotation angle. The motor types available are a stepper motor or a servo. A stepper motor will be used as it provides the ability to perform accurate incremental rotations at a resolution sufficient to meet the requirement of a 5 degree rotation resolution. In order to drive the stepper, a stepper motor driver will be required. The choice of stepper motor and driver will be detailed in section 5.5.2.

Chapter 4

Theory

4.1 Binaural Audio

4.1.1 Spatial Audio Representations

This section will present a brief definition of binaural listening and spatial audio. The term *binaural* is a reference to human hearing, describing the process of listening with two ears. In the case of artificial sound source localisation, the term refers to the use of two microphones to capture the data used in localisation. This is in contrast to the use of multi-microphone arrays or a single microphone for SSL. Spatial audio describes how information about the position of a sound source can be represented in the recorded audio waveform. The most common file type used for audio processing is a wav file. Wav files can hold an arbitrary number of audio channels. In the case of binaural audio, two channels are required to create a stereo image. Thus, in order to represent the audio recorded by two microphones and retain information about the spatial characteristic of the recording, a stereo wav file must be recorded from two microphones. The audio from each microphone forms a single channel of the file. This process is illustrated in figure 4.1.

4.1.2 Front-Back Confusions

An important consideration when working with binaural microphone configurations is the inherent ambiguity in determining whether a sound source is emanating from the front

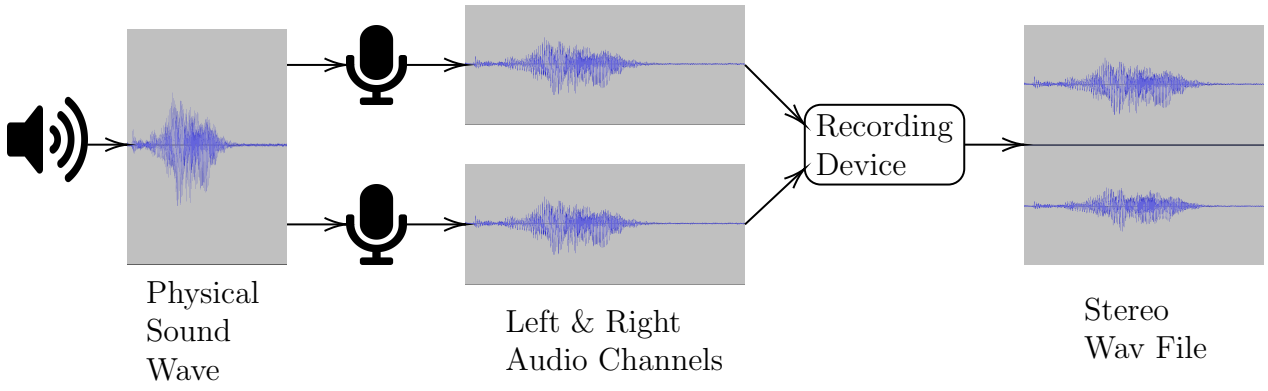

Figure 4.1: Representing Spatial Information in Audio

or the back of the recording apparatus. This problem is illustrated in figure 4.2. In this figure, a binaural microphone configuration is shown in relation to two sources labeled *Front Source* and *Back Source*. These sources are configured such that they are directly reflected across the microphone axis. It is clear that the distance from each source to each microphone is equal, and thus assuming that each source is identical, the power and phase information that can be obtained from a stereo recording will be identical for each source. However, as evident in the diagram, the DOA of each source is different. This phenomenon is present across every positional pair reflected across the microphone axis.

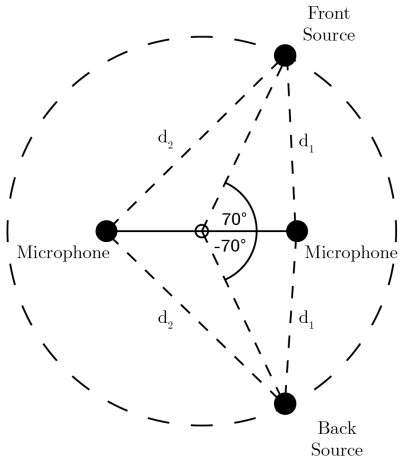

Figure 4.2: Front-Back Confusions

4.2 Data Synthesis

4.2.1 Small Room Acoustic Modelling

A room impulse response (RIR) is a time-domain representation of a point-to-point transfer function that represents the acoustic properties of a room. For example, consider a microphone placed on one side of a concert hall and a loudspeaker on another. If an anechoic recording is played through the speaker, the corresponding audio captured by the microphone will represent the anechoic recording filtered by the transfer function of the room. If one records the impulse response of a room given a particular source and microphone configuration, the convolution of this impulse response with any audio recording will impart the room characteristics onto the recording. If two RIRs are recorded, they can be used to create a stereo image of the source with respect to the configuration of the microphones. Thus, using RIRs in conjunction with a existing dataset of anechoic audio recordings, a dataset of spatial audio can be created. However, manually recording the necessary RIRs in every room, source and microphone configuration is unfeasible. Thus a method of synthesising RIRs using small room acoustic modelling is necessary.

The image source method presented by Allen and Berkley [**?**] is a popular way of synthesizing RIRs due to its computational efficiency and accuracy. Image methods are frequently used to model the acoustic properties of an enclosure. The alternative approach is to generate a model by calculating room modes. Room modes are a collection of resonant frequencies present when a room is excited by an acoustic source. Standing waves occur when acoustic energy is present in a room at one of these modal frequencies. Room modes shape the frequency response of the room, and hence influence the reverberation time. Calculating the modes of a particular room is a complex endeavor. Furthermore, in order to determine the RIR, the calculation of all room modes within the frequency range of interest is required. For an audible frequency range, this becomes a prohibitively computationally expensive task.

In contrast, the image method only requires calculation of the images that influence the RIR. These images are defined as those within the radius $r = c_{sound}RT_{60}$ where c_{sound} is the speed of sound and RT_{60} is the reverberation time. In the case of a point source emitting acoustic energy, images are scaled and time-delayed repetitions of the source, and are emitted from points of reflection in the room. In the time-domain, images represent a single pure impulse with a known scaling and time delay, where room modes represent a decaying exponential that influences all time. A limitation of the image method is that its

application is constrained to rooms of rectangular dimension. More complex techniques are required for polyhedral rooms of arbitrary dimension. However, the training data to be used in this project is constrained to a rectangular room dimension, so this limitation is not a concern. In the case of a 3D rectangular room (commonly referred to as a shoebox), symmetry causes the number of image sources to grow cubicly in the order of reflections [**?**]. Specifying the order of reflections in the calculation of image sources can constrain the number of image sources calculated for a given room.

An intuitive explanation of the ISM is shown in figure 4.3. The grey rectangular cuboid represents a shoebox room. The two black circles represent a pair of microphones and the yellow rectangle represents an acoustic point source. The turquoise rectangles represent images of the yellow point source. Each image can be represented as $\alpha_i\delta(t-\beta_i)$ for image $i = 0, 1, 2 \ldots, n$. The superposition of these images creates an impulse response as shown in figure 4.3.

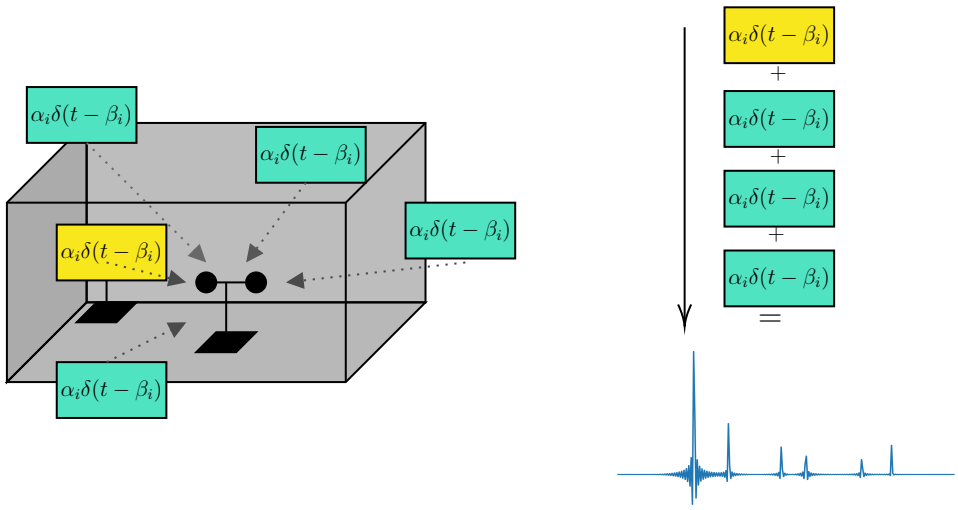

Figure 4.3: Image Source Method

The locations of the images can be calculated using an algorithm presented by Allen and Berkley in [**?**]. Once the locations of the images have been computed, the impulse response between a given microphone $\boldsymbol{r}$ and a source $\boldsymbol{s}_0$ sampled at F_s is given by equation 4.1.

$$a_{\boldsymbol{r}}(\boldsymbol{s}_0, n) = \sum_{s \in \nu_{\boldsymbol{r}}(\boldsymbol{s}_0)} \frac{(1-\xi)^{gen(\boldsymbol{s})}}{4\pi||\boldsymbol{r}-\boldsymbol{s}||}\delta_{LP}\left(n - F_s\frac{||\boldsymbol{r}-\boldsymbol{s}||}{c}\right) \tag{4.1}$$

Where $\nu_{\boldsymbol{r}}(\boldsymbol{s}_0)$ is a set of image sources and $gen(\boldsymbol{s})$ is the reflection order of $\boldsymbol{s}$. $\xi \in [0, 1]$ is the absorption factor of the walls, c is the speed of sound and δ_{LP} is the windowed sinc function given in equation 4.2, where T_ω gives the width of the window [**?**].

$$\delta_{LP}(t) = \begin{cases} \frac{1}{2}\left(1 + cos\left(\frac{2\pi t}{T\omega}\right)\right) sinc(t) & if \ -\frac{T\omega}{2} \leq t \leq \frac{T\omega}{2}, \\ 0 & otherwise. \end{cases} \tag{4.2}$$

The absorption factor $\xi \in [0, 1]$ is related to reverb time RT_{60} by Sabine's Formula,

$$RT_{60} = \frac{24ln10}{c} \frac{V}{S\xi} \tag{4.3}$$

Where V is the volume of the room in m^3 and S is the total surface area of the room in m^2.

Pyroomacoustics [**?**] is a python package that provides a fast C implementation of the ISM. Given a $3 \times 3 \times 2.5$ meter room as specified in section 3.6.1, a single source and a single microphone, computations of the image source positions for increasing reflection orders are shown figures 4.4, 4.5, 4.6.

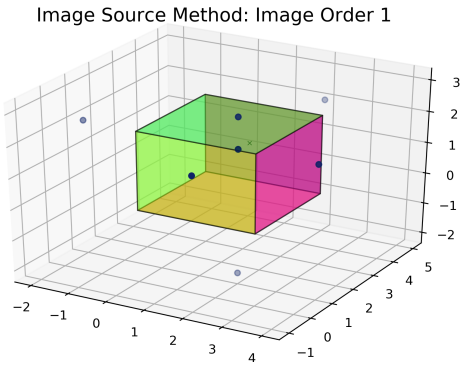

Figure 4.4: ISM: Order 1

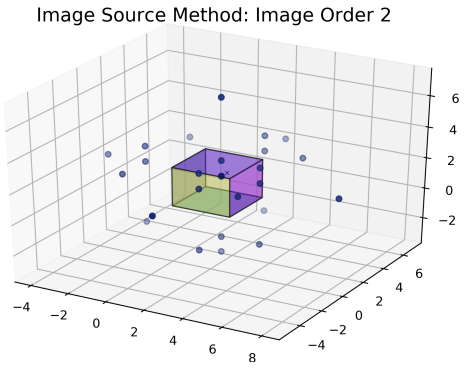

Figure 4.5: ISM: Order 2

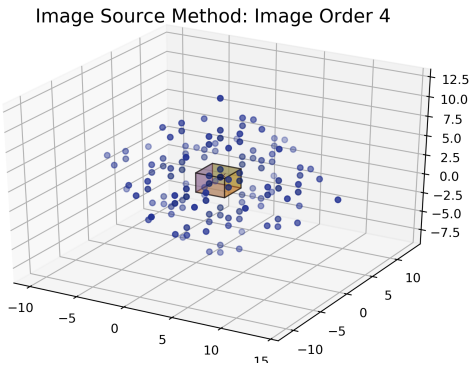

Figure 4.6: ISM: Order 4

Figure 4.11 shows the room impulse responses calculated with equation 4.1 and the sets of image sources shown in figures 4.4, 4.5, 4.6. It is clear in figure 4.11 that the RIRs approach a more realistic room response with increasing reflection order.

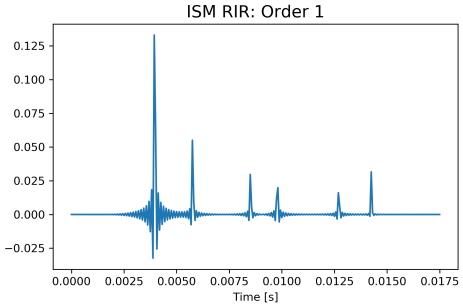

Figure 4.7: ISM RIR: Order 1

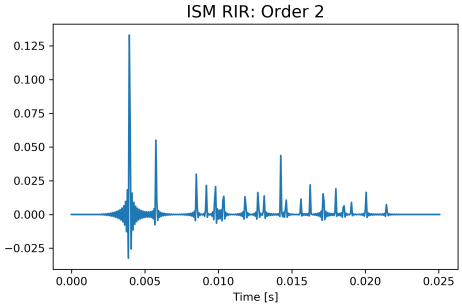

Figure 4.8: ISM RIR: Order 2

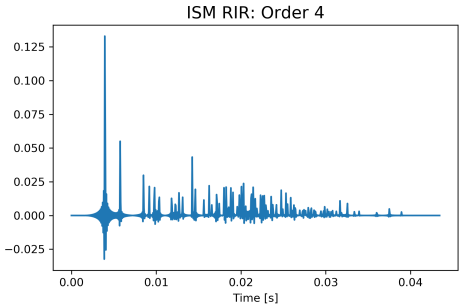

Figure 4.9: ISM RIR: Order 4

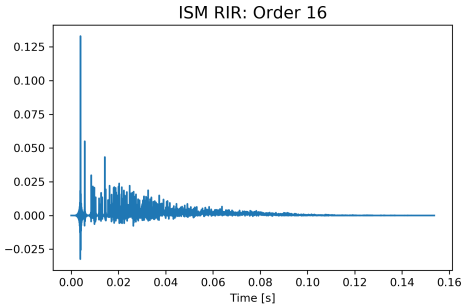

Figure 4.10: ISM RIR: Order 16

Figure 4.11: Image Source Method Room Impulse Responses

Using the ISM to compute a large dataset of RIRs can be used to synthesize a dataset of spatial audio that can be used for training a machine learning algorithm to predict the DOA of an incoming sound source. The implementation of this will be presented in section 5.1.

4.3 Classical Sound Source Localisation Techniques

4.3.1 GCC-PHAT TDOA

Generalised Cross-Correlation

Estimating the time delay between signals received at two spatially distinct sensors allows the calculation of the direction of arrival of an acoustic signal. This technique is known as time delay of arrival (TDOA) estimation. Computing the cross-correlation function

of the signals observed at each microphone is a common way of estimating the TDOA. The generalised cross-correlation proposed by Knapp and Cater in [**?**] is a method of determining the cross-correlation with an appropriate weighting function in order to improve the signal-to-noise ratio in the frequency band of interest. The choice of weighting function is dependent on application. As discussed in section 2.2, the phase transform (PHAT) weighting function is the most commonly used for speech signals. This section will present a theoretical development of the estimation of the time delay of arrival using a generalised cross-correlation with a phase transform (GCC-PHAT TDOA).

Two continous time signals, $x_1(t)$ and $x_2(t)$, are observed in the presence of noise at two spatially separated microphones. An acoustic signal $s_1(t)$ emanates from a sound source and can be modelled at the microphones as

$$x_1(t) = s_1(t) + n_1(t) \tag{4.4}$$

$$x_2(t) = \alpha s_1(t + D) + n_2(t) \tag{4.5}$$

where $s_1(t)$, $n_1(t)$ and $n_2(t)$ are real, jointly stationary random processes. The acoustic signal $s_1(t)$ is assumed to be uncorrelated with noise $n_1(t)$ and $n_2(t)$. Given that speech signals, noise, attenuation α and time delay D cannot be assumed to remain stationary over an entire period of observation, in order to estimate the time delay, Knapp and Cater propose the use of a maximum likelihood estimator. This constrains the time delay estimation to a finite observation interval. The basis of this method is the computation of the cross-correlation function

$$r_{x_1x_2}(\tau) = E[x_1(t)x_2(t - \tau)] \tag{4.6}$$

where E is expectation. The argument τ that maximizes equation 4.6 provides an estimate of time delay.

$$\hat{D} = arg \underset{\tau}{max}\ r_{x_1x_2}(\tau) \tag{4.7}$$

However, it must be noted that due to the finite observation time, $r_{x_1x_2}(\tau)$ can only be estimated. An estimate of the cross correlation is given by

$$\hat{r}_{x_1x_2}(\tau) \;=\; \frac{1}{T-\tau}\int_{\tau}^{T} x_1(t)x_2(t-\tau)dt \tag{4.8}$$

with T as the observation interval. Prior to the integration in equation 4.8, it is necessary to pre-filter $x_1(t)$ and $x_2(t)$ in order to improve the accuracy of the delay estimate $\hat{D}$. The choice of this prefilter specifies the aforementioned weighting function. Figure 4.12 illustrates this process. The signals $x_1(t)$ and $x_2(t)$ are filtered through $H_1(\omega)$ and $H_2(\omega)$ to obtain y_1 and y_2, which are then multiplied, integrated and squared for a range of time shifts τ. The time shift at the peak of this function corresponds to an estimation of the time delay D. The generalised cross-correlation method is concerned with selecting the filters $H_1(\omega)$ and $H_2(\omega)$ appropriate for the application.

The generalized cross-correlation of $x_1(t)$ and $x_2(t)$ is given by

$$r_{x_1x_2}(\tau) \;=\; \int_{-\infty}^{\infty} \psi(\omega)X_1(\omega)X_2^*(\omega)e^{j\omega\tau}d\omega \tag{4.9}$$

Where $X_1(\omega)$ and $X_2(\omega)$ are the Fourier transforms of $x_1(t)$ and $x_2(t)$ and * denotes complex conjugation. Thus the generalized cross-correlation is calculated as the inverse Fourier transform of the received signal cross-spectrum scaled by the weighting function shown in equation 4.10 where $H_1(\omega)$ and $H_2^*(\omega)$ correspond to the prefilters in figure 4.12.

$$\psi(\omega) \;=\; H_1(\omega)H_2^*(\omega) \tag{4.10}$$

$$S_{x_1x_2}(\omega) \;=\; X_1(\omega)X_2^*(\omega) \tag{4.11}$$

Equation 4.11 shows the cross power spectral density function of $x_1(t)$ and $x_2(t)$. For reasons previously discussed, only an estimate of this can be obtained. The phase transform weighting function is described in equation 4.12.

$$\psi_{PHAT}(\omega) = |S_{x_1x_2}(\omega)|^{-1} \tag{4.12}$$

Thus the final expression of the estimate of the generalised cross-correlation with phase transform is

$$\hat{r}_{x_1x_2}(\tau) \;=\; \int_{-\infty}^{\infty} \frac{\hat{S}_{x_1x_2}(\omega)}{|S_{x_1x_2}(\omega)|} e^{j\omega\tau} d\omega \tag{4.13}$$

In order to compute GCC-PHAT a discrete time formulation of equation 4.13 needs to be developed. Discrete time cross correlation can be described as

$$\hat{r}_{x_1x_2}[k] = \frac{1}{N-k} \sum_{n=0}^{N-1-k} x_1^*[n] x_2[((n+k))_N],\ k = 0, 1, 2, \ldots, L-1 \tag{4.14}$$

where $L \ll N$ is chosen in order to have enough lagged products $x_1^*[n]x_2[((n+k))_N]$ at the highest lag $L-1$ such that a reasonably accurate average in obtained [**?**]. Summation stops at $n = N-1-k$ in order to prevent circular wrap around of n *modulo* N.

Cross-correlation can be computed faster via DFT using zero padding [**?**]:

$$\hat{r}_{x_1x_2}[k] = \frac{1}{N-k} IDFT_k(X_1^* \cdot X_2) \tag{4.15}$$

Where

$$X_i = DFT[CasusalZeroPad_{N+L-1}(x_i)], i = 1, 2 \tag{4.16}$$

Thus including the phase transform weighting and adapting equation 4.13 for discrete time yields

$$\hat{r}_{x_1x_2}[k] = \frac{1}{N-k} IDFT_k \left(\frac{X_1^* \cdot X_2}{|X_1^* \cdot X_2|} \right) \tag{4.17}$$

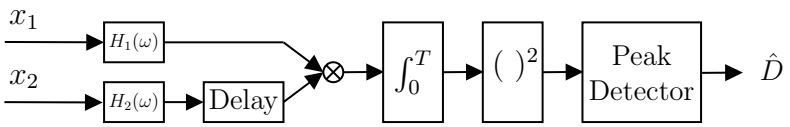

Figure 4.12: Continuous Time Generalised Cross Correlation

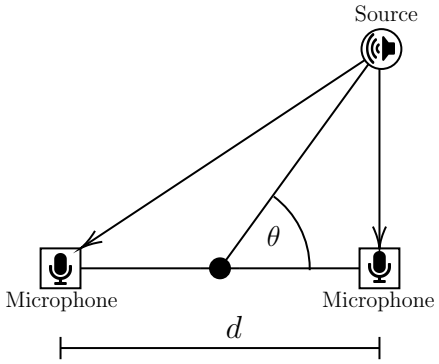

Figure 4.13: Experimental Set Up

Computation of Time Delay of Arrival

This section will discuss the algorithmic implementation of equation 4.17 in order to compute the time delay. Figure 5.28 shows a typical experimental setup with two microphones and a single audio source. The distance between the microphones is denoted as d and direction of arrival is denoted as θ. Consider the case where a decaying sinusoidal signal is propagated by the audio source. The microphones are separated by a distance $d = 0.2$ meters and the sample rate of the recording device is $8000Hz$. Figure 4.14 shows the signals captured at each microphone. These signals are identical but separated by a time delay. Figure 4.15 shows an example case where these two signals separated with a 3 sample delay. The free-field/far-field propagation model shown in equation 4.18 can be used to calculate the direction of arrival θ. Where V_{sound} is the speed of sound ($\approx 343m/s$), f_{sample} is the sample rate in Hz, d is the distance between microphones in meters and τ_{sample} is the estimated delay in samples.

$$\theta = \arcsin\left(\frac{V_{sound} \cdot \tau_{sample}}{f_{sample} \cdot d}\right) \tag{4.18}$$

If we consider that the maximum time delay that can be observed corresponds to $\tau_{max} = d/V_{sound}$, then we can simplify equation 4.18 to

$$\theta = \arcsin\left(\frac{\tau}{\tau_{max}}\right) \tag{4.19}$$

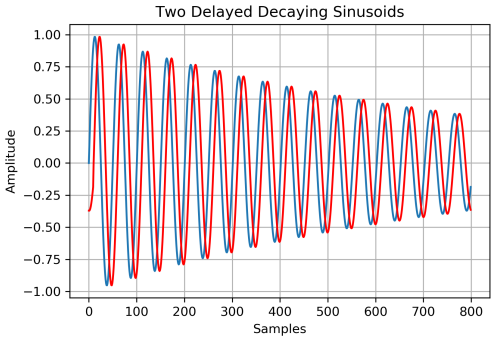

Figure 4.14: Two Delayed Decaying Sinusoidal Signals

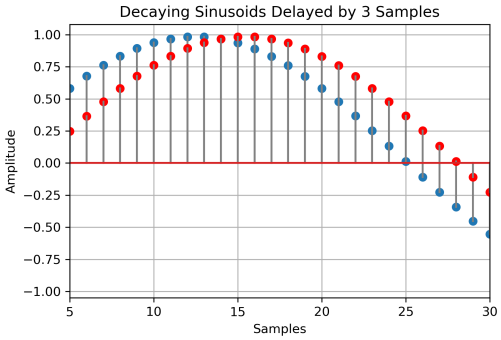

Figure 4.15: Two Identical Sinusoidal Signals With a 3 Sample Delay

Thus the direction of arrival θ corresponds to the ratio of the estimated time delay τ and the maximum time delay τ_{max}. In order to calculate τ, the maximum possible shift in samples given τ_{max} must be subtracted from the argument of the peak of the GCC-PHAT function and scaled appropriately given the sample rate.

$$\tau = \frac{\underset{\tau}{arg\ max}(r_{x_1x_2}[k]) - max\_shift}{fs} \tag{4.20}$$

In the case presented thus far the maximum time delay would be $\tau_{max} = 0.2/343 = 5.83\text{x}10^{-4}$ seconds. Thus, at the sample rate of $8000Hz$, the maximum possible shift in samples will be $max\_shift = 4$. The GCC-PHAT of the signals in figure 4.15 is shown in figure 4.16. It is clear from this figure that the maximum occurs at index 7. From this, the time delay can be calculated as

$$\tau = \frac{7-4}{8000} = 0.000375\ seconds \tag{4.21}$$

Given that we know that the two signals have a 3 sample delay and that the sample rate is $8000Hz$, we know that the true time delay must be $delay = 3/8000 = 0.000375\ seconds$. Thus the computed value of tau is sensible. Finally, the direction of arrival can be computed using equation 4.19

$$\theta = arcsin\left(\frac{0.000375}{0.000583}\right) = 40\ degrees \tag{4.22}$$

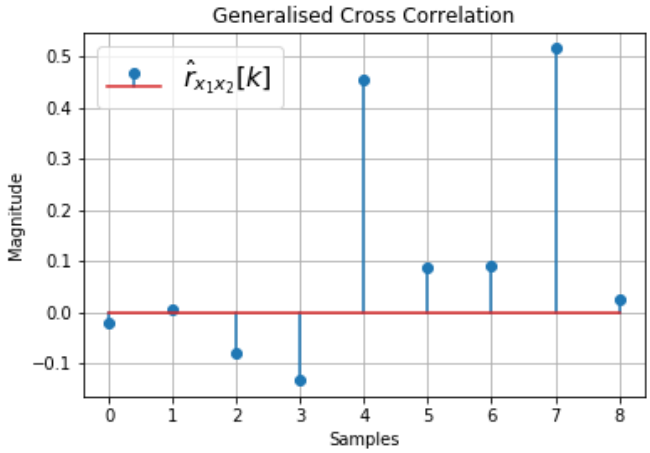

Figure 4.16: Generalised Cross Correlation

4.4 Deep Learning For Sound Localisation

4.4.1 Model Architectures for Audio Classification

As described in section 3.6.3, the focus of this project will be on using deep convolutional networks for audio classification. Figure 4.17. CNNs used for image processing traditionally utilize 2-dimensional kernels for convolutional operations. However, in the case of a 1-dimensional vector of time-domain data, a 1-dimensional convolutional operation must be used. The main difference between 1-D and 2-D CNNs is that 1-D vectors replace 2-D matrices for kernels and feature maps [**?**]. A simple example of a 1-D CNN architecture is shown in figure 4.17. Here, a raw time-domain vector is used as an input signal. Three 1-D convolutional layers are used in conjunction with two fully connected MLP layers.

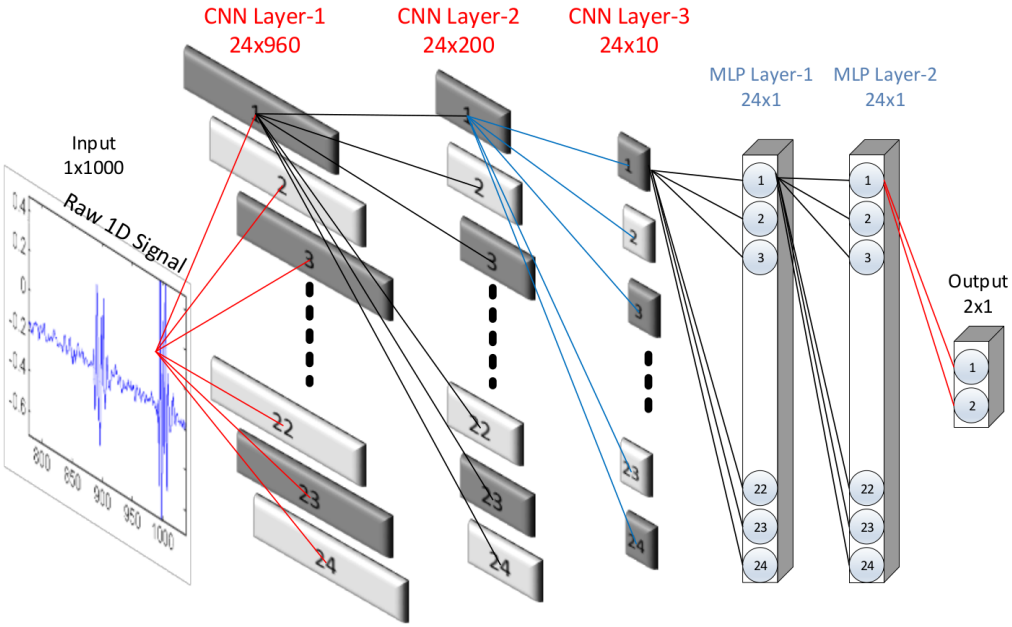

Figure 4.17: 1-Dimensional Convolutional Networks [?]

Deep convolutional networks are a massive theoretical topic beyond the scope of this work. This section will briefly elaborate only on the topics relevant for our use case. As specified, the work in [?] will be used as a reference in designing the networks for this project. The models presented in the paper are fully convolutional, without fully connected layers or drop out. The architectures use a small receptive field in the convolutional layers. However, in the first layer, a large receptive field is chosen based on the audio sampling rate to mimic the effects of a bandpass filter. Furthermore, batch normalisation is used to overcome the difficulties of training very deep models whilst keep computational expense low. For this project we will implement and modify 2 of the 5 architectures considered in [?]. The basic architecture of these models is shown in figure 4.18. Some key choices in the design of these models will now be elaborated upon.

A goal of these models is to utilize very deep architectures in an attempt to aid extraction of high-level discriminative features when used with acoustic data. However, this choice could lead to an extremely large number of parameters and computationally unfeasible model sizes. Thus to account for this a very small receptive field of 3 is used for all but the first layer. Furthermore, in the first two layers, large convolutional and max pooling strides are used to further reduce the computational complexity of the network. ReLU activation functions are used throughout the network.

A challenge of using time-domain audio waveforms sampled at a useful sample rate ($8kHz$ +) is that they could lead to a very large one dimensional input vector. For example, using one of the Google Speech Command data points with a 3 second length

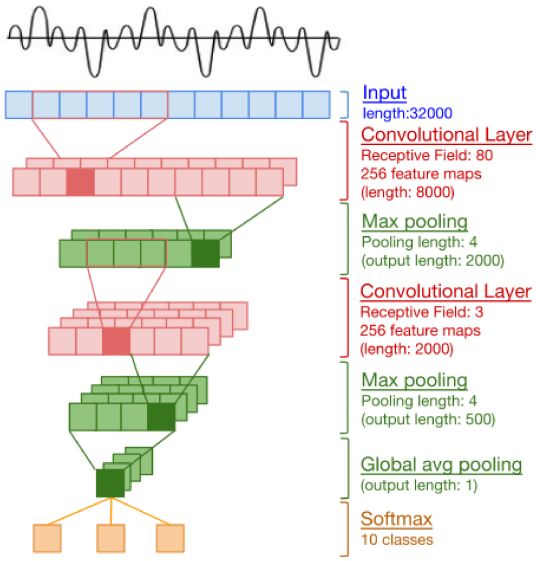

Figure 4.18: Raw Audio CNN Model Architecture [**?**]

and a sample rate of $16kHz$ would result in an input vector of dimension 1×48000. Using this in conjunction with a very deep CNN could lead to a unfeasibly large model. In order to account for this, the first layer receptive field is set to cover a 10-millisecond window. At a sample rate of $8kHz$, this would equate to a receptive field of 80.

A common problem in optimizing deep architectures is the issue of exploding or vanishing gradients. To account for this, batch normalisation is applied on the output of each convolutional layer before applying the ReLU activation function. Finally, in one of the networks to be implemented, residual learning blocks are used. This is a technique introduced to ease the training of very deep networks [**?**]. This is achieved by implementation of a skip-connection in the residual block as shown in figure 4.19. Details of the implementation of these models will be discussed is section 5.2

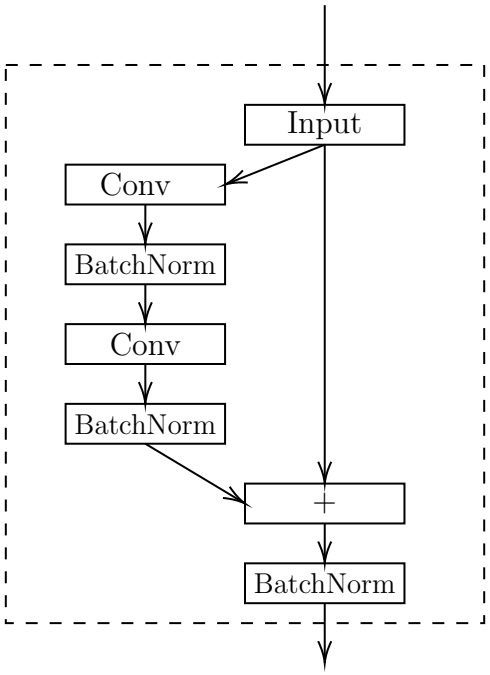

Figure 4.19: Residual Block

Chapter 5

Subsystem Design

5.1 Data Synthesis

5.1.1 Image Source Method and Room Synthesis

The Image Source Method and the synthesis of spatial audio were implemented using the Pyroomacoustics python package [**?**]. The dataset used as the anechoic audio was the Google Speech Commands Dataset [1]. Using these two datasets, the propagation of a source (a command from the Google Speech Commands Dataset) to a pair of microphones in a room could be synthesized and stored as a stereo Wav file. A wrapper class `binaural.py` [2] was written around the Pyroomacoustics package to implement this specific use case. An instantiation of a `Binaural` object specifies the following constants:

- The room dimensions
- The reflection order of the image source method
- The speed of sound
- The distance between microphones (referred to in the code as the inter-aural mic distance)
- The height of the microphones in the room

[1] Available: https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html

[2] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/binaural.py`

Following this, a impulse response pair (one for each microphone) can be generated and written to a stereo Wav file using the function `generate_impulse_pair()`. Arguments of this function can specify the following degrees of freedom:

- The rotation of the source from the center of the room
- The distance of the source from the center of the room
- The signal-to-noise ratio of the recording
- The reverb time of the room (calculated using Sabine's Formula)
- The location of the center of the microphone array
- The microphone rotation in degrees
- The sample rate
- The source signal (a recording from the Google Speech Command Dataset)

A plotting function was written to display the room configuration when a pair of impulse responses are recorded. Given the degrees of freedom presented above, for a fixed room size, an extremely large dataset of individual room configurations, noise levels, reverb times and sound sources can be synthesized. Some examples of these configurations are shown in figure 5.1. The red dots represent the microphone pair and the triangle points in the frontal direction of the binaural configuration. The yellow circle is the sound source. A Wav file containing a single-channel, anechoic recording of a clap was used in synthesizing an example pair of microphone recordings shown in figure 5.2. These recordings were generated using one of the configurations shown in figure 5.1.

5.1.2 Parallelism

Synthesising a dataset using the method described in the previous section is computationally expensive. Experiments run using the python package `tqdm`[3] indicated that generating a dataset of $10,000$ points serially would have taken approximately 4 days on the author's laptop [4]. Ultimately, a much larger dataset was needed to introduce sufficient variation in the data in the pursuit of training a useful model. Thus it was required to modify the data synthesis software for parallel execution. Considering that the computation

[3]Available: `https://github.com/tqdm/tqdm`

[4]Specs: Intel i5 quad-core processor, 12Gbs of Ram

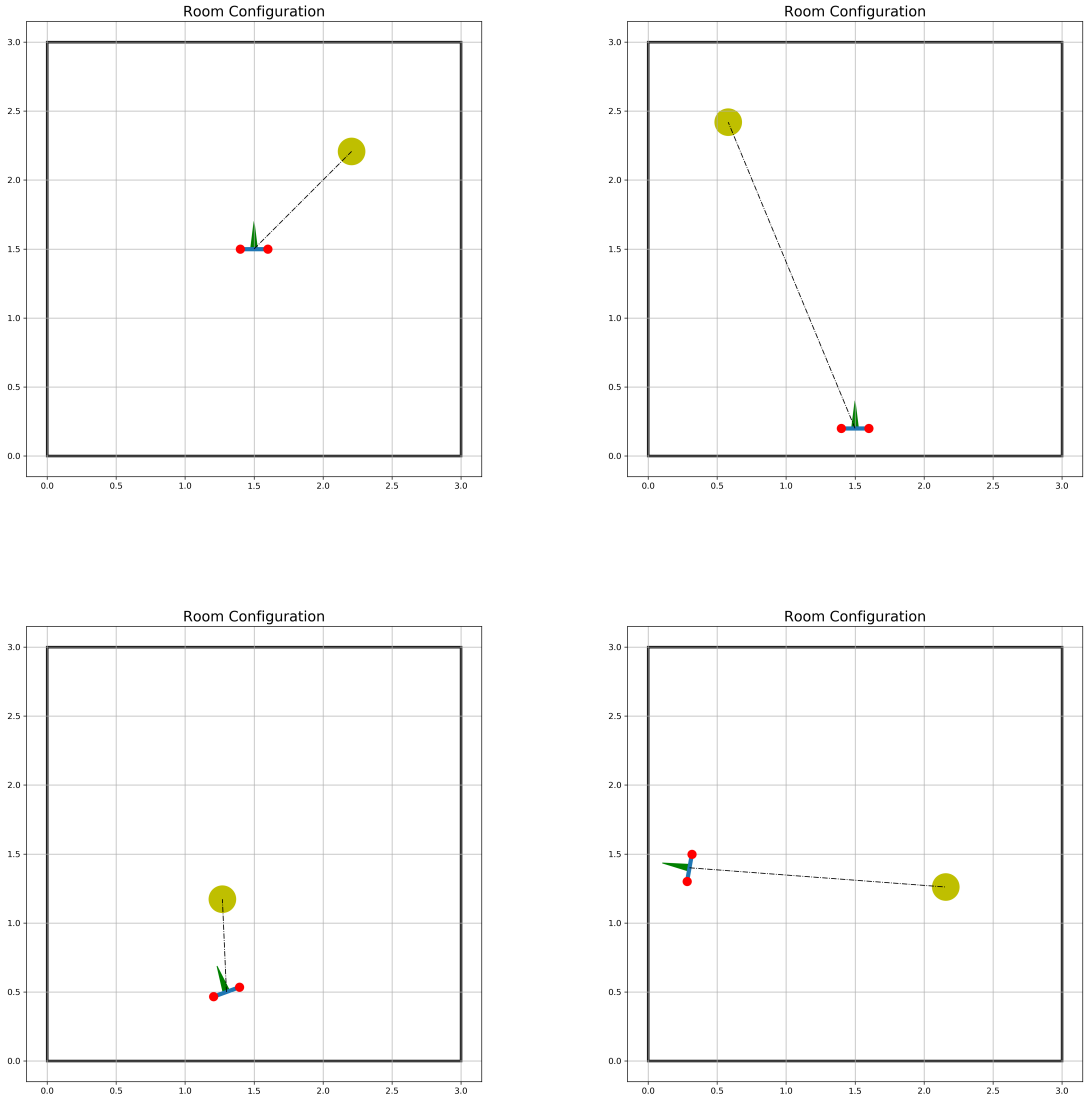

Figure 5.1: Room Configurations

of each stereo wav file was it's own self-contained operation with no reliance on shared memory, concurrency issues associated with data-parallelism did not have to be taken into account. Thus an asynchronous process-based parallelism approach was taken. This was implemented using the python `multiprocessing` library. The implementation of this can be seen in `generate_data.py`[5]. An issue that was encountered was the problem of generating unique filenames for a large number of wav files. The solution to this problem was to use the hex value of the uuid of the process that generated the file as the filename. The data was stored in a `csv` file as with the wav name in one column and the associated direction of arrival in another.

[5] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/generate_data.py`

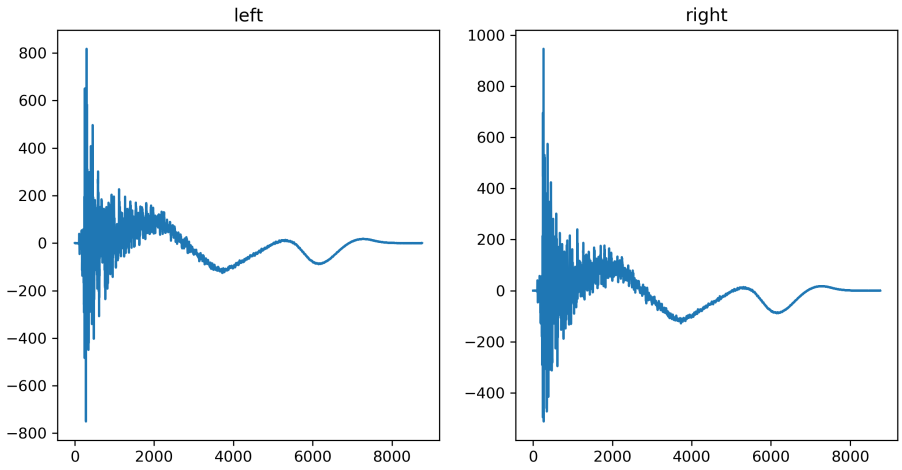

Figure 5.2: Real Room Impulse Response

This implementation provided a significant speed up, reducing the estimated computation time on the aforementioned laptop to just under 1 day. However, this was still a prohibitively long time for a relatively small dataset. Furthermore, the laptop consistently ran out of ram when running experiments. In order to remedy this, the program was deployed on a Google Cloud Platform (GCP) instance with 24 cores and 128Gbs of RAM. This provided a significant speed-up, reducing the computation time to under 30 minutes.

5.1.3 Datasets

Four datasets were designed for experimentation, training and blind evaluation. Each dataset was stored as a `.tar.gz` archive of wav files, and a `.csv` containing the data labels. These datasets are described in table 5.1. *Fixed Variables* specifies variables that were kept constant throughout the synthesis of the data. *Free Variables* represent the degrees of freedom in the dataset. Each dataset is comprised of individual combinations of free variables. The Experimentation and Training datasets were synthesised for the room dimension specified by the project. The Unseen Blind evaluation dataset was synthesised for a room double the size. The Seen Blind evaluation dataset was synthesised for the same size room as Experimentation and Training, but with microphone and source configurations unseen in the Training dataset. The remainder of this section will present a brief description of each dataset.

Dataset	Size	DOA Range	Fixed Variables	Free Variable
Experimentation	70,000	0 -180	SNR Reverb Time Source Distance Mic Centre Mic Rotation	Source Azimuth Sound Source
Training	388,800	0-355	SNR	Sound Source Source Azimuth Reverb Time Source Distance Mic Rotation Mic Center
Unseen Blind Evaluation	1100	0-355	SNR Reverb Time Source Distance Mic Centre Mic Rotation	Source Azimuth Sound Source
Seen Blind Evaluation	1100	0-355	SNR Reverb Time Source Distance Mic Centre Mic Rotation	Source Azimuth Sound Source

Table 5.1: Datasets

Experimentation

This dataset was generated for the purposes of evaluating data preprocessing techniques and deep learning models. The DOA labels present in the data were limited to a range of 0° to 180° with increments of 18°. The microphone configuration was placed in the room center with no rotation. The following constants were chosen to represent reasonable operating conditions:

- SNR = $0dB$
- Reverb Time = $0.3s$
- Source Distance = $1m$

Training

Once the useful models and preprocessing techniques had been established, a large dataset of 388,800 points was synthesised for the purpose of training the models. This dataset

contained DOA labels for every angle between 0° and 360° with a 5° resolution. Only the SNR was held constant at $-20dB$. This value was chosen in the pursuit of noise robustness as this would ensure that the models were trained on data with a relatively high noise level. In a similar vein, reverb times were varied from 0.3 to 0.7 seconds to ensure the model was exposed to adverse operating conditions. Finally, the source distance, mic rotation and mic center were all varied throughout the dataset to ensure robustness to room configuration changes.

Blind Evaluation

Two blind evaluation datasets, Seen and Unseen, were created to evaluate the models on data that was completely different to that which it had been trained on. Here, *seen* refers to room dimensions seen in training. The Seen dataset has the same room dimensions as the training set, with a different experimental configuration. The Unseen dataset has room dimension of twice the dimension of the specified dataset. These datasets were used as a test of a model's ability to generalize to new environments. The reverberation time was set to 0.7 seconds. The microphone set up and the source were arbitrarily positioned in each room, without including positions used in training. The SNR was set to $-20dB$.

5.2 Models

5.2.1 GCC-PHAT TDOA

This section presents an implementation of the GCC-PHAT TDOA technique described in section 4.3.1. The method was implemented in the python file `gccphat.py` [6] . A problem encounted when implementing the GCC-PHAT model was that, given the chosen sample rate of 8kHz (explained in section 5.2.2), the GCC-PHAT vector did not have sufficient resolution to accurately calculate the time delay. This was solved by introducing interpolation by increasing the inverse `fft_size` as shown in the code snippet below:

[6] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/gccphat.py`

Listing 5.1: Interpolation in GCC-PHAT

```
import numpy as np

interpolation=16

# GCC-PHAT
SIGNAL = np.fft.rfft(signal, n=fft_size)
REFERENCE_SIGNAL = np.fft.rfft(reference_signal,
                               n=fft_size)
R = SIGNAL * np.conj(REFERENCE_SIGNAL)
cross_correlation = np.fft.irfft(R / np.abs(R),
                                 n=(interpolation *
                                    fft_size))
```

The results of introducing this interpolation are shown in figure 5.3. An interpolation factor of 16 was used in the GCC-PHAT model. A futher justification for the use of interpolation is in the application of the GCC-PHAT vector as the input to the CNN model. Using an interpolation factor of 1 would possibly provide insufficient information with which to extract feature representations. As can be seen in figure 5.3, an interpolation factor of 16 significantly increases the resolution of the GCC-PHAT vector.

5.2.2 Deep Learning Models

Three deep learning models where designed based on the recommendations in [**?**] [7]. These models are named and briefly described in table 5.2. Two models use raw time-domain audio as the input and the other uses the GCC-PHAT vector calculated from the left and right audio channels. The models were implemented using Keras [**?**], a popular high-level API for the deep learning framework Tensorflow [**?**]. These models were implemented in the file `deep_learning_models.py` [8]. All of the models used ReLU activation functions. Furthermore, each model used the Adam optimizer and the Categorical Cross-Entropy loss function. The same training methodology was applied to each model. The Keras callback `ReduceLROnPlateau()` was used to reduce the learning rate if the validation accuracy did not improve over 10 epochs. In a similar fashion, the `EarlyStopping()`

[7] The code used in this paper was made publicly available in [**?**], and was used as a reference for my implementation

[8] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/deep_learning_models.py`

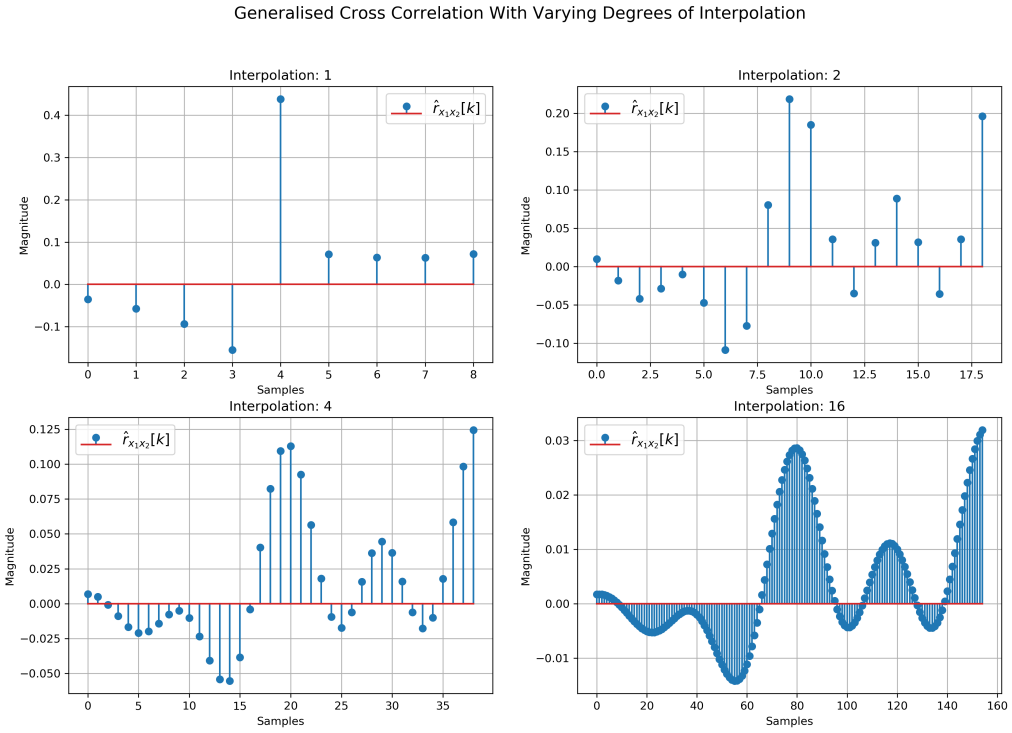

Figure 5.3: GCC-PHAT Interpolation

callback was used to end the training if the validation accuracy did not improve for 15 epochs. A batch size of 128 was used and the models were trained for 200 epochs. A Google Cloud Compute instance with a Tesla P100 GPU was used for training. Details on training times and results will be discussed in Chapter 8.

Model Name	**Input**	**Description**
Raw CNN	Raw time domain audio	Deep conv net
GCC CNN	GCC-PHAT vector	Deep conv net
Raw ResNet	Raw time domain audio	Deep conv net with residual layers

Table 5.2: Deep Learning Models

Raw CNN

The architecture of this model is shown in figure 5.4. This was adapted from model *M11* described in [**?**]. The input shape was chosen to be a time domain waveform of shape 16000×1. This is based on a fixed audio input 2 seconds in length and sampled at 8kHz. A fully connected layer before the final softmax layer was added during an early stage of development to allow the possibility of transfer learning at a later stage. The multiplications in figure 5.4 represent repetitions of the layer configurations that were

omitted for conciseness.

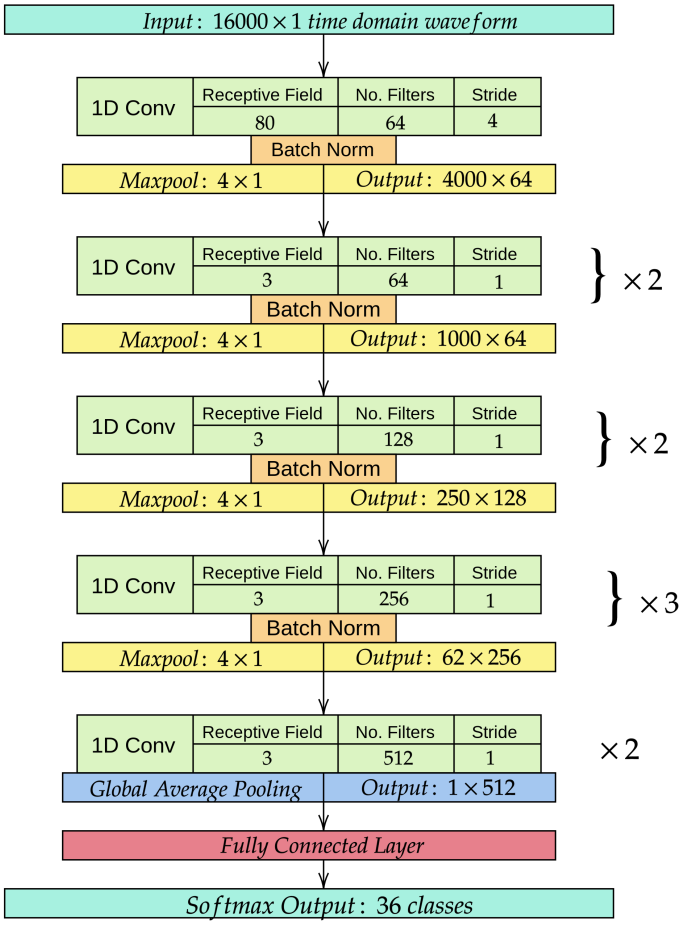

Figure 5.4: Raw Audio CNN Architecture

Raw ResNet

The architecture of this model is shown in figure 5.5. This model is adapted from model *M32* described in [**?**]. The input layer is the same as in *Raw CNN*. The residual layers used are described in the green block. Similarly to *Raw CNN*, a fully connected layer was added before the final softmax layer to allow transfer learning if needed.

GCC CNN

The architecture of this model is the same as *Raw CNN* with some slight modifications. Instead of using raw audio waveforms as network input, this model used the interpolated GCC-PHAT vector described in section 5.2.1. This dramatically reduces the size of the

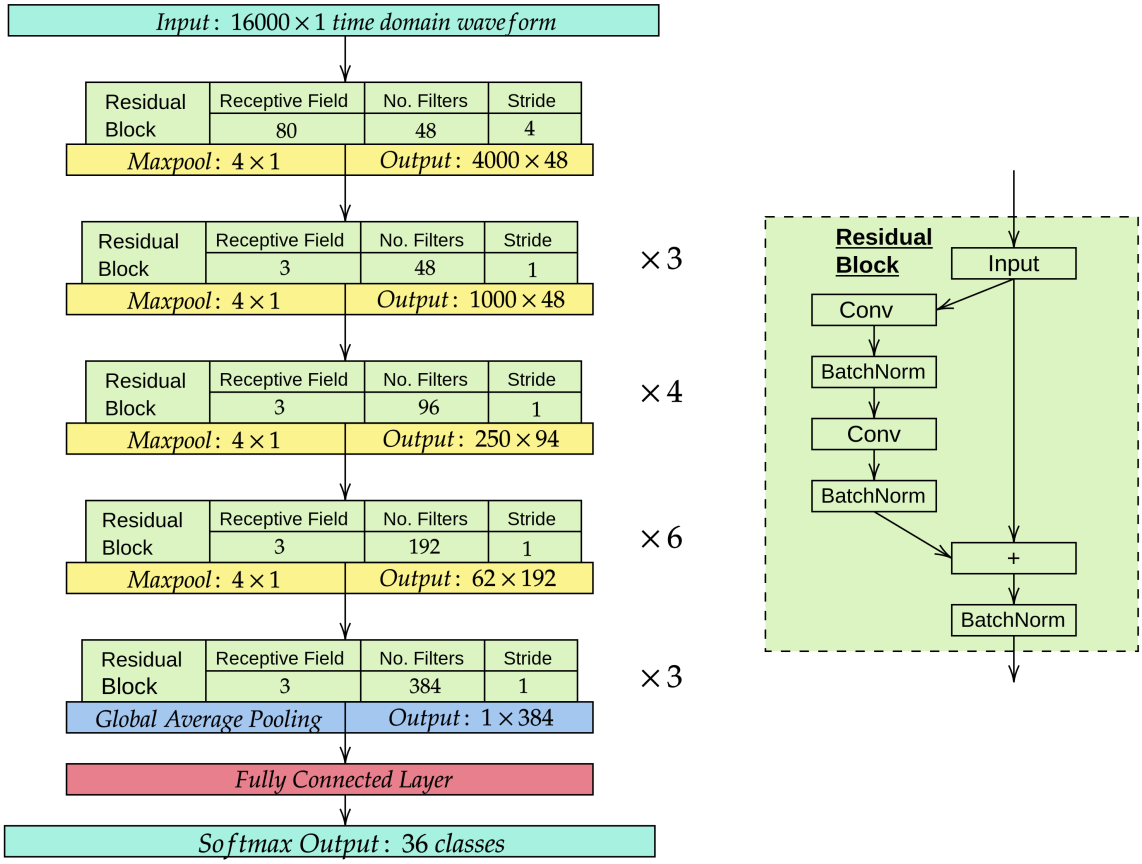

Figure 5.5: Raw Audio ResNet Architecture

input layer to (149×1). The only other modification made to the model was reducing the stride in the first convolutional layer by half.

5.3 Data Preprocessing and Labelling

5.3.1 Labelling

Considering that the problem at hand is a classification problem, the most obvious method to label the datasets was to label each datapoint with an integer value corresponding to its DOA. For the *Training* dataset, this corresponded to 72 integer labels, each corresponding to a specific DOA in a range of 360° at 5° intervals. This labelling technique is shown in figure 5.6. The models were initially tested by training *Raw CNN* with the *Evaluation* dataset. This test achieved promising results as a validation accuracy of 92.34% was achieved. However, when the same method was applied to the *Training* dataset, a validation accuracy of 48.75% was achieved. This vast degradation in performance was

explained by the fact that the *Evaluation* dataset only contained DOAs within the frontal hemifield. Once the range of DOAs extended to the full 360° range, front-back confusions as described in section 4.1.2 predictably caused the accuracy of the model to fall by approximately half.

The solution to this problem was to pre-emptively account for the problem of front-back confusions by labelling the datasets in a way that would give each front-back DOA pair the same integer label. With this method, instead of predicting the exact DOA class, the model would predict a front-back pair. This labelling method is shown in figure 5.7. Training a model with this labelling would account for the uncertainty introduced by front-back confusions. However, the use of this labelling technique would mean that a system of rotations would have to be developed to determine the true DOA given a predicted front-back pair. With this labelling technique the accuracy of *Raw CNN* on the *Training* dataset dramatically improved. The performance of all the models will be discussed in Chapter 8.

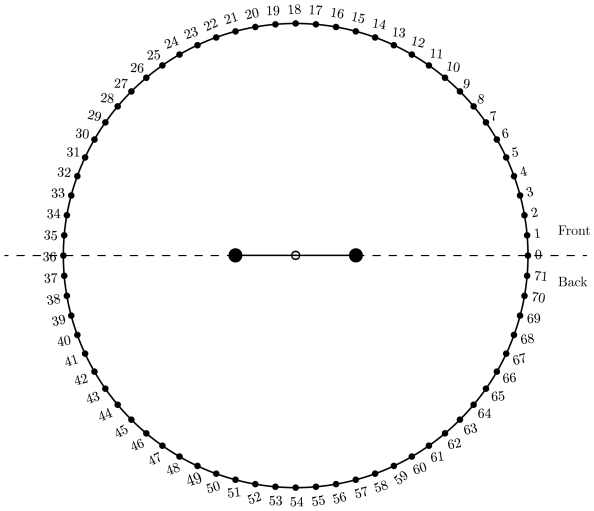

Figure 5.6: Full Circle Labelling

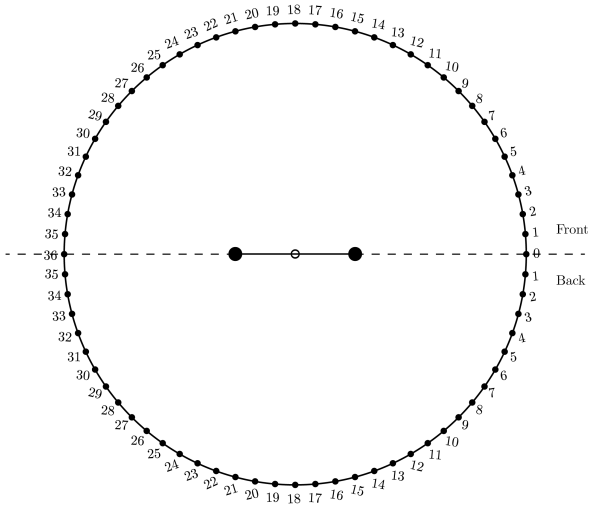

Figure 5.7: Front-Back Labelling

5.3.2 Data Preprocessing

The raw audio CNN architectures described in section 5.2 have 1-dimensional input layers. The stereo wav files are 2-dimensional in nature. To reduce the dual channel audio to a single vector, the left channel and the right channel of the file were concatenated as shown in figure 5.8.

Due to the fluctuating length of the recordings in the Google Speech Commands dataset, each synthesised data point was of different length. Thus it was necessary to truncate

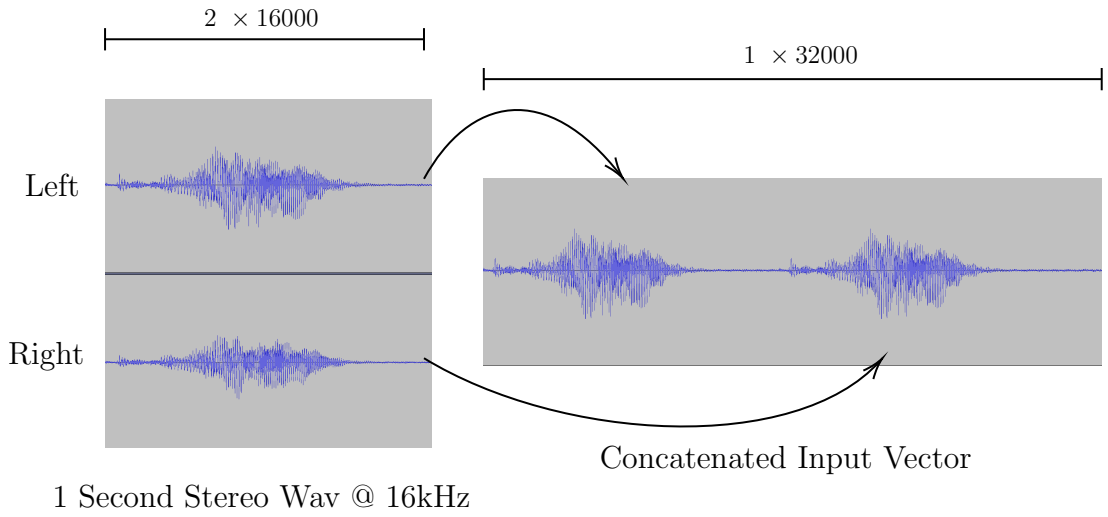

Figure 5.8: Stereo Audio Concatenation

each sample to a fixed length. This length was chosen as 1 second. Recordings that were longer than 1 second were truncated and those that were shorter underwent zero-padding. All of the datasets were synthesised at 16kHz, the same sample rate as the Google Speech Commands dataset. In the case both models *Raw CNN* and *Raw Resnet*, as shown in figure 5.8, this would require a network input layer of dimension 1×32000. As discussed in section 4.4.1, an input dimension of this length could lead to a very large model. To avoid this, the audio was downsampled to 8000kHz, reducing the input shape to 1×16000. Finally, a simple normalisation was applied to achieve a mean of 0 and a standard deviation of 1. The complete data preprocessing pipeline for the models that use raw audio inputs is shown in figure 5.9.

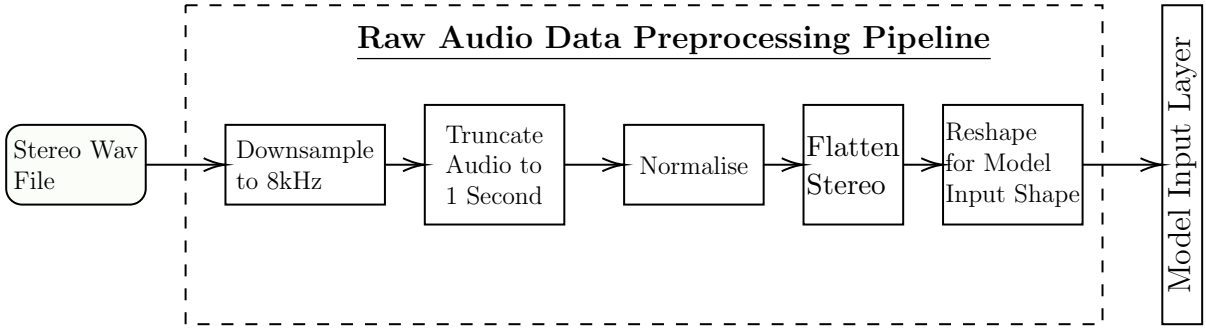

Figure 5.9: Data Preprocessing Pipeline

In the case of *GCC CNN*, the data preprocessing pipeline is much the same, with the exception of a block to compute the GCC-PHAT vector in place of the block that flattens the audio. The full pipeline is shown in figure 5.11 and the full data preprocessing and model prediction process, with the GCC-PHAT vector, is shown in figure 5.10.

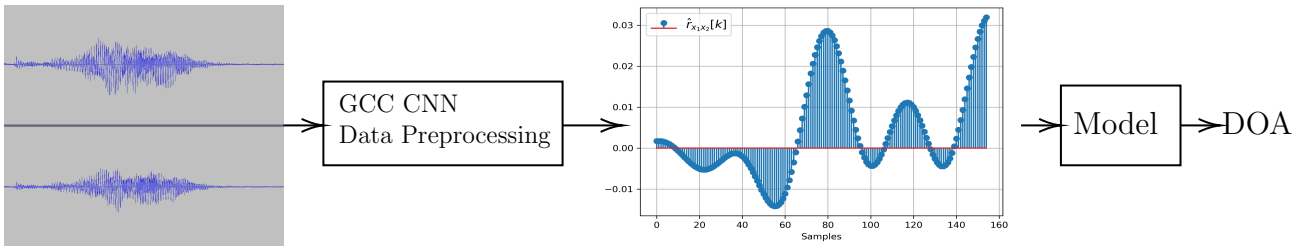

Figure 5.10: GCC CNN Data and Model

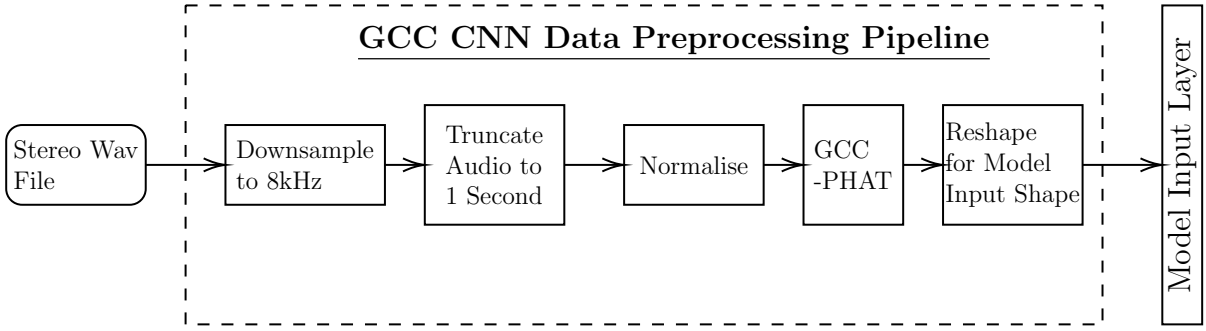

Figure 5.11: GCC CNN Data Preprocessing Pipeline

5.4 Rotation Model

A rotation algorithm had to be designed to account for the ambiguity introduced by front-back confusions. An activity diagram describing this model is shown in figure 5.12. Predictions are represented as turquoise dots and the recording apparatus is represented as the connected black circles with an arrow to indicate the frontal direction. First, a prediction is made with the recording apparatus in its initial configuration. Due to the labelling method used, any pair of DOAs contained in a single prediction will either fall in the left or right hemifield. Thus for the first rotation, labelled in red, the recording apparatus will rotate 90 deg either clockwise or counter-clockwise depending on which hemifield the predictions appear in. In figure 5.12, the regions shaded green indicate the locations of highest probability of an accurate DOA prediction. Once the first rotation is made, another prediction is run. Following this, if any predicted DOA has occurred twice, the model returns this value as the DOA and exits. If no prediction has occurred twice, the recording apparatus will rotate to face the quadrant with the highest prediction count. Another prediction is then made and the prediction list is checked for repeated predictions. After two rotations, a generic rotation loop is entered. A rotation count is kept and rotation-prediction pattern is implemented. This pattern is described in pseudo-code in the grey block in figure 5.12. If 6 rotation-prediction patterns occur without a single repeated prediction, the mean of the predictions in the quadrant with the highest prediction count is returned as the predicted DOA.

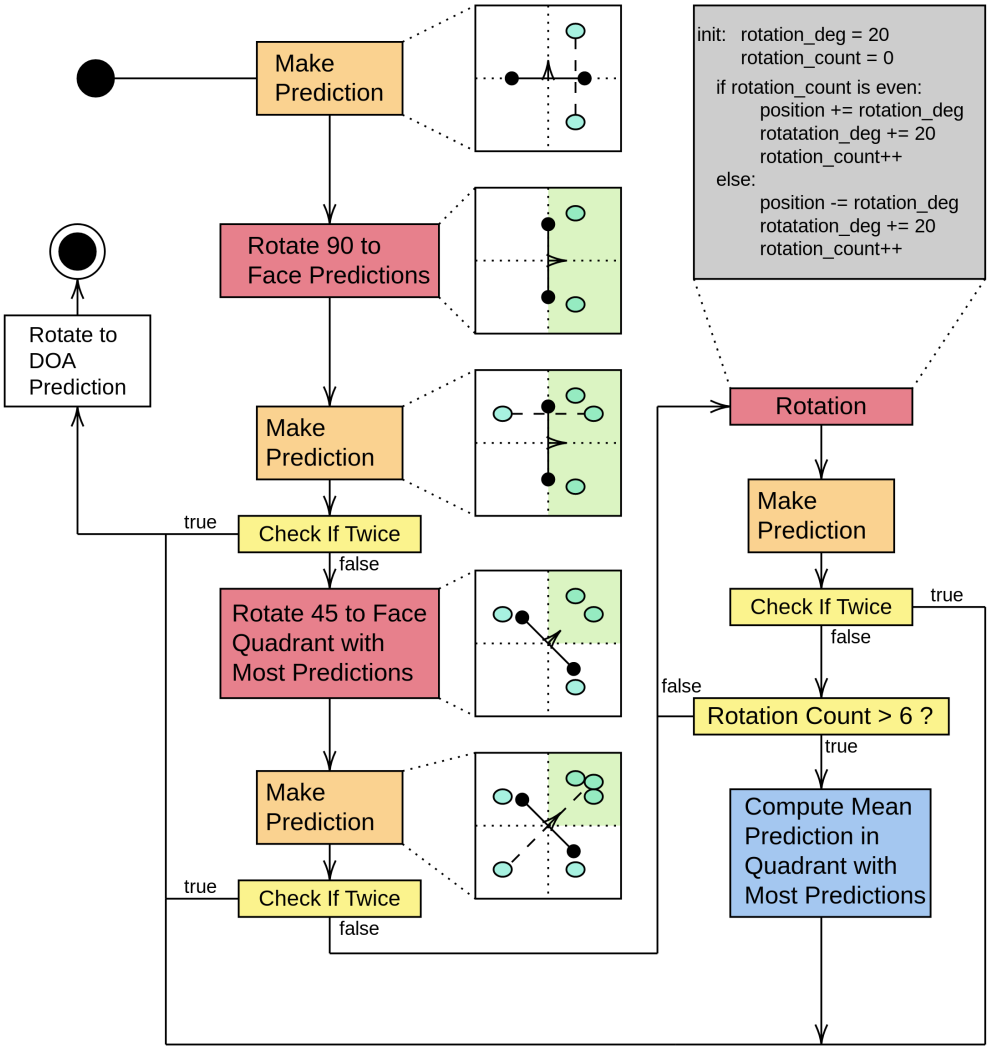

Figure 5.12: Rotation Model Activity Diagram

A simulated example of this rotation model making predictions using *GCC CNN* is shown in figure 5.13. The sound source is represented by the yellow circle, the predictions by black circles and the final prediction by a blue circle. In this case, the ground truth DOA was 70°. Two rotations are made until a prediction occurs twice, at which point the apparatus rotates to face the prediction. In this case, the DOA was correctly predicted as 70°.

A second simulation, using the *GCC CNN* model, is shown in figure 5.26. In this case, the ground truth DOA is 5° and is once again shown as the yellow circle. In this case, no prediction is made twice and thus the algorithm does not exit early, completing all 6 rotation iterations. As can be seen in figure 5.25, by the time the algorithm exits, the predictions have clustered near the true DOA. Given that no prediction occurs twice,

Figure 5.13: Rotation Algorithm: 70°

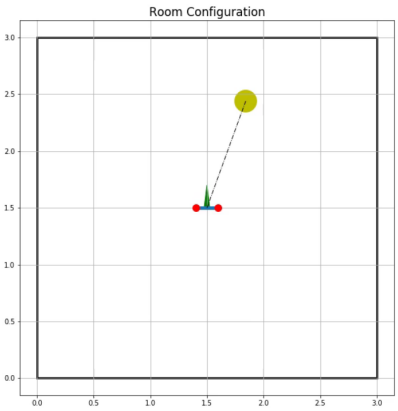

Figure 5.14: Initial Condition

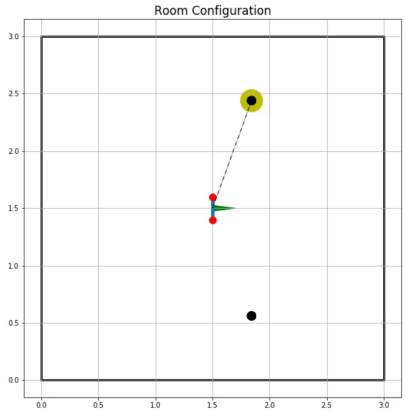

Figure 5.15: First Prediction and Rotation

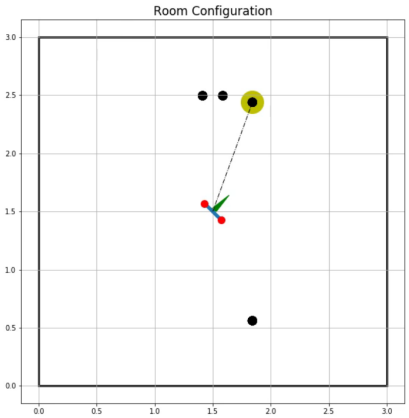

Figure 5.16: Second Prediction and Rotation

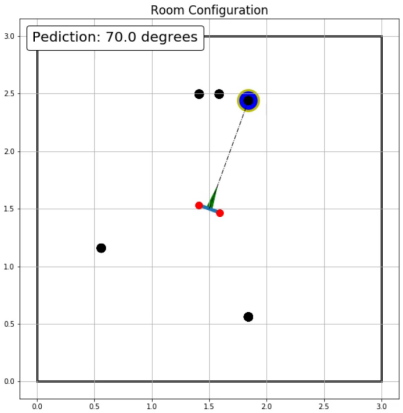

Figure 5.17: Final Prediction and Rotation

the mean value of the predictions in the quadrant with the highest prediction count is returned as the DOA prediction. In this case, the final prediction is 7°. In the practical implementaion of this algorithm, this prediction is rounded to the nearest 5° interval. Thus the final prediction is correct. On the other hand, this example highlights an inherent flaw with the algorithm in its current form. As evident in figure 5.21, the correct prediction is made only in the second iteration, and thus the algorithm could exit early and dramatically reduce execution time. This is a problem that could be solved in future work. A final note on this example is that predicting DOAs that are close to quadrant boundaries is another challenge to the algorithm as it stands. This is due to the fact that the number of predictions in two quadrants could end up being equal. As evident in figure 5.25, this was nearly the case in this example. If the fourth quadrant had the highest prediction count, the error in the mean prediction would have been significant. This problem did occur in simulation, but not sufficiently frequently to warrant a modification to the algorithm. However, this is another problem that could be

addressed in further research. A python module `rotate.py`[9] was written to implement the rotation calculations.

Figure 5.18: Rotation Algorithm: 5°

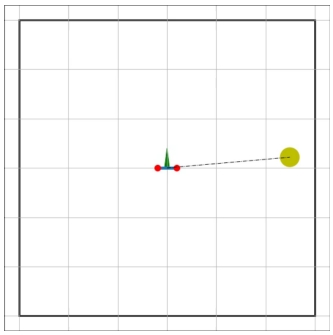

Figure 5.19: Initial Condition

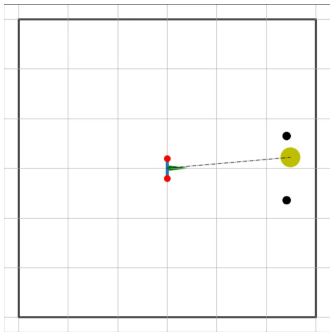

Figure 5.20: 1st Prediction and Rotation

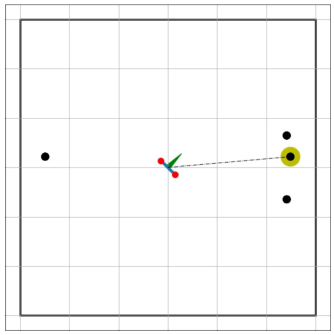

Figure 5.21: 2nd Prediction and Rotation

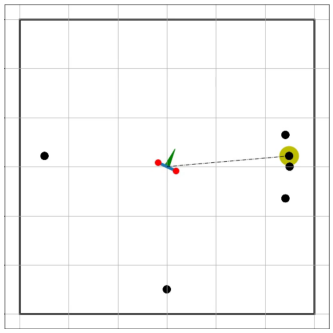

Figure 5.22: 3rd Prediction and Rotation

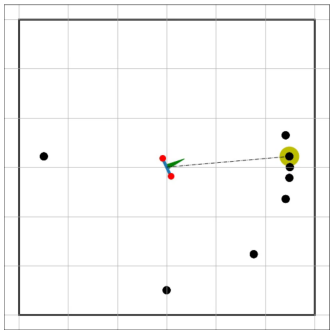

Figure 5.23: 4th Prediction and Rotation

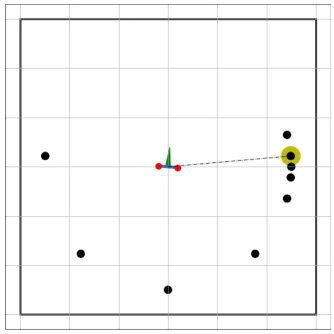

Figure 5.24: 5th Prediction and Rotation

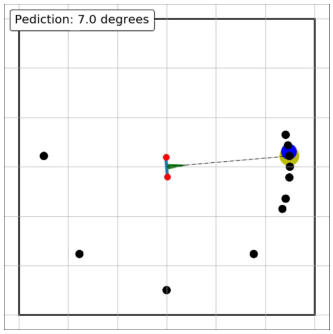

Figure 5.25: Final Prediction and Rotation

Figure 5.26: Rotation Algorithm: 5°

[9] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/rotate.py`

Figure 5.27: Apogee Duet Audio Interface

Figure 5.28: Waveshare Sound Sensor

5.5 Hardware

A Raspberry Pi Model B+ [**?**] was chosen as the embedded platform for the hardware implementation. This choice was made due to the easy availability of the device. The design choices in this section were informed by this choice.

5.5.1 Audio

The Raspberry Pi model 3 B+ has no built in ADC, and thus the use of an External ADC is required. Human hearing is approximately limited to a frequency range of $20Hz - 20kHz$. Using this range as a guideline for the frequency response requirements of the system, according to the Nyquist sampling theorem an ADC sample rate of at least $40kHz$ is required. Most commonly available ADC ICs such as the ADS1115 [**?**] can only sample at frequencies below $1kHz$. Thus in order to record audio on to the Raspberry Pi an external audio interface needs to be used. The interface utilised will be an Apogee Duet [**?**], a two channel audio interface with a sample rate of $44.1kHz$. This device is shown in figure 5.27. The audio interface will be connected to the Raspberry Pi via a USB connection and powered by a separate $5V$ $1.5A$ power supply. In terms of microphone, many inexpensive sound sensors exist. Two Waveshare Sound Sensors [**?**] were chosen for the project. The Waveshare sensor is shown in figure 5.28. The sensor has an onboard LM386 audio power amplifier and has an adjustable gain of up to $200dB$. To access the audio interface and record audio the Python Library Pyaudio [**?**] will be used. The complete audio pipeline is shown in figure 5.29. The python implementation of this can be seen in the module `audio_recorder.py` [10] .

[10] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_hardware/audio_recorder.py`

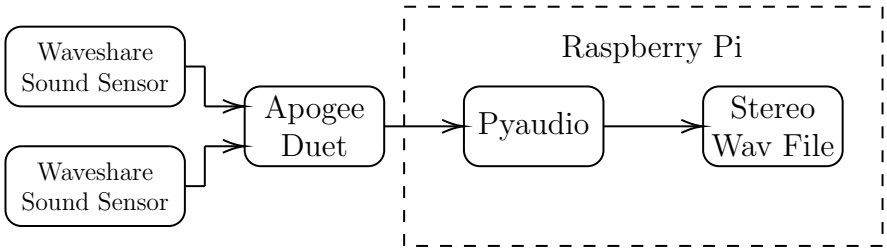

Figure 5.29: Audio Subsystem Pipeline

The positioning of the microphones is important for this application. Taking inspiration from the positioning of the human ears, the microphones are positioned $20cm$ apart.

5.5.2 Motor

The stepper motor chosen was a 0.28Nm NEMA17[11] with a 1.8 degree per step resolution. A torque of 0.28Nm is more than sufficient to rotate the small microphones. The step resolution was the best available at a reasonable price point. However, a step resolution of 1.8 degrees per step will introduce a degree of error into the actual rotation angle performed. For example, if a rotation of 85 degrees is required, with amount of steps calculated as `amount_steps = int(degrees) / 1.8`, an actual rotation of 84.6 degrees will be performed. This margin of error is acceptable given the technical requirement of a 5 degree prediction resolution.

An A4988[12] stepper driver was chosen. These are versatile, inexpensive and readily available which made it the obvious choice. A schematic showing the connection of the Raspberry Pi to the A4988 is shown in figure 5.30.

In order to implement the rotation, a wrapper class `motor.py` [13] was written around the python library. *RpiMotorLib*[14].

[11]Datasheet: `https://www.openimpulse.com/blog/document-viewer/?pdf_file=42BYGHW208-Stepper-Motor-Datasheet.pdf`

[12]Datasheet: `https://www.pololu.com/file/0J450/A4988.pdf`

[13]Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_hardware/motor.py`

[14]https://github.com/gavinlyonsrepo/RpiMotorLib

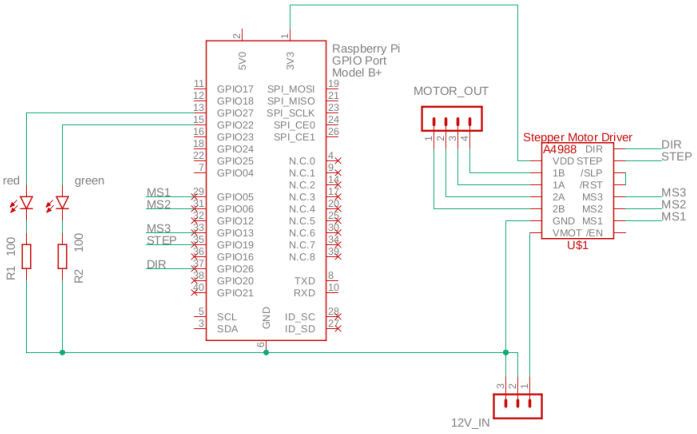

Figure 5.30: Hardware Schematic

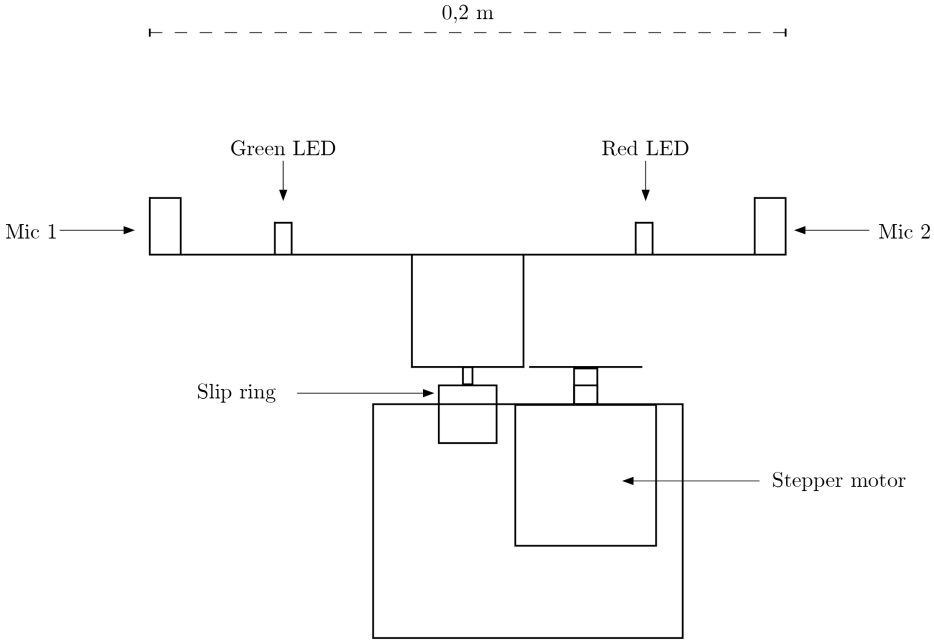

Figure 5.31: Hardware Physical Design

5.5.3 Housing

In order to perform 360 degree rotation of the microphones, the design incorporates a slipring. To house the stepper motor and slipring, an enclosure was 3D printed. The design of the enclosure was adapted from an open source design of a Lidar Scanner housing [**?**]. A mock up of the physical design of the hardware system is shown in figure 5.31. To perform rotation, gears are attached to the stepper motor and slipring.

Chapter 6

Subsystem Integration

6.1 Data, Models and Training

The integration of the Data, Models and Training subsystems will be presented as a discussion of model training, as this was the point in the design and implementation process in which these subsystems were used together to form a final set of models. As described in section 5.2.2 Each model was trained using the same hyperparameters and GCP instance. Model checkpoints were used to save the weights of the model with the highest validation accuracy. Each model was trained on the *Training* dataset. The dataset was split into 90% for training and 10% for validation. Each model was initialised to train for 200 epochs, however, due to the use of early-stopping, the number of epochs trained varied from model to model.

6.1.1 GCC CNN

GCC CNN is the smallest model of the three considered in this project. Thus it is not surprising that the training time was significantly shorter than that of the other two models. Training took approximately 3hrs, and lasted for just over 100 epochs. The training history is shown in figures 6.1 and 6.2. The sharp improvements apparent in both the model accuracy and model loss can be explained by a reduction in learning rate triggered by the `ReduceLROnPlateau()` callback, after no improvement in validation accuracy occured for 10 epochs. The model achieved a validation accuracy of $\approx 99\%$.

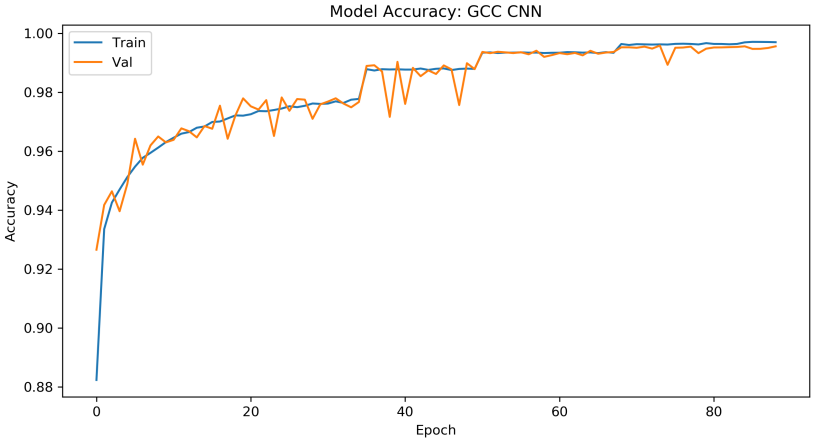

Figure 6.1: GCC CNN Training: Accuracy

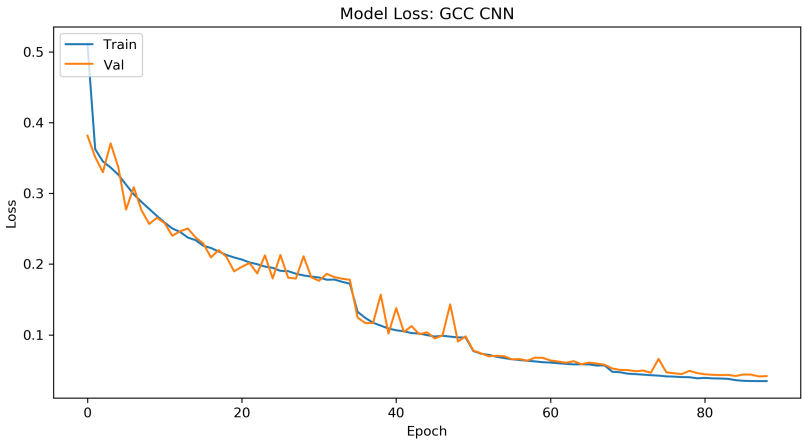

Figure 6.2: GCC CNN Training: Loss

6.1.2 Raw CNN

Raw CNN is a significantly larger model than *GCC CNN*. As expected, the training time was longer at approximately 5hrs. Training stopped at just more than 90 epochs. The training history is shown figures 6.3 and 6.4. The sharp improvements in training accuracy and training loss can similarly be explained by the use of `ReduceLROnPlateau()`. However, these improvements are not directly apparent in the validation accuracy and loss, which both display far more noise than their *GCC CNN* counterparts. The model achieved a validation accuracy of $\approx 98\%$.

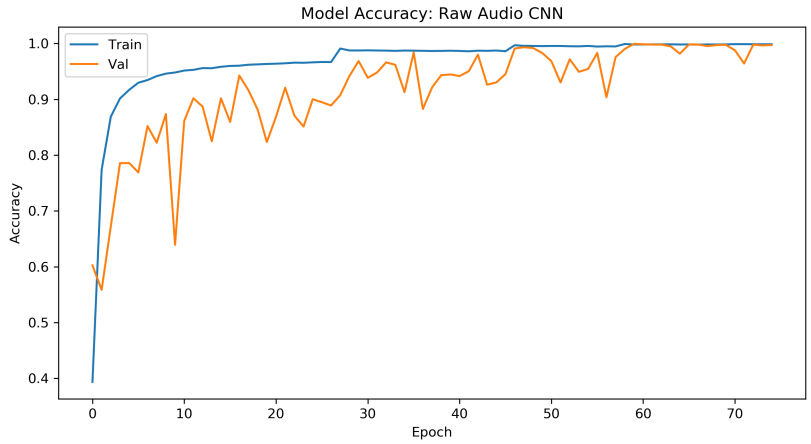

Figure 6.3: Raw CNN Training: Accuracy

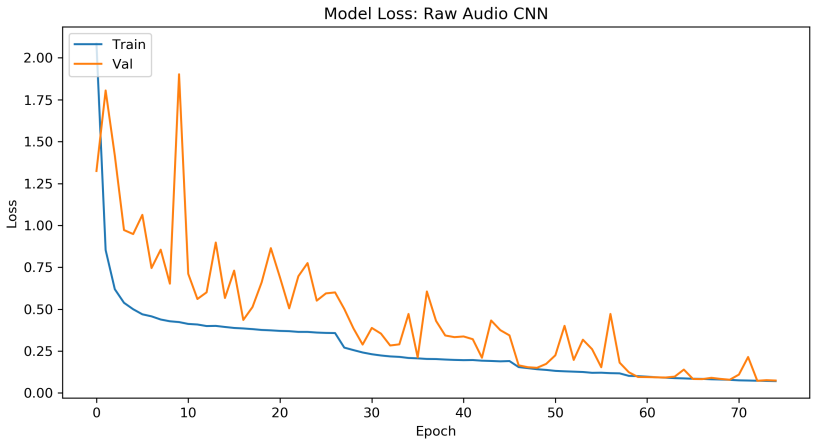

Figure 6.4: Raw CNN Training: loss

6.1.3 Raw ResNet

Raw ResNet took the longest time to train, at approximately 9hrs. This is to be expected due to the larger and more complex network architecture. Futhermore, compared to the other models, *Raw ResNet* trained for the greatest number of epochs, approximately 120, before the validation accuracy plateaued. The training history is shown figures 6.5 and 6.6. In comparison to *Raw CNN*, the validation loss and validation accuracy are both significantly less noisy, with the exception of a large spike around epoch 30. What is notable is that, after around epoch 70, the validation accuracy and loss converge closely to the training accuracy and loss. This could possibly be interpreted as an indication that the model is over-fitting. Further investigation into this will be performed in section 8 The model achieved a validation accuracy of $\approx 98\%$.

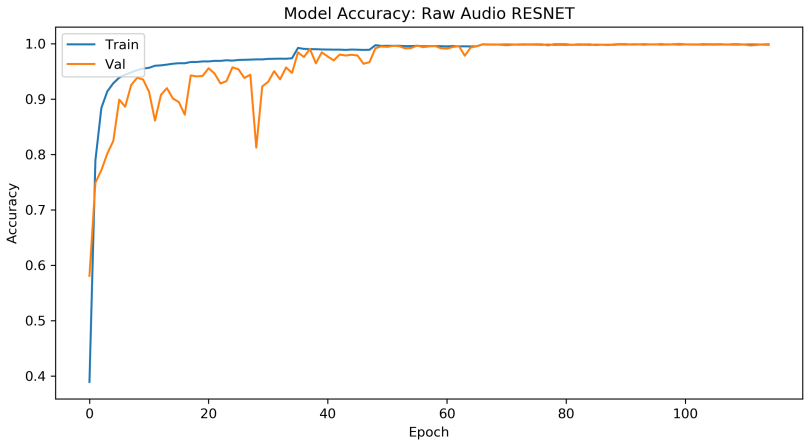

Figure 6.5: Raw ResNet Training: Accuracy

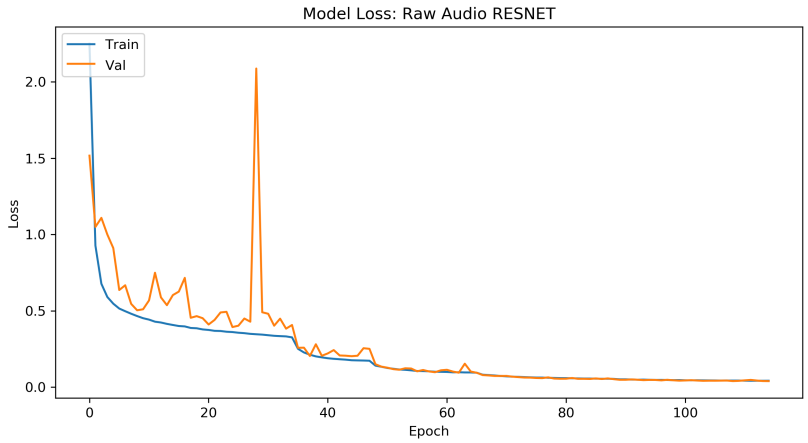

Figure 6.6: Raw ResNet Training: loss

6.2 Rotation, Models and Simulation

To integrate the models with the rotation algorithm, a simulation module `simulation.py` [1] was written. This module provided the interface shown in figure 6.7 with which simulations could be run. The results of each simulation were saved as an animation [2]. The high level architecture of the simulation program is shown in figure 6.8. This architecture represents the integration of the following subsystems:

[1] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/simulation.py`

[2] Available `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/README.md`

- Data Synthesis
- Data Preprocessing
- Deep Learning and DSP models
- Rotation Algorithm

This software architecture was designed to be entirely modular. For example, the DL models and the DSP model can be used interchangeably through implementation of the module `final_models.py` [3]. Moreover, the *Record, Load and Process Audio* block shown in figure 6.8 can be directly swapped with an different audio processing pipeline for implementation on the hardware. The same applies for the *Compute Rotation* blocks, which can be directly swapped for a module that controls the motor. Implementation of this algorithm on the hardware will be discussed in section 6.3.

```
if __name__ == '__main__':
    source_distance = 1  # source distance in meters
    source_azimuth = 10  # source azimuth in degrees

    # Instantiation of the simulation class
    sim = Simulation(directory="simulation_test",
                     source_azimuth=source_azimuth,
                     source_distance=source_distance,
                     model="raw_resnet")

    prediction = sim.simulate()  # Run simulation and get prediction

    print("Final Prediction: {prediction}".format(prediction=prediction))
```

Figure 6.7: Simulation Code Interface

A demonstration of this simulation is presented in figures 6.9 - 6.19. A simulation is run with two models, *GCC-PHAT TDOA* and *GCC CNN*. Both are present under the same operating conditions and both have a ground truth DOA of 45° shown in figure 6.9. Note that each figure represents a prediction-rotation pair, and shows the resting position of the apparatus after the prediction and corresponding rotation has occured. The first prediction-rotation iteration is shown in figure 6.12, where both models predict the correct hemifield of the source and rotate 90° to face it. The second prediction-rotation pair is shown in figure 6.15, where *GCC-PHAT TDOA* behaves as to be expected and *GCC CNN* makes two predictions far from the ground truth. The fourth iteration is shown in figure 6.18. Here, *GCC-PHAT TDOA* is shown correctly clustering a number of predictions around the true DOA. On the other hand, *GCC CNN* makes the same prediction twice and exits the rotation algorithm. The final prediction of *GCC CNN* is

[3] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/final_models.py`

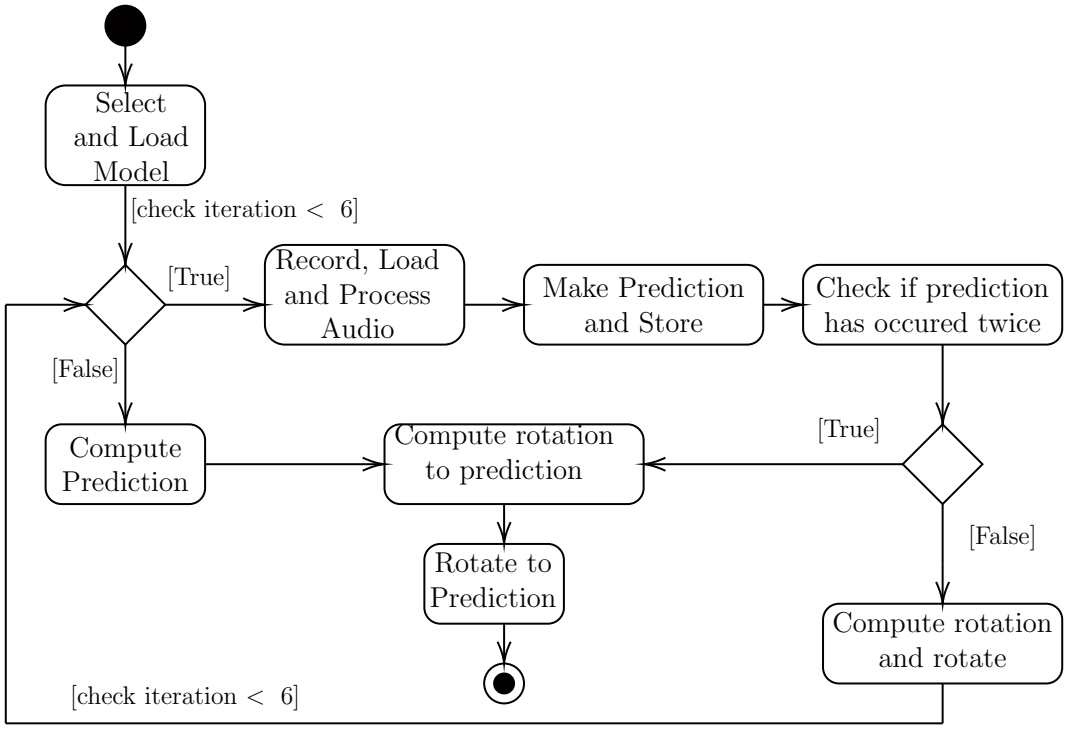

Figure 6.8: Simulation Architecture

45°. *GCC-PHAT TDOA* makes the same prediction twice in the fourth iteration shown in figure 6.19, with a final prediction of 50°.

In this example, both models performed DOA estimation succesfully, with a 0° error for *GCC CNN* and a 5° error for *GCC-PHAT DSP*. Deeper testing and analysis of the models will be performed in section 8.

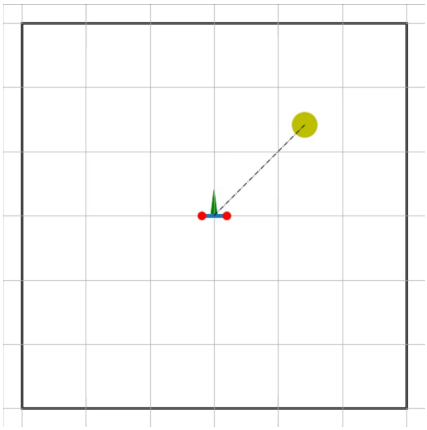

Figure 6.9: Simulation Starting Position: 45°

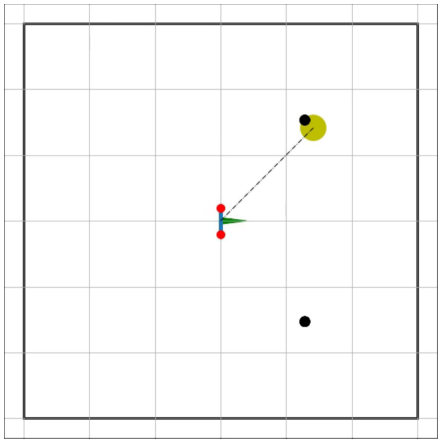

Figure 6.10: *GCC-PHAT TDOA*

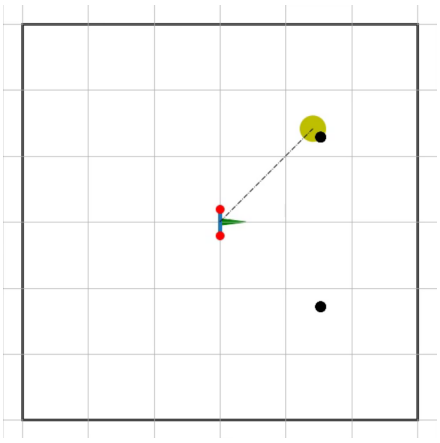

Figure 6.11: *GCC CNN*

Figure 6.12: Prediction-Rotation 1

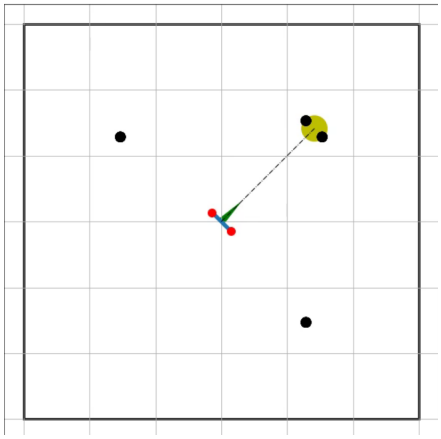

Figure 6.13: *GCC-PHAT TDOA*

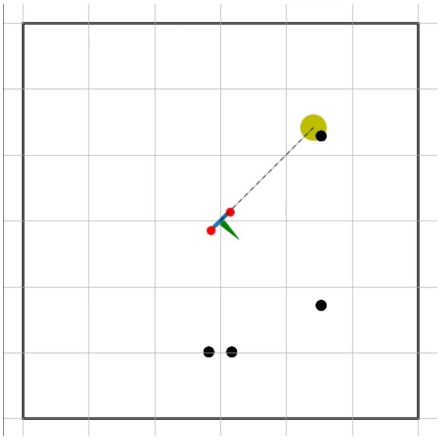

Figure 6.14: *GCC CNN*

Figure 6.15: Prediction-Rotation 2

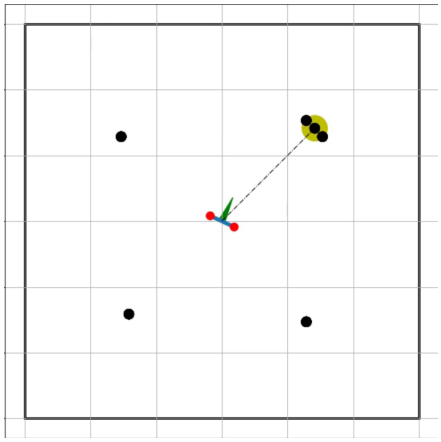

Figure 6.16: *GCC-PHAT TDOA*

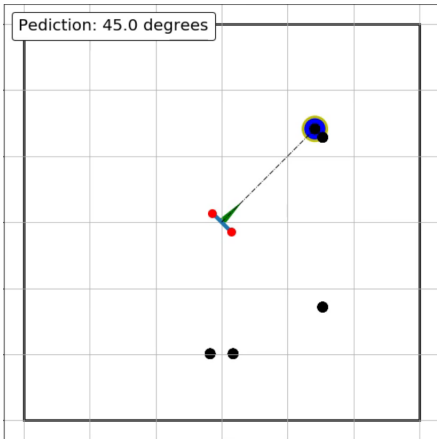

Figure 6.17: *GCC CNN* final prediction: 45°

Figure 6.18: Prediction-Rotation 3

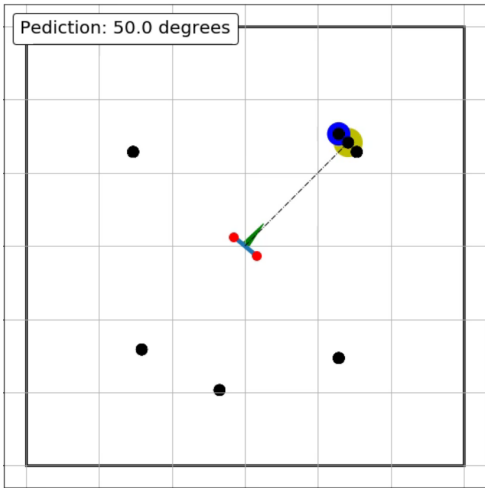

Figure 6.19: *GCC-PHAT TDOA* final prediction: 50°

6.3 Hardware

Subsystem integration of the hardware entailed the combination of the following subsystems

- Audio
- Motor
- Embedded Software

A full hardware block diagram is shown in figure 6.20. The circuit shown in the schematic in figure 5.30 was implemented on stripboard and is shown in figure 6.22. The final integration of the enclosure, audio subsystem and motor subsystem is shown in figure 6.21. The real-world implementation of the hardware block diagram is shown in figure 6.23. Note for the purpose of reducing clutter the various PSUs are not included in the image. The final hardware setup is shown in figure 6.24. A box was made and the system shown in figure 6.23 was placed inside. Heating issues where encountered with the Raspberry Pi, specifically during the deployment of the CNN Model. Thus, ventilation holes where placed strategically around the box.

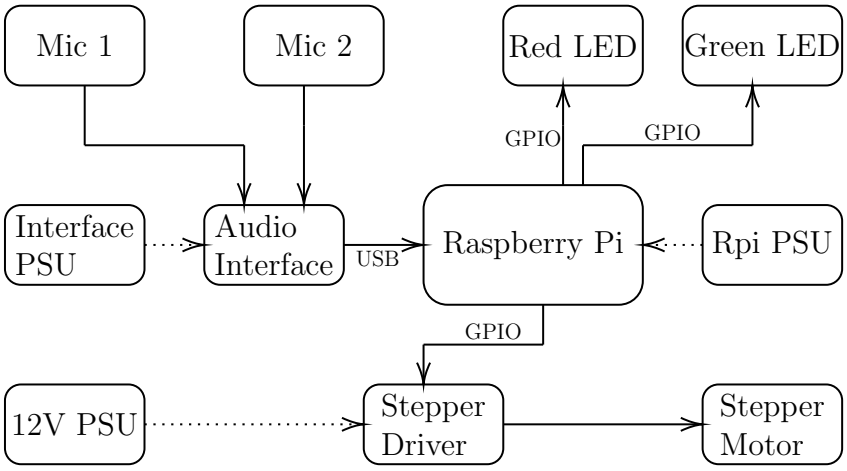

Figure 6.20: Hardware Block Diagram

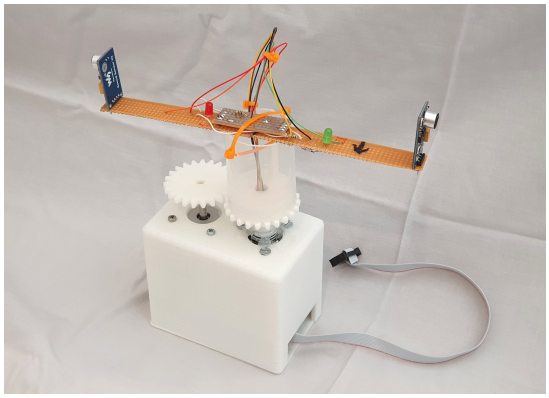

Figure 6.21: Hardware Integration Step 1

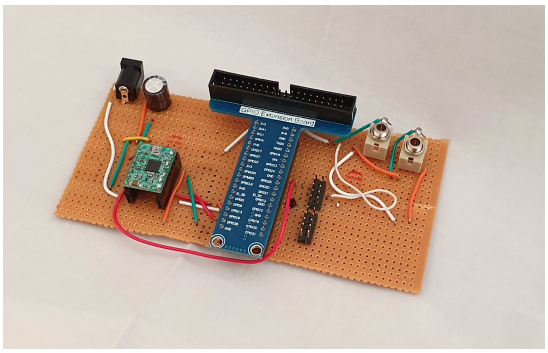

Figure 6.22: Stripboard Implementation of Circuit

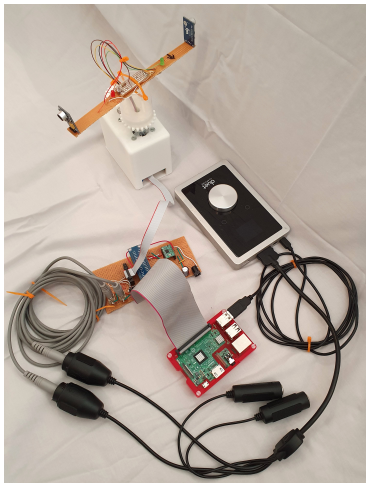

Figure 6.23: Hardware Integration Step 2

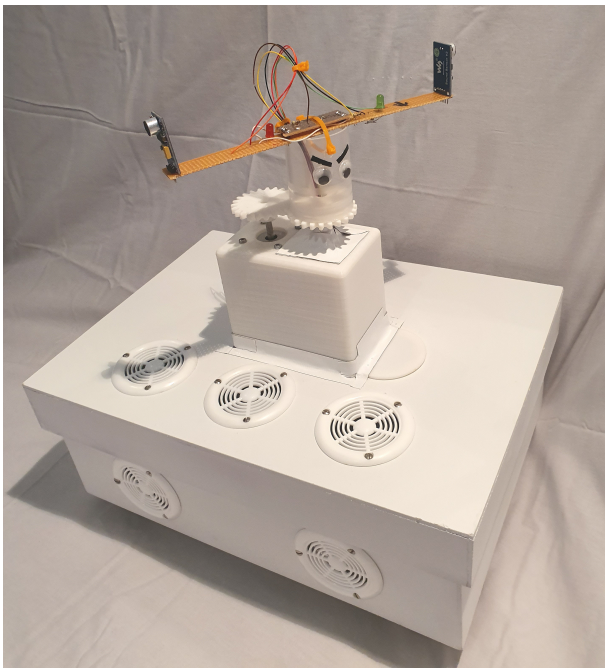

Figure 6.24: Full Hardware

6.4 Embedded Software

To implement the embedded software, the simulation module `simulation.py` described in section 6.2 was modified for deployment on the Raspberry Pi. As previously discussed, this entailed replacing the relevent blocks in figure 6.8 with modules appropriate for the hardware.

6.4.1 Audio

In the case of the audio pipeline, the hardware audio module architecture is shown in diagram 6.25. This was implemented in the python module `audio_recorder.py` [4]. The system was designed to record 3 seconds of audio on each iteration to ensure that some useful audio perturbations appeared in the recording. This was chosen as experimentation showed that a recording time fixed at 1 second (as is required by the models) was prone to catching a speaker in a moment of silence, and thus recording no useful information. However, due to the length of the audio, truncation was required for use with the models. Simply truncating the audio to a 1 second length defeats the purpose of recording a longer sample. Thus an onset detection and truncation algorithm was created. This was implemented using Librosa [**?**], a comprehensive python module for audio analysis. A

[4] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_hardware/audio_recorder.py`

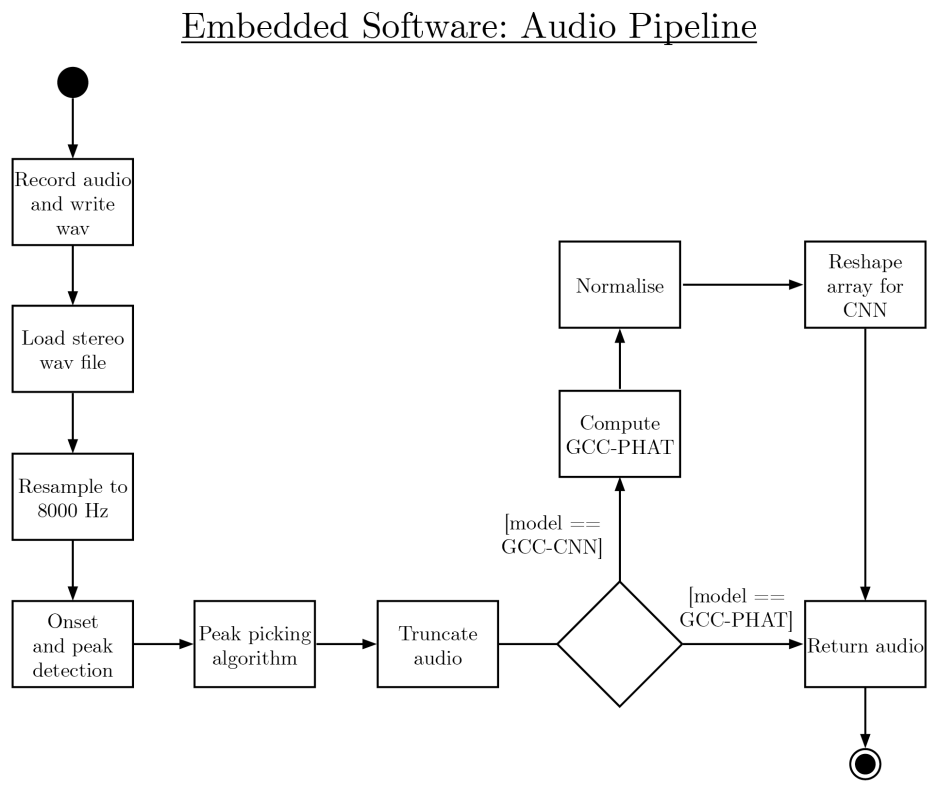

Figure 6.25: Audio Activity Diagram

block diagram of the audio truncation algorithm is shown in figure 6.26.

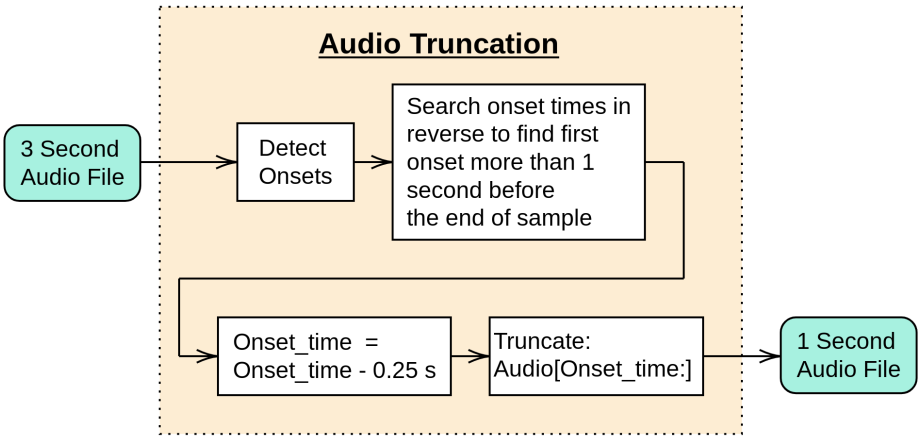

Figure 6.26: Audio Truncation Algorithm

Only a single channel of the stereo audio is used for onset detection, and both channels were truncated at the same time frames as to not loose pertinent phase information. This algorithm will be demonstrated using the 1.5 second waveform shown in figure 6.28. Onset detection is performed, the results of which are shown in figure 6.31. Peak picking (choosing the correct onset) and truncation are then performed as described in figure 6.26.

The final 1 second truncated waveform is shown in figure 6.30.

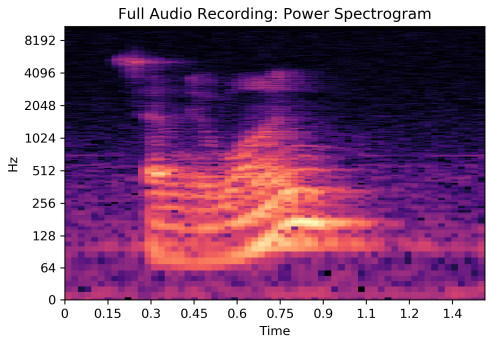

Figure 6.27: Power Spectrogram

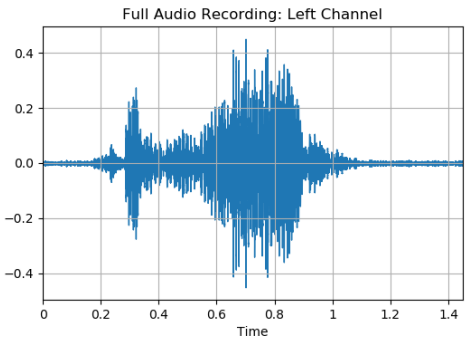

Figure 6.28: Full Waveform

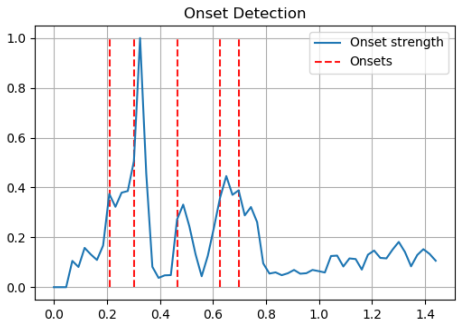

Figure 6.29: Onset Detection

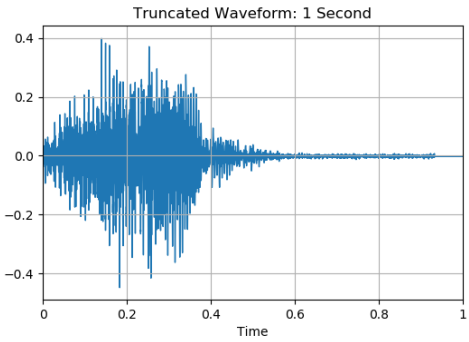

Figure 6.30: Truncated Waveform

Figure 6.31: Onset Detection Algorithm

6.4.2 Motor

The replacement of the simulation rotation algorithm with the motor rotation algorithm was relatively straight-forward. The implementation of this used the python module `motor.py`.

Chapter 7

Testing

This section will describe the design of various levels of tests. First subsystem testing will be described, then subsystem integration testing, full system evaluation and finally system verification. System verification will detail the process of determining if the acceptance tests described in section 3.5.2 have been passed.

7.1 Subsystem Testing

The subsystem tests described in this section are designed to verify that each subsystem identified in section 3.4 is functioning as specified.

7.1.1 Data Synthesis

The data synthesis subsystem is entirely software based. Qualitative verification testing of the synthesis results will be performed.

Verification

To verify that data synthesis was successful and that the data generated presented a clear spatial representation of the audio, three data points were generated and listened to with a pair of headphones to confirm the stereo image was accurate. Note that it is not

possible to provide a precise determination of the accuracy of the DOA from this test, as the test is purely based on human perception. However, it is possible to determine that the DOA is within the correct quadrant. These tests will also be used to verify that the generation of noise in the data is audible. Finally, the tests will show if changing room dimension and thus increasing reverberation is represented in the audio. To perform these verification tests, the following three audio files will be synthesised:

- An arbitrary sound source at a 45° DOA
- The same recording with low SNR
- The same recording with large room dimension

If each of these recordings qualitatively represent their specifications, the test is passed.

7.1.2 Data Preprocessing

The purpose of testing the data preprocessing subsystem is to ensure that once the data has been reshaped, normalised and downsampled it is still representitive of the unprocessed data. That is to say that no errors have occurred during processing that would damage the data such that an accurate representation of spatial audio is present. This is once again a qualitative test based on perception. This test also exists to ensure that the labelling of the data has remained intact through preprocessing. This is particularly important due to the asynchronous nature of the parallel implementation of the data preprocessing

Verification

The process of this verification test is to preprocess some existing data, and then extract a single data point and its label from the data-structure. This data point would be listened to in order to verify that the data has remained in tact and the DOA label is correct.

7.1.3 Classical Model Software Implementation

Testing the classical model will involve a set of unit tests of the module `gccphat.py` . These tests are shown the file `test_gccphat.py` [1]. The purpose of these tests are not to evaluate the performance of the model, but rather to ensure that the code is functioning as expected.

7.1.4 Hardware

Testing the hardware subsystems verifies there are no defects in their implementation as to introduce error into later experiments. In the case of the audio subsystem, this test exists to evaluate the signal to noise ratio of the recording. In the case of the motor the purpose of the test is to ensure that the real world rotation performed corresponds to the rotation specified in software.

Audio Subsystem Tests

The audio subsystem will be configured and tested with a recording. If there is no audible noise or clipping of the signal, the test has been passed.

Motor Subsystem Tests

Specify a rotation in software, if the motor rotation corresponds to the requested rotation, the test is passed.

7.2 Subsystem Integration and Evaluation

Unlike the previous section, as opposed to merely verifying that the subsystems are functioning as expected, the tests in this section serve to evaluate the performance of the integrated subsystems.

[1] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/tests/test_gccphat.py`

7.2.1 Deep Learning Model Evaluation on Simulated Audio

Corresponding to the subsystem integration described in section 6.1, the tests in this subsection serve to evaluate the performance of the deep learning models on simulated audio datasets that vary from the training dataset. The datasets to be used have been described in section 5.1.3. Testing the models on the *Unseen Blind Evaluation* dataset, referred to here as *unseen*, serves to act as a measure of generalisation, as the dataset is constructed using fixed variables (room dimension, snr, reverb time) that are not present in the training dataset. A second evaluation dataset *Seen Blind Evaluation*, referred to here as *seen*, will be used to determine whether the models have developed a useful model of DOA estimation. This dataset is constructed from room conditions that have been seen in the training data, but at a different source distance. The following tests will be performed for each model.

- Model accuracy for seen and unseen datasets
- Confusion matrix on the following datasets:
  - Validation Set (from the training dataset)
  - Seen Dataset
  - Unseen Dataset

7.2.2 Rotation Model Evaluation on Simulated Audio

An evaluation program `evaluate.py` [2] was written to evaluate the performance of all the models, when deployed in conjunction with the rotation algorithm and tested on simulated data. This program uses the simulation class `simulation.py` [3]. The evaluation program runs a loop of simulations, verifying a specified degree of freedom such as source distance or DOA. Each model is evaluated under a number of conditions. The results will be presented as a confusion matrix and an *errorcircle* (to be defined in section 8.2.1). The following simulated conditions will be tested, each corresponding to an evaluation loop for each model. Each evaluation loop was be run 10 times in order to determine an average error.

[2] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/evaluate.py`

[3] Available: `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/localisation_software/simulation.py`

- Seen training conditions, with the source 1 meter from the recording apparatus
- Seen training conditions, varying distance, all else fixed.
- Low SNR stress test
- High reverb stress test

7.2.3 Model Predictions and Hardware Rotations

The purpose of this test is to evaluate if the final rotation of the recording apparatus corresponds to the final prediction with reference to the starting position. This is a simple test that will ensure that if, for example, a prediction of 10° is made, at the end of rotation the recording apparatus faces 10° with respect to the initial position. A hardware prediction will be run and the corresponding DOA prediction will be compared to the final position w.r.t the origin.

7.3 Full System Evaluation

This set of tests serves to evaluate the fully integrated hardware-software system. Like the tests in the previous section, here all degrees of freedom were held constant barring the one under test. In each test, a deep learning model will be evaluated against the DSP model *GCC-PHAT TDOA* as a performance baseline. Only the DL model that showed the best performance in the simulation test will be evaluated on the hardware. Thus each hardware test will be run twice, once for the DSP model and once for the DL model. The experimental apparatus is shown in figure 7.1. For each test, the loudspeaker was kept at a fixed volume. For all tests barring the distance test, the hardware system was positioned according to the specified recording conditions, and the DOA was incrementally changed for each prediction session. To change the DOA, the starting position of the recording apparatus was rotated instead of rotating the loudspeaker position. To ensure accurate results, the rotations were performed using software. For example, instead of manually rotating the apparatus to 5°, a 5° rotation instruction was given through the software CLI. The final prediction made by the system was then printed to the SSH terminal. Each test will now be described in it's own subsection.

Figure 7.1: Experimental Apparatus

7.3.1 Seen Room Conditions

This test places the experimental apparatus in the room conditions upon which the training dataset was based. This is a room of dimension $3 \times 3 \times 2.5$ metres. The apparatus was positioned in the centre of the room, with the loudspeaker 1 metre from the system. This testing configuration is shown in figure 7.2. A long recording of speech (a podcast) was used as the testing audio. Each DOA was tested in a 5 degree resolution. The purpose of this test was to evaluate how well the models performed when deployed on the hardware, given data of the same dimensions as the training set.

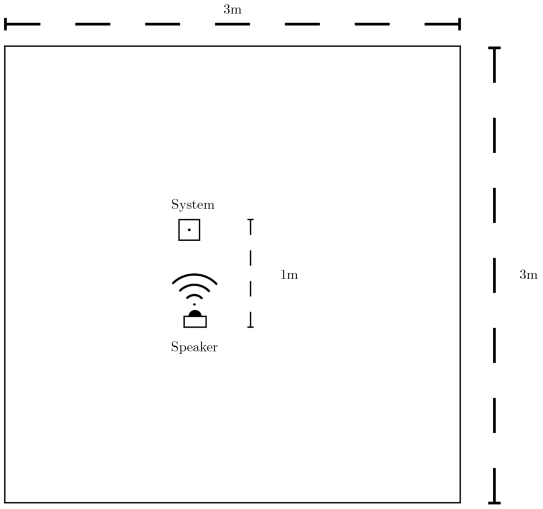

Figure 7.2: Seen Hardware Testing: Experimental Configuration

7.3.2 Unseen Audio Conditions

This test evaluates the performance of the system on audio sources unseen in training. The tests keeps all the same conditions as the *Seen Room Conditions* test, but instead of using speech as the testing audio, music was used. The purpose of this test is evaluation of the model's robusteness to changes in audio source type. DOAs were tested with a 10 degree resolution. The experimental configuration was the same as seen in figure 7.2.

7.3.3 Unseen Room Conditions

The purpose of this test is to evaluate the performance of the system in conditions not seen in the training dataset. This entails a different room shape and a source distance larger than the maximum distance in the training set. Once again, a podcast was used as the testing audio. DOAs were tested with a 20 degree resolution. This testing configuration is shown in figure 7.2.

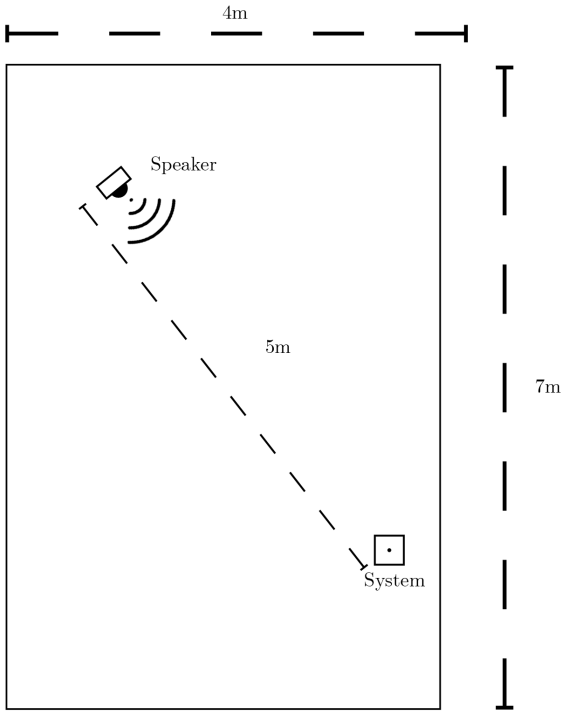

Figure 7.3: Unseen Hardware Testing: Experimental Configuration

7.3.4 Cocktail Party Test

In order to evaluate the system's robustness to noise, a two noise sources and a sound source were placed in seen room conditions. The room dimensions were the same as specified in the training dataset. The DOAs were tested with a 20 degree resolution. The noise sources were a cellphone and a laptop, both playing a recording of coffee shop ambient noise. This recording includes voices, footsteps, sounds of cutlery hitting plates and background music. The sound source used was the same podcast used in the previous sections.

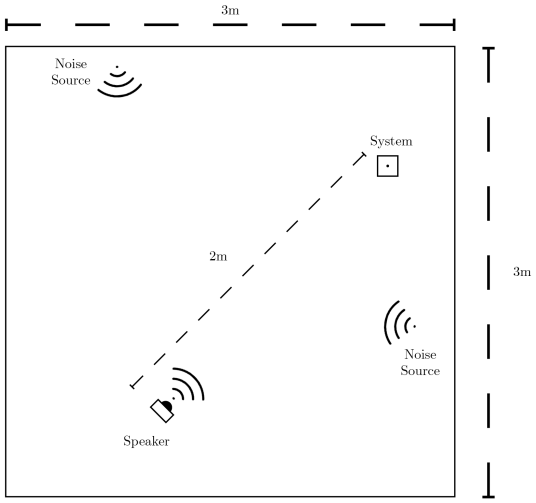

Figure 7.4: Cocktail Party Hardware Testing: Experimental Configuration

7.3.5 Hardware Distance Test

This test serves to evaluate the performance of the system with changes in source distance from the recording apparatus. The dimension of the room used was the same as seen in figure 7.3. The apparatus was placed in the center of the room and fixed at a rotation of 45°. The same podcast was used as the sound source. The loudspeaker was moved away from the apparatus in 0.5 metre increments, starting at 1 metre and ending at 3.5 metres.

Acceptance Test Procedure	Corresponding Evaluation Test	Section
AT1	Hardware Seen	7.3.1
AT2	Hardware Seen	7.3.1
AT3	Hardware Cocktail Party	7.3.4
AT4	Simulation High Reverb	7.2.2
AT5	Model Predictions and Hardware Rotations	7.2.3
AT6	Hardware Unseen	7.3.3

Table 7.1: Acceptance Tests and Corresponding Evaluation Tests

7.4 System Verification

The purpose of this section is to link the acceptance test procedures described in section 3.5.2 with the tests described in the preceding sections of this chapter. The ATPs and their corresponding evaluation tests are shown in table 7.1.

Chapter 8

Results

8.1 Subsystem Testing

The results of the Data Synthesis [1], Data Preprocessing and Classical Model Software Implementation tests are shown in the table 8.1.

Subsystem	**Test Results**
Data Synthesis	All verification tests were passed
Data Preprocessing	The verification tests were passed
Classical Model Software Implementation	All unit tests were passed

Table 8.1: Subsystem Testing Results

8.1.1 Deep Learning Model Evaluation on Simulated Audio

The results in this section correspond to the tests described in section 7.2.1.

Deep Learning Models Accuracy Test

Figure 8.1 shows the raw accuracy of the three DL models evaluated on two datasets distinct from the training set. *Raw CNN* and *Raw Resnet* perform poorly on both dataset, even though a high validation accuracy was achieved in training. This could be

[1]Example Audio of the results of the test can be heard here `https://github.com/murning/EEE4022S_Final_Year_Project/blob/master/README.md`

evidence that a useful feature representation has not been extracted by these models, and that they have been subject to over-fitting to the training data.

In the case of *GCC CNN*, the accuracy for the seen dataset is comparable to the validation accuracy in training. The accuracy degrades with a change to an unseen environment, but this accuracy is still significantly higher than that of the other two models. Thus we can conclude that *GCC CNN* performs the best out of the three models and displays the best ability to generalize to new environments.

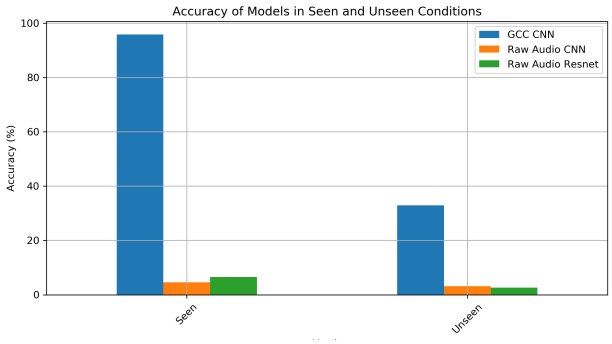

Figure 8.1: Model Accuracy Plot

Raw Audio ResNet

Figure 8.5 shows confusion matrices of *Raw ResNet* for the three datasets used in evaluation. Upon inspection of these matrices, it is clear that the model has over-fit to the training data. In comparison to the validation set confusion matrix, the matrices of the *seen* and *unseen* show that the model does not generalise well to new environments.

Raw Audio CNN

The confusion matrices of *Raw CNN* are shown in figure 8.9. The same conclusions drawn in the case of *Raw ResNet* can be made here.

GCC CNN

The results of this test are consistent with the *accuracy* results shown in figure 8.1. The model accuracy in the validation set and the *seen* dataset are equitable. A degradation

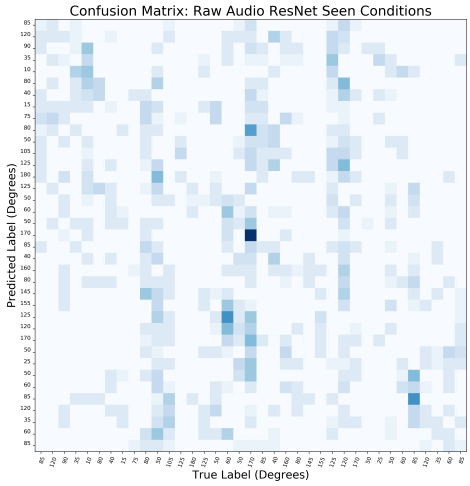

Figure 8.2: Seen Dataset

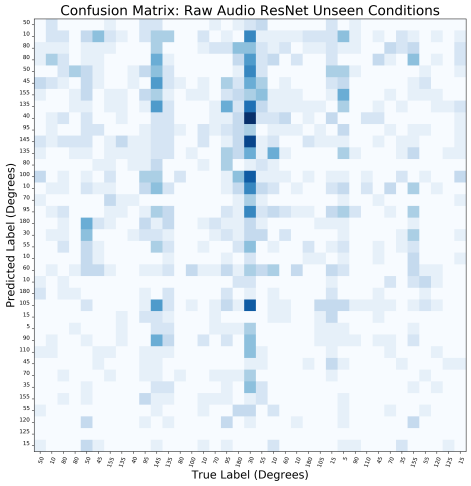

Figure 8.3: Unseen Dataset

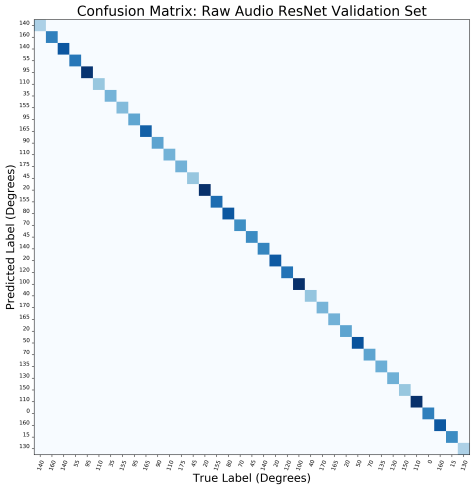

Figure 8.4: Validation Set

Figure 8.5: *Raw ResNet* Confusion Matrices

in accuracy is seen in the confusion matrix of the *unseen* dataset. From these results we can conclude that *GCC CNN* vastly outperforms the other DL models.

8.1.2 Hardware

Audio Subsystem Tests

A number of recordings with audio sources at different volume levels. Clipping of the signal was observed in one test, and thus the amplifier gain was adjusted appropriately. Audible noise was initially present, but this was rectified by changing the power supply. Following these adjustments, the test was passed.

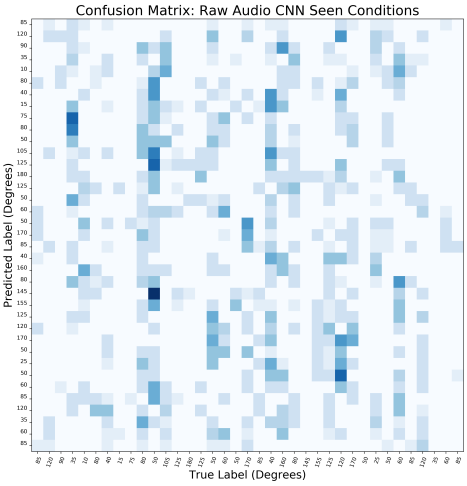

Figure 8.6: Seen Dataset

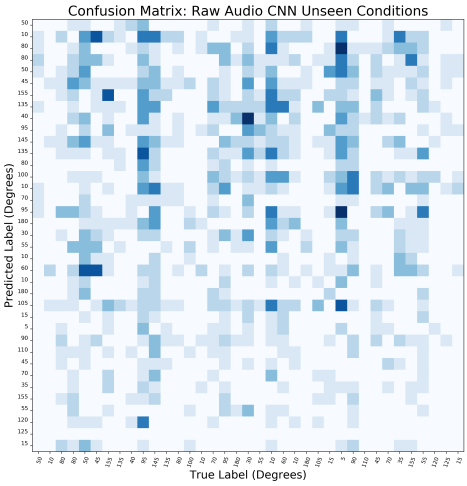

Figure 8.7: Unseen Dataset

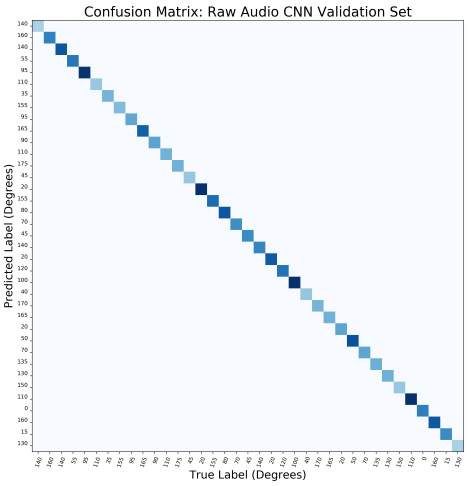

Figure 8.8: Validation Set

Figure 8.9: *Raw CNN* Confusion Matrices

Motor Subsystem Tests

The motor subsystem test was passed. The correct physical rotation was performed given a specified software rotation.

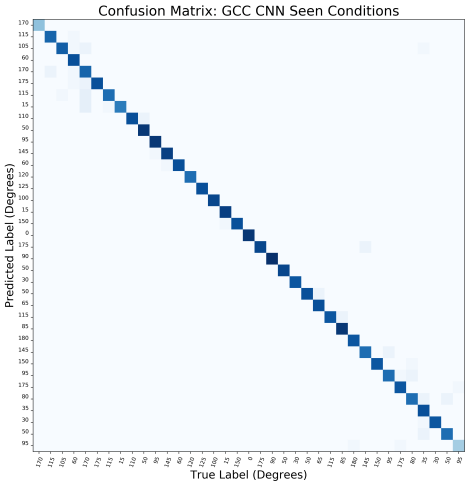

Figure 8.10: Seen Dataset

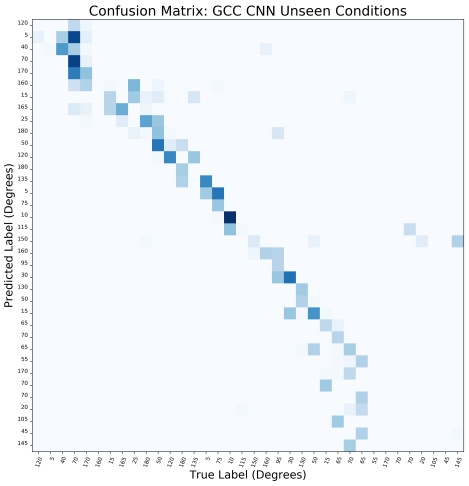

Figure 8.11: Unseen Dataset

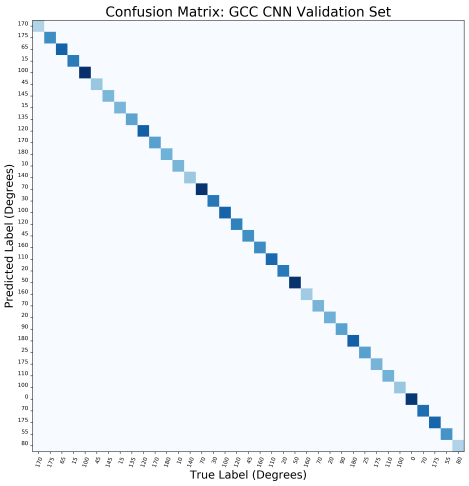

Figure 8.12: Validation Set

Figure 8.13: gcccnn net bare model conmat

8.2 Subsystem Integration Testing

8.2.1 Rotation and Models Evaluation on Simulated Audio

The results in presented in this section correspond with the tests defined in section 7.2.2. This section is the first point in the results that makes use of the error circle for visualisation. The error circle is a way of visualising the *average error* of a model in the azimuth plane. The magnitude of the average error for each DOA is represented by it's distance from the origin. For example in the error circle of *GCC CNN* in figure 8.23, the purple dot represents that at a DOA of $\approx 280°$, the average error of predictions was $10°$. The red region of the error circle shows the region of error deemed acceptable by the technical specifications. This is a region in which the average prediction is within the correct

quadrant. For example, as shown in figure 8.23, all of the predictions of *GCC-PHAT TDOA* are within the acceptable region, whereas in the case of *Raw CNN*, the majority of predictions are out of the acceptable region.

Seen Conditions: 1 Metre Source Distance

The results of this test are shown as a confusion matrix in figure 8.18 and an *error circle* in figure 8.23. Even with the inclusion of the rotation algorithm, accuracy of both *Raw CNN* and *Raw ResNet* are not within acceptable limits. On the other hand, *GCC CNN* and *GCC-PHAT TDOA* are both well within the acceptable error limits as shown in figure 8.23. One interesting observation from the results of this experiment is that with the inclusion of the rotation algorithm, the confusion matrices of *Raw CNN* and *Raw ResNet* start to display a type of vertical pattern that is not apparent in the test without the rotation model.

Distance test simulation

Figure 8.24 shows the results of the distance error test described in section 7.2.2. Given the results up to this point it is not surprising that performance of *Raw CNN* and *Raw ResNet* is poor and that no apparent pattern is evident in the results. On the other hand, what is interesting in these results is that the average error of *GCC CNN* and *GCC-PHAT TDOA* increases the closer the sound source gets to the apparatus. This result makes sense, as the distance between the two microphones is fixed at $0.2m$. The shortest distance in this experiment is only $0.1m$ from the apparatus, and thus the propagation delay observed at the microphones may be harder to discern.

Simulated Low SNR test

The results of the Low SNR test are shown in figure 8.29. Compared to the results from the seen conditions test, both *GCC CNN* and *GCC-PHAT TDOA* display a degradation in performance. *GCC-PHAT TDOA* remains relatively robust to noise, with no predictions falling out of acceptable limits. In the case of *GCC CNN*, the average error of a number of predictions fall beyond the acceptable error limits. However, the average error of the majority of the DOAs remains within the prescribed limits. Compared to the seen test, no discernible difference in the performance of the other two DL models can be observed.

Figure 8.14: *GCC-PHAT TDOA*

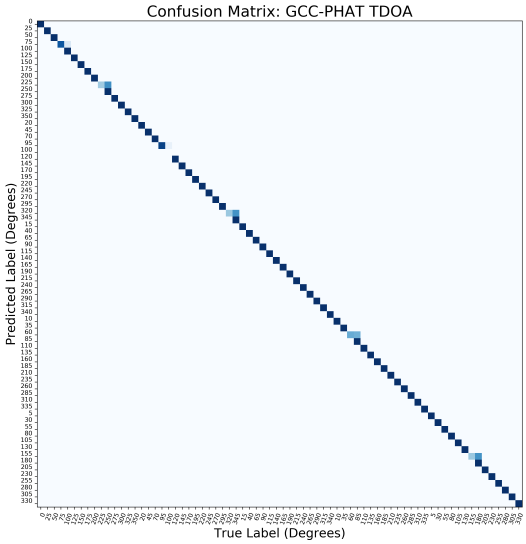

Figure 8.15: *GCC CNN*

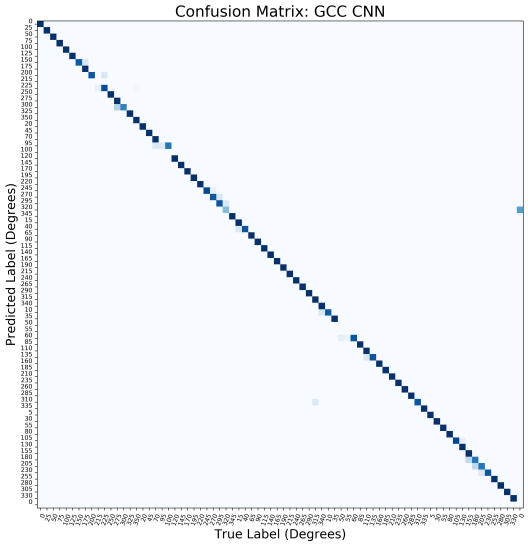

Figure 8.16: *Raw CNN*

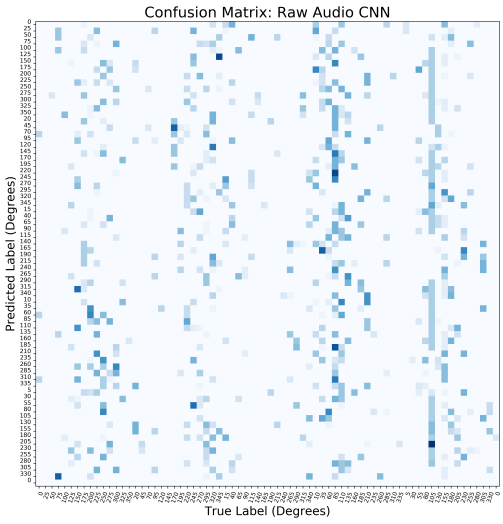

Figure 8.17: *Raw ResNet*

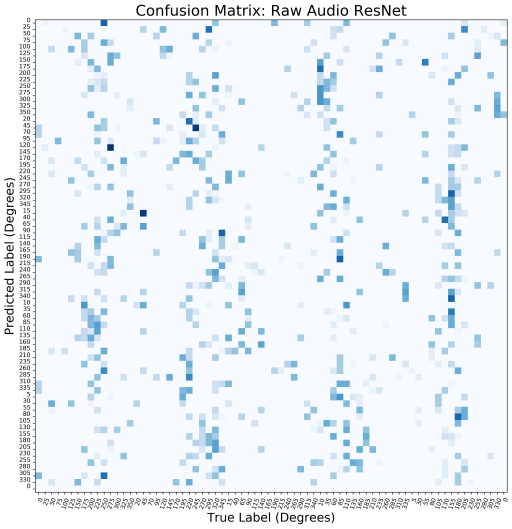

Figure 8.18: Seen Training Conditions, 1 Metre Distance Confusion Matrix

We can conclude from this experiment that *GCC-PHAT TDOA* is the most robust to noise.

Simulated High Reverb Test

Considering the poor performance of *Raw ResNet* and *Raw CNN* observed thus far, only the other two models were considered in this experiment, the results of which can be seen in figure 8.30. These results are the most interesting of all observed thus far, as a clear pattern in the average error appears to emerge in an experiment with extremely adverse reverberant conditions. The code was carefully examined and the experiment was run

Figure 8.19: *GCC-PHAT TDOA*

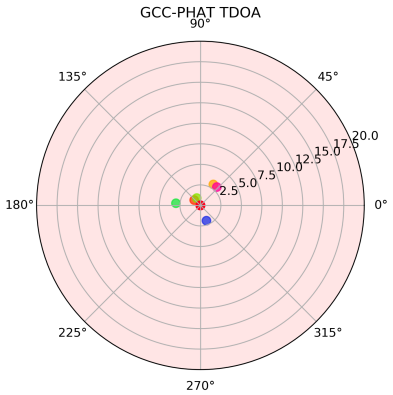

Figure 8.20: *GCC CNN*

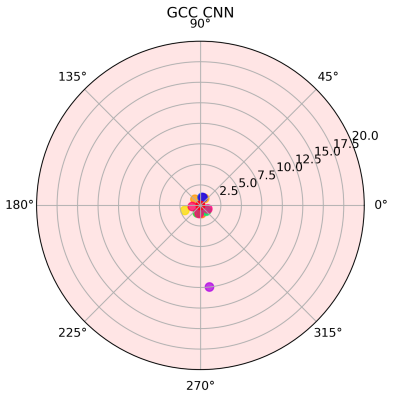

Figure 8.21: *Raw CNN*

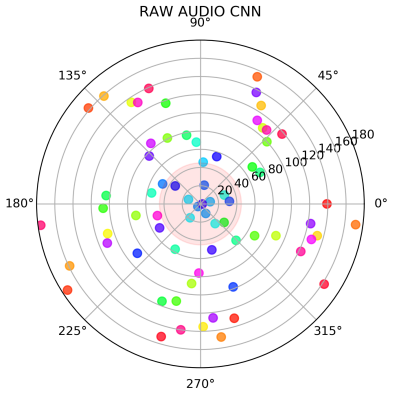

Figure 8.22: *Raw ResNet*

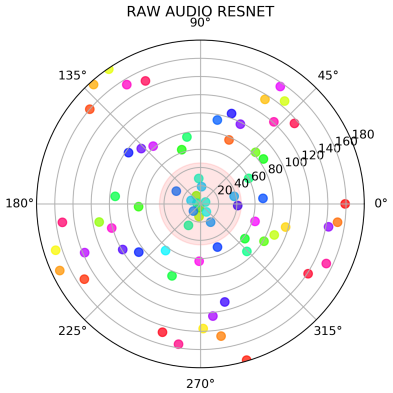

Figure 8.23: Seen Training Conditions, 1 Metre Distance Error Circle

multiple times and the pattern continued to emerge. Further research will be required to interpret these results. However, we can confirm that both models undergo severe degradation in performance under these conditions.

8.2.2 Model Predictions and Hardware Rotations

To perform this test, multiple predictions were run and the final position consistently matched the DOA prediction. This was measured visually using a protractor. The degree of uncertainty associated with this measurement is not significant due to the fact that the prediction models only have a 5° resolution.

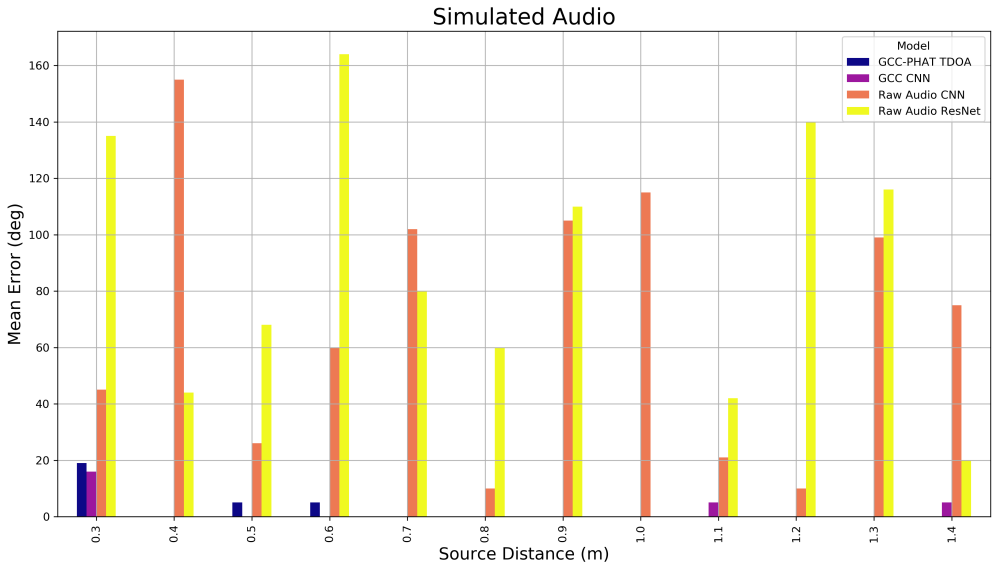

Figure 8.24: Simulated audio distance test

8.3 Full System Evaluation: Results

The results in this section correspond to the tests outlined in section 7.3. These tests evaluate the performance of the full system in real world environments. Note that for these experiments, only *GCC CNN* and *GCC-PHAT TDOA* were evaluated.

8.3.1 Seen Room Conditions: Results

The results of the test described in section 7.3.1 can be seen in figure 8.31. It is clear from the error circle that both models meet the maximum error specifications. The error in *GCC-PHAT TDOA* is consistently less than 20°, whereas in the case of *GCC CNN*, the error varies to a greater degree. It can be concluded from these results that both models meet the project specifications when evaluated under controlled, seen conditions.

8.3.2 Unseen Room Conditions: Results

The results of the test described in section 7.3.3 are shown in figure 8.32. Interestingly, both models undergo degradation under these conditions. It is expected that *GCC CNN* may undergo degredation under unseen conditions. However, *GCC-PHAT TDOA* should

Figure 8.25: GCC-PHAT TDOA

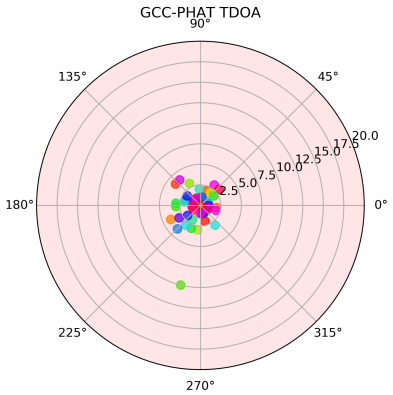

Figure 8.26: GCC CNN

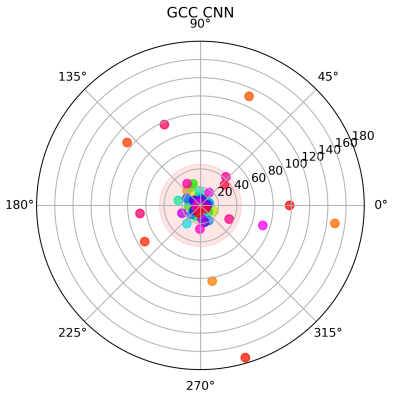

Figure 8.27: Raw CNN

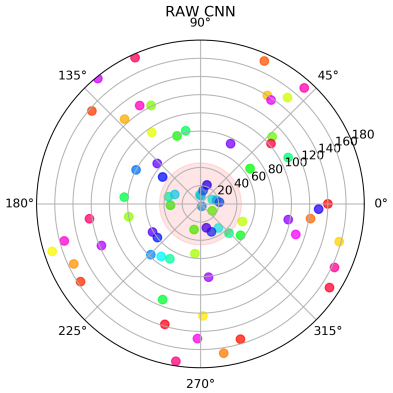

Figure 8.28: Raw ResNet

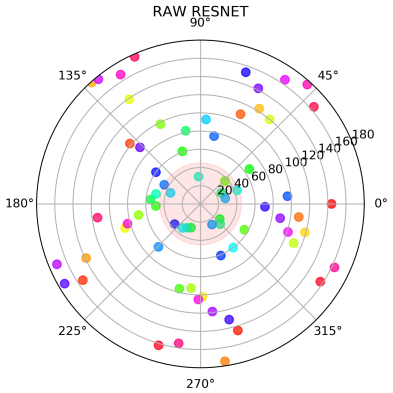

Figure 8.29: Low SNR Test

be impervious to the changes in conditions observed in this experiment, as it's model is not reliant on training data. Thus the fact that degredation is observed in both models suggests that the conditions were adverse to start with, and the results of this test cannot be used as a robust indication of the generalisation of *GCC CNN* to new environments. This being said, although errors do occur outside the acceptable range, their frequency is not sufficient to claim the models are no longer useful.

8.3.3 Seen Room Conditions: Music

The results shown in figure 8.33 correspond to the test described in section 7.3.2. Using the results for the seen speech test (figure 8.31) as a baseline, it is clear that both models do not suffer severe degredation in performance. Some high errors are evident in both models, but these are so few that they can be considered statistically insignificant. Thus we can conclude from these results that *GCC CNN* is not significantly sensitive to type

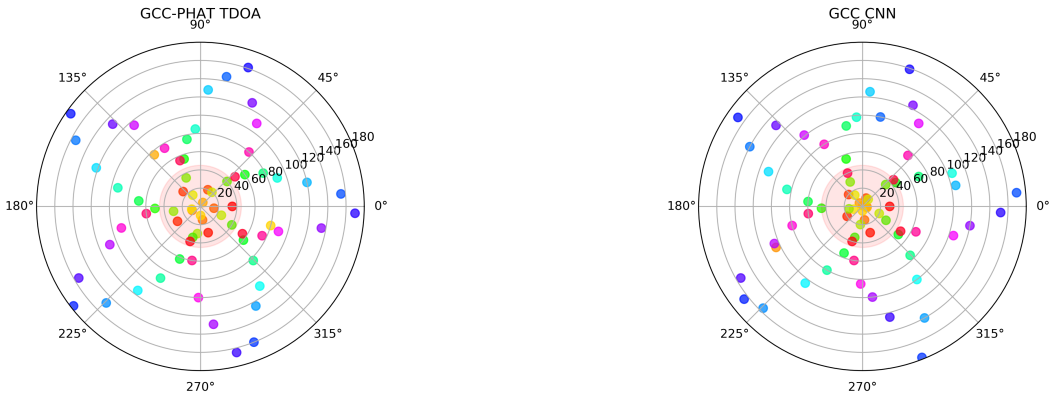

Figure 8.30: 1m seen Reverb error circle

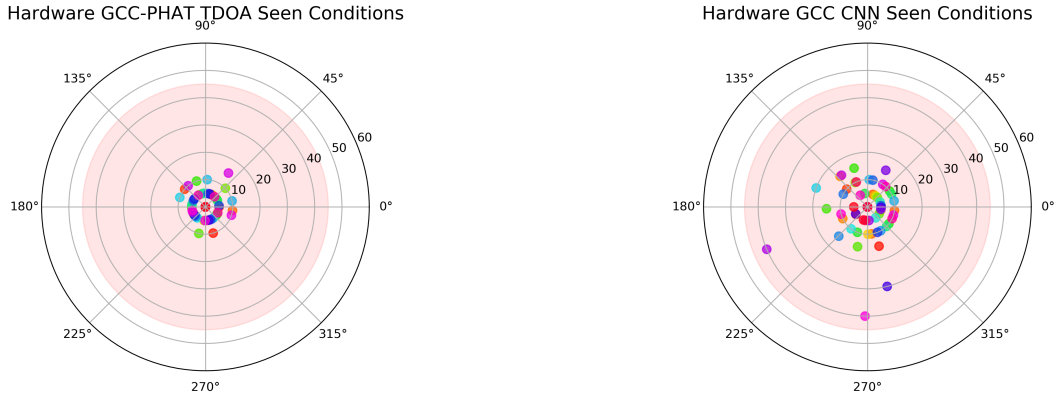

Figure 8.31: Seen Room Conditions Error Circle

of audio input, even though only speech was used for training.

8.3.4 Cocktail Party Test: Results

The results of the test described in section 7.3.4 are shown in figure 8.34. When compared to the results of the seen room conditions test, under these testing conditions, a minor degredation in performance is observed for *GCC-PHAT TDOA* and a relatively significant degredation is observed for *GCC CNN*. This result is consistent with the findings of the low SNR test in simulation (section 8.2.1), in which *GCC-PHAT TDOA* remained relatively unchanged and *GCC CNN* took a severe performance hit. Thus the conclusion reached in the case of the low SNR simulation has been confirmed by this experiment. *GCC CNN* is susceptible to noise interference but *GCC-PHAT TDOA* is not.

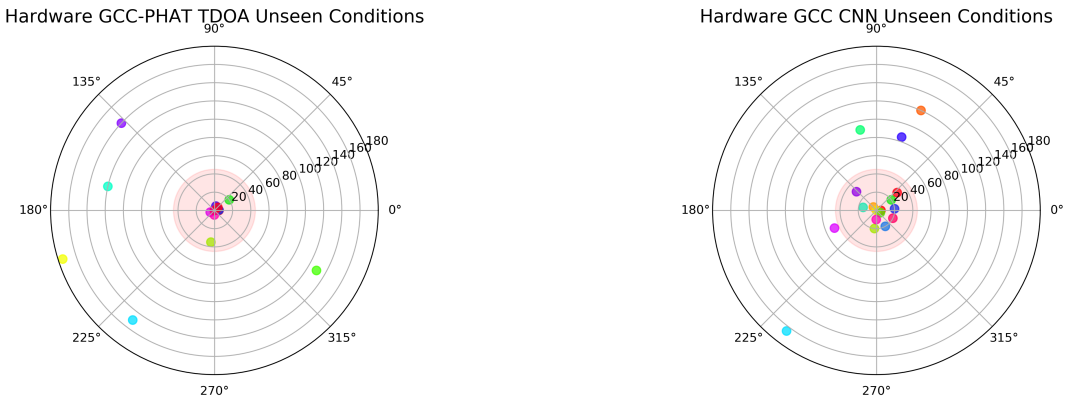

Figure 8.32: Unseen Room Conditions Error Circle

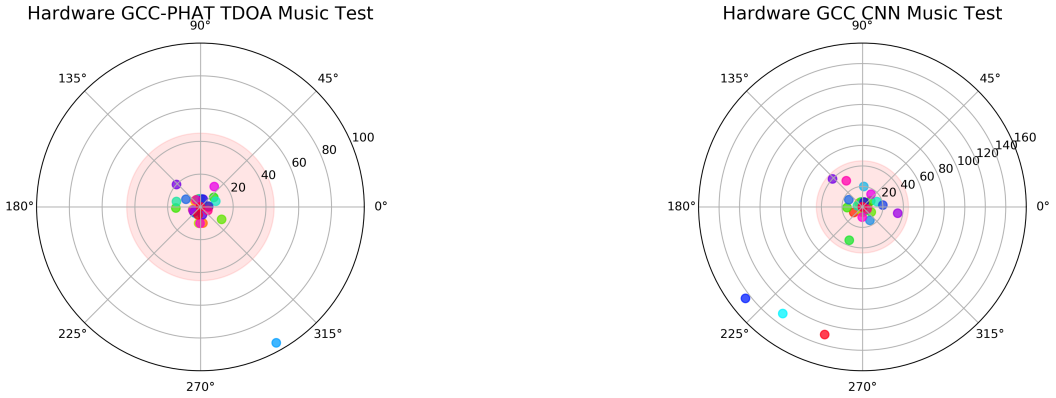

Figure 8.33: Seen Room Conditions: Music

8.3.5 Hardware Distance Test: Results

The results of the hardware distance test are shown in figure 8.35. A phenomenon not evident in the simulation distance test (figure 8.24), the performance of *GCC-PHAT TDOA* appears to degrade with distance, whereas the accuracy of *GCC CNN* remains relatively unchanged. This is an interesting result, however, before any conclusions can be made more experimentation needs to be performed. The distance error was only evaluated for one DOA, and in a single room configuration. So although these results seem promising for *GCC CNN*, further validation is required before any conclusions can be made.

Figure 8.34: Hardware Cocktail Party Results

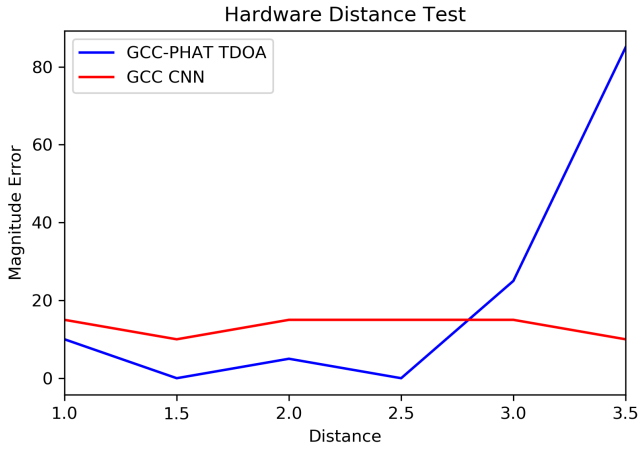

Figure 8.35: Hardware Distance Test

8.4 System Verification

The results of system verification are summarized in table 8.2. AT1 was passed, as the Hardware seen conditions test presented the error within the correct quadrant for each DOA. AT2 was failed, As some front back confusions did occur. Note that a front-back confusion can occur whilst the error falls in a 90° range, thus allowing AT1 to be passed and AT2 to be failed. AT3 and AT4 were both failed, as a robustness to reverberation and noise was not demonstrated. AT5 was passed, as the final rotation position corresponded to the DOA prediction for each experiment. Finally, AT5 was failed, as the system did not adapt successfully to unseen environments.

Acceptance Test Procedure	Result	Results Section
AT1	Passed	8.31
AT2	Failed	8.31
AT3	Failed	8.34
AT4	Failed	8.30
AT5	Passed	8.2.2
AT6	Failed	8.31

Table 8.2: Acceptance Test Results

Chapter 9

Discussion and Conclusions

This section will first present a discussion of the results obtained, and how they pertain to the literature discussed in Chapter 2. Following this the successes and failures of the project will be discussed with reference to the project specifications outlined in Chapter 3. Finally, some general concluding remarks will be made. To add context to the discussions presented in this section, it is necessary to reiterate the project's research question:

> In comparison to using classical signal processing techniques, can deep learning be used in the task of binaural single source sound localisation in order to develop a system that is robust to noise and reverberation? Furthermore, can the deployment of this system on a rotating robotic platform improve localisation accuracy?

In a response to this question, a brief summary of work in this project is presented as follows:

- A data-synthesis method was designed and implemented to generate large datasets to be used for sound source localisation studies.
- Three deep learning models where trained and tested using these datasets.
- A signal processing method was implemented to act as a baseline with which the DL models could be compared.
- A rotation algorithm was concieved of to account for problems inherent in in 360° binaural SSL.

- A simulation program was written to evaluate the aforementioned methods in software.
- A hardware system was designed and built with which the models were deployed.
- Comprehensive testing was performed in order to evaluate the system.

Discussion of Results

Given the evaluation results of *GCC CNN*, it is evident that deep learning can be used to implement an effective binaural sound source localisation system. However, the results also show that in comparison to a signal processing baseline, this model does not offer a significant improvement in noise and reverberation robustness.

Addressing the second part of the research question, the deployment of the SSL model in conjunction with a rotation algorithm made it possible to overcome the limitations of a binaural microphone configuration. This Enabled the system to perform robust 360° localisation. This finding reiterates the results presented [**?**], in which a different rotation scheme was used to similar effect. The experimentation in this project also highlighted the difficulty of performing 360° binaural SSL without a rotation scheme. The ultimate goal of an end-to-end model of 360° binaural SSL using deep learning is a problem requiring further research.

The poor performance of the models that used unprocessed audio as input suggests that using the techniques presented in [**?**] may not apply directly to binaural SSL. Relatively little literature exists where deep learning based binaural SSL is attempted without using extensive data preprocessing. Furthermore, the literature that does approach this problem in this way uses frequency-domain input representations. Thus, the problem of using deep learning with unprocessed time-domain input waveforms for binaural SSL is a very much an open research question. Much of the research that has successfully used minimal feature-engineering for SSL has done so in the context of multi-microphone arrays.

The results show that the most useful DL model developed in this study was *GCC CNN*. The idea of using GCC-PHAT input features in machine learning SSL systems is commonplace, however, comparable results to those presented in this project have been achieved with far simpler and shallower networks [**?**]. Thus using the very deep CNNs presented in this project is likely not necessary to perform machine learning based SSL with hand-crafted features.

Successes and Failures

The technical specifications outlined in Chapter 3 are relatively stringent. These were created with the example use case of a search and rescue sound localisation system in mind. The fact that all the acceptance test procedures were not passed indicates that in its current form, the system needs further development to be considered useful in a real life setting. However, that is not to say that the system developed in this project does not show promise. The system as it stands performs reliable DOA estimation provided the conditions are not excessively noisy or reverberant. The system has also been able to generalise to types of audio not seen in training, however, it could be argued that the heavy-lifting in this instance is taken care of by the GCC-PHAT preprocessing, as the generalised cross-correlation of speech signals and music are not too dissimilar.

One of the most exciting results of this project is that a deep learning model of binaural SSL was trained entirely on synthesized data and then deployed relatively successfully on a real world system. This is a promising developement as, given sufficient compute resources, these techniques could be used to generate arbitrarily large datasets to be used for deep learning based SSL.

Concluding Remarks

This has been a challenging research project, requiring the development of an understanding of various domains. Binaural sound localisation is a complex engineering problem that has a myriad of possible applications. The use of machine learning in pursuit of a robust solution to the problem is a useful and open research question. This project has successfully met the requirements laid out in the project brief. However, as shown in the results, further research is required if a useful real world system is to be developed.

Chapter 10

Recommendations

With a goal in mind of a robust end-to-end SSL model, the most pertinent problem requiring further research is the choice of deep learning techniques used. In this paper, only convolutional networks were considered. However, the use of recurrent networks such as LSTMs could prove useful as they have been shown to be particularly well suited to sequential time-domain data. Futhermore, improvements to the work presented in this paper could be made with hyper-parameter optimization. The time constraints of this project prevented detailed experimentation with different training methodologies. The improvement of said methodologies could possibly lead to better performance of the raw audio models. Another approach that could be taken is to train a model using a very large dataset synthesised using the techniques outlined in this paper. Following this, a smaller dataset recorded using the relevant hardware configuration could be used for transfer learning. Transfer learning has proven to be extremely useful in image recognition and NLP. Thus there is reason to believe that it may have application in this field as well.