Multi-channel Enhancement and Diarization for Distant Speech Recognition

Submitted in partial fulfillment of the requirements

of the degree of

Master of Technology

by

Sachin Nayak

17307R002

Under the Guidance of:

Prof. Preeti Rao

and

Prof. Rajbabu Velmurugan

Department of Electrical Engineering

Indian Institute of Technology Bombay

2020

Dissertation Approval

This dissertation entitled **Multi-channel Enhancement and Diarization for Distant Speech Recognition** by **Sachin Nayak** is approved for the degree of Master of Technology.

Roll Number: 17307R002

Guides: Preeti Rao, Rajbabu Velmurugan (IIT B)

Internal Examiner: Kumar Appaiah (IIT B)

External Examiner: Sunil Kopparapu (TCS)

Contents

List of Figures

Abstract

This work proposes a system to solve the task of generating transcriptions for meetings sessions where multiple speakers are speaking in the presence of ambient noise and reverberation, using multi-channel audio recordings. Multi-channel speech is enhanced to improve the speech quality and then it is used for speaker diarization and speech recognition. Single channel methods: Weighted Prediction Error (WPE) and Convolutive Non-Negative Matrix Factorization (CNMF) are initially used for dereverberation. The multi-channel output of each of the single-channel dereverberated audio was fed as input to source localization and beamforming. Source localization Generalized Cross Correlation (GCC) techniques: GCC-PHAT (Phase-Transform), GCC-SCOT (Smooth Coherence) were implemented. The localization estimates are used to perform beamforming, which produces single-channel enhanced audio assuming that only one speaker was active at any instance. Several beamforming methods: conventional Delay Sum Beamforming (DSB) and other methods: neural-network-based mask estimation: Minimum Variance Distortionless Response (MVDR) and Generalized Eigenvalue (GEV) beamforming were implemented. Diarization is used as a pre-processing step before ASR. One way to solve the diarization problem is to use TDOA cues over the entire conversation length and assign speaker ids depending on their estimated spatial location. Another way is to use models that exploit speaker characteristics to obtain the diarization output. This work also exploits the complementary information between TDOA and x-vectors feature types. For ASR, multi-condition training was done using clean and reverberated audio to observe ASR improvements using DNN-HMM and TDNN-HMM acoustic models. The output of diarization is fed to the best performing ASR system to finally obtain the transcripts. For evaluations, we investigated two datasets: simulated multi-channel data using LibriSpeech and recorded natural conversations at Tata Consultancy Services (TCS) office rooms. An interface (GUI) based on Python is built and implemented, incorporating this enhancement-diarization-ASR system. A synthesis GUI was developed for generating the multi-channel audio. The developed system performed well and was demonstrated on both simulated and realistic multi-channel audio.

Chapter 1

Introduction

The performance of conventional Automatic Speech Recognition (ASR) systems degrades as the distance between the speaker and the microphone increases. This is mainly due to effects such as background noise, overlapping speech from other speakers, and reverberation. While traditional ASR systems underperform for speech captured with far-field sensors, there are several novel techniques within the recognition system as well as techniques developed in other areas of signal processing that can mitigate the degrading effects of noise and reverberation, as well as separating speech from overlapping speakers. The advancement in speech recognition has led to the development of real-world applications ranging from Smart Homes, voice-controlled personal assistants like Apple's Siri, Google Assistant, or Amazon's Alexa, where the speaker can be distant while using it. But, the deployment of speech recognition systems into the real world also comes with a lot of challenges. For example, an ASR system should be robust to noise but only low reverberation. On the other hand, meeting room environments, home environments typically have a much higher SNR but have moderate to high amount of reverberation with an additional challenge of overlapping talkers [1]. Also the distance of the person speaking from the microphone also affects the ASR performance of ASR especially in these challenging acoustic conditions. Current DSR systems for conversational speech are considerably less accurate than their close-talking equivalents, and usually require complex multi-pass decoding schemes and sophisticated front-end processing techniques [2], making distant speech recognition a highly challenging area. The problem we are trying to solve here is to automatically transcribe a conversation during a meeting in a closed room. There are multi-speakers seated around a table and usually several microphone sensors placed on the table. The entire meeting session which is being recorded by these microphones is to be recognized to obtain which speaker spoke what as shown in Figure 1.1

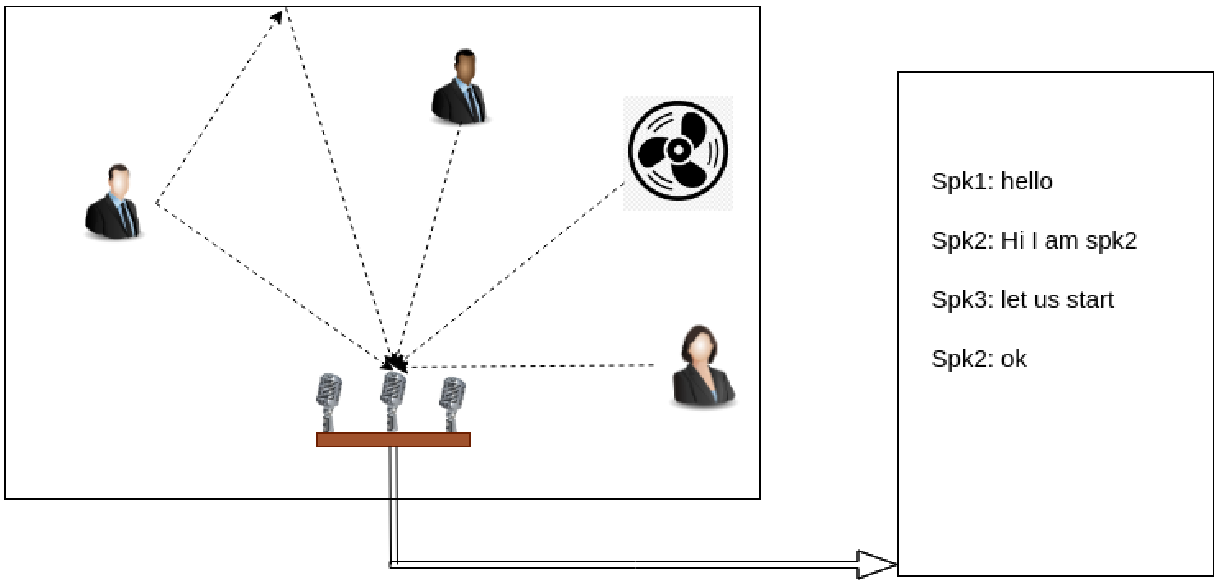


Figure 1.1: Scenario of distant ASR

This work tries to overcome the existing challenges of the distant ASR (DSR), namely noise and reverberation. Also when multiple speakers are speaking in a conversation, we may want to know who spoke what (diarization) in the recorded speech. This work also tries to study and pose a solution to the speaker diarization problem, which is discussed in subsequent chapters.

1.1 Multi-channel audio processing

Single-channel recognition results in poor speech recognition when the speaker is distant from the recording microphone device [3] (typically of something more than 0.2m). Also, single-channel approaches are affected by low SNR and high reverberation conditions. This made the popular use of microphone array along with the added advantage of using its strategic microphone placement to obtain spatial information. Firstly a microphone array can locate and track a speaker since different positions of speakers produce different instances of this signal being received at the microphones. Secondly, simultaneous source signals overlapping in frequency domain but coming from different directions can be separated using such an array. Microphone arrays can steer its response in different directions, allowing it to extract the signal from a particular direction attenuating other signals from other directions, which is called beamforming.

A large number of algorithms for microphone array processing were borrowed or generalized (in a very simple manner) from narrowband array processing. The advantage of this is that most algorithms conceived for decades in antenna arrays can be extended without much effort [4]. In antennas, array processing is used for directional reception as well as transmission of narrowband signals. So much of the theory behind the construction of spatial filters were derived from these narrowband processing techniques. Since speech is a wideband signal, most of the array processing algorithm works by considering each frequency bin as a narrow band signal and applying the narrowband algorithms to each bin.

1.2 System Description

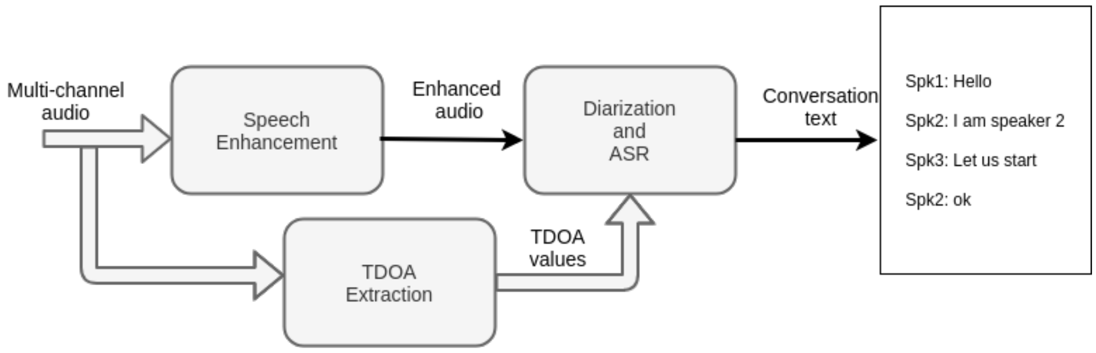


Figure 1.2: Enhancement-Diarization-ASR system

The system we plan to incorporate to deal with DSR is shown in Figure 1.2 and the decomposed blocks are shown in Figure 1.3.

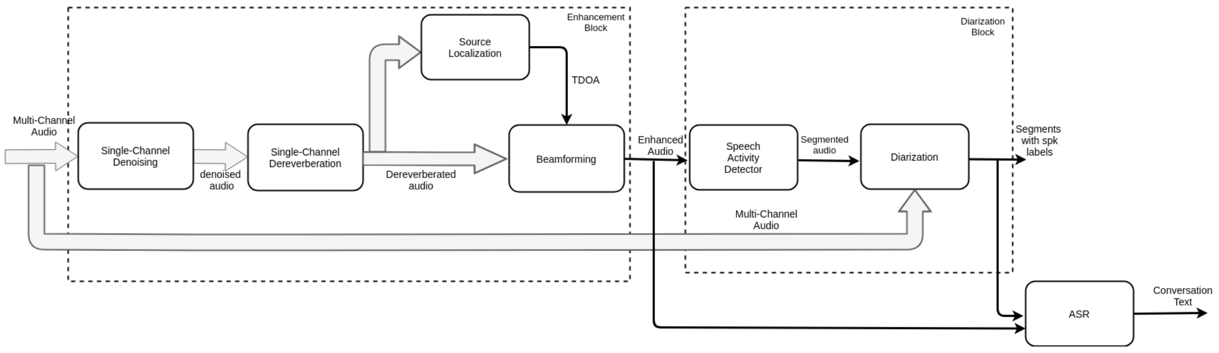


Figure 1.3: Block diagram of the multi-channel DSR system

The main components of the entire system is shown in Figure 1.3 comprising of front end enhancement stage followed by a speech recognition system which takes multi-channel audio as input and outputs the conversation decoded text in terms of who spoke what and when. Following is a brief description of the different stages involved:

- Enhancement block:

  Single Channel Enhancement : Single-channel dereverberation is done before source localization improve localization estimates. Some single-channel denoising methods before beamforming is also sometimes seen to have ASR performance improvement.

  Source Localization : The technique of finding the direction of the speaker using the information from the signals received at the microphone arrays. This spatial information is used as a steering direction by the beamforming algorithm. Various source localization algorithms are described in Chapter 2.

  Beamforming : The process of steering the response of the microphone array towards the source direction thereby attenuating the undesired signals from other directions. Chapter 3 explains the working of different beamforming techniques.

- Diarization/ASR:

  Speech Activity Detection (SAD): The enhanced output from the beamforming is used to obtain speech only segments in the audio, having start-end timings, where the role of SAD is to obtain non-silence regions in the audio.

  Diarization : Solves the problem of who spoke from 'what time to what time'. Features are extracted out of segment output of the SAD and then clustering is done to assign speaker labels to each such segment. TDOA features that capture spatially location of speakers and embeddings obtained from speaker-recognition models that capture speaker-characteristic information were studied and are discussed in Chapter 4.

  ASR : The speech recognition system generates a hypothesis regarding what exactly the speaker said from the acoustic waveform with the help of trained acoustic and language models. These hypotheses are compared with reference text to compute accuracy in terms of WER. Chapter 5 presents the speech recognition accuracies of various beamforming methods under different conditions. The ASR output is guided with diarization speaker labels to obtain the conversation type decoded text as shown in Fig 1.2

Although we investigated single-channel enhancements like WPE [5], NMF [6] for dereverberation, and Wiener [1], Spectral subtraction [7] for denoising, the subsequent chapter will discuss only the part of the system after the single-channel enhancement step.

1.3 Datasets

We have analyzed and investigated the above system blocks on two different types of datasets, one which is simulated from single-channel audio recordings from a standard speech research dataset and the other being natural microphone array recordings, recorded in different meeting rooms of TCS.

1.3.1 Simulated Multi-channel Data

The idea was to simulate multi-channel audio, to be used for our study of multi-channel enhancement, diarization, and their ASR performance. These audio recordings consist of multiple speakers speaking in front of the single microphone array(the configuration of the array is fixed for all recordings). Although in any multi-speaker recording there is a chance of two or more speakers speaking simultaneously, we only consider a scenario when there is no speaker overlap. Also, the position of the speaker and the array remains the same throughout the recording. A typical configuration is shown in Figure 1.4. Typically rooms of 3 different sizes are considered from a small to a large room. The setup is such that the array is assumed to be placed around a meeting table with speakers seating around it. The array configuration is fixed to a circular array with a radius of 5cm (4 on the circumference at 90^o apart and 1 at the center). An entire array recording is called a session. A session consists of speakers speaking continuously or taking turns while speaking. These continuous chunks of spoken audio of a speaker are termed as utterances. So a session consists of multiple utterances of one or more than one speaker. We use the image method proposed by Allen and Berkley [8] which is the most commonly used method in the acoustic signal processing community to create synthetic room impulse responses. was used to generate multi-channel audio. This audio generator tool generates the Room Impulse Response (RIR) for each microphone a given source speaker, for every pair of microphones and speakers in the session. The position of each microphone, each speaker, room dimensions, and reverberation time constant (T60) are arguments to this RIR-generator.

If the number of channels is say M and number of speakers are N with labels from S={s$_1$,s$_2$,..s$_N$}, then the RIR-generator it will give M x N RIRs for each speaker-channel given as a matrix $\mathbf{R}_{M\times N}$ pair. So the signal generated for ith microphone pair for the source signal x$_{s_j}$ is:

$$y_{ij} = \mathbf{R}[ij] \circledast x_{s_j} \tag{1.1}$$

This is done for all speakers j=1:N in the session for the ith channel. Then all the obtained y$_{ij}$ are concatenated in the time domain for all j speakers by adding random silence of 02.s-0.5s to give channel single z$_i$. In the last step, noise is at to the all the channel signals z$_i$, for i=1,2..M.

After it generates the RIR, corresponding speaker utterances are convoluted to obtain the multi-channel session audio. Silence is added after each utterance and stationary noise added at different SNR to each channel of the audio. For enhancement analysis, we considered simulated multi-channel audio, where sessions have multiple speakers but with only one utterance each (we refer to this as 'Dataset A'). Speakers are placed at different azimuths at an angular separation of 30^o from the array. The placement of speakers is restricted to one side of the array. The speaker array distance was varied with a minimum distance of 1m to a maximum of 2m. The total duration of the dataset is 5hrs. The details of the Dataset A are shown in Table 1.1.

Table 1.1: Parameters varied simulating multi-channel audio for single-utterance per speaker type session: Dataset A

Parameter	**Value**
No. of Speakers	1 - 5
Duration of each utterance	4 - 8 secs
Duration of silence after each utterance	0.5s-1s
T60	0.2s-0.5s
SNR	5-10dB
Number of rooms	3
Distance of speaker from array	1m - 2m
Session duration	6s-30s

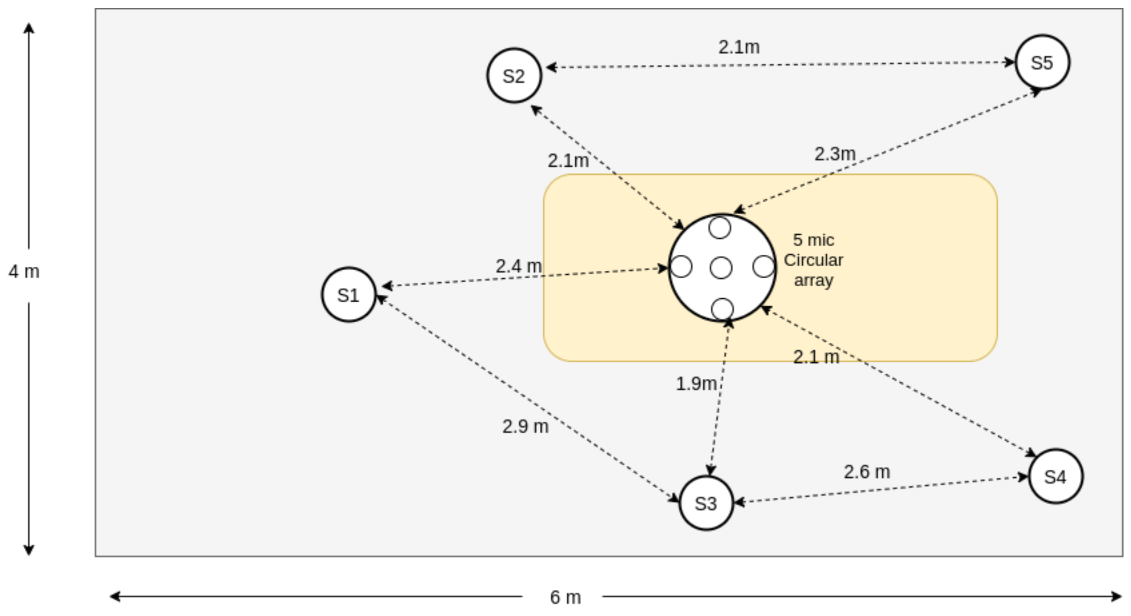


Figure 1.4: Meeting room configuration

The individual speaker utterances are chosen from the ’dev-clean’ set of LibriSpeech [9](LibriSpeech is a corpus of approximately 1000 hours of 16kHz read English speech, the data is derived from reading audiobooks from the LibriVox project), a dataset of single-speaker utterances. The simulated sessions are stored along with the corresponding ground transcripts for each session, taken from LibriSpeech.

For diarization analysis, we simulated meeting type conversation recordings, to be used as reference dataset (Dataset B). The idea was to have speakers having a conversation where each speaker speaks more than once. We assume that the speakers and microphone positions are considered in a stationary position and no two or more speakers speak at a time (same as in dataset A). Here the speakers sit all around the microphone array. Either the speakers sit near to the array (∽1m) or far from the array (∽2m) in a session.

To generate a conversation session for a given number of speakers, say ’n’, then we choose n distinct speakers from the mini-LibriSpeech (Utterances of individual speakers are chosen from the ’dev-set’) dataset and then choose the utterances corresponding to those speakers. Similar to the way we to generated ’dataset A’ data, we obtain RIRs for each speaker and microphone and then convolve with corresponding speaker utterance, only in this case the convolution is done for several utterances of a given speaker in the turn. Silence is added after each utterance to have the pause in a conversation after a speaker speaks. Noise is added from Reverb Challenge [3]. The parameters that are varied to generate such multi-channel sessions are given in Table 1.2.

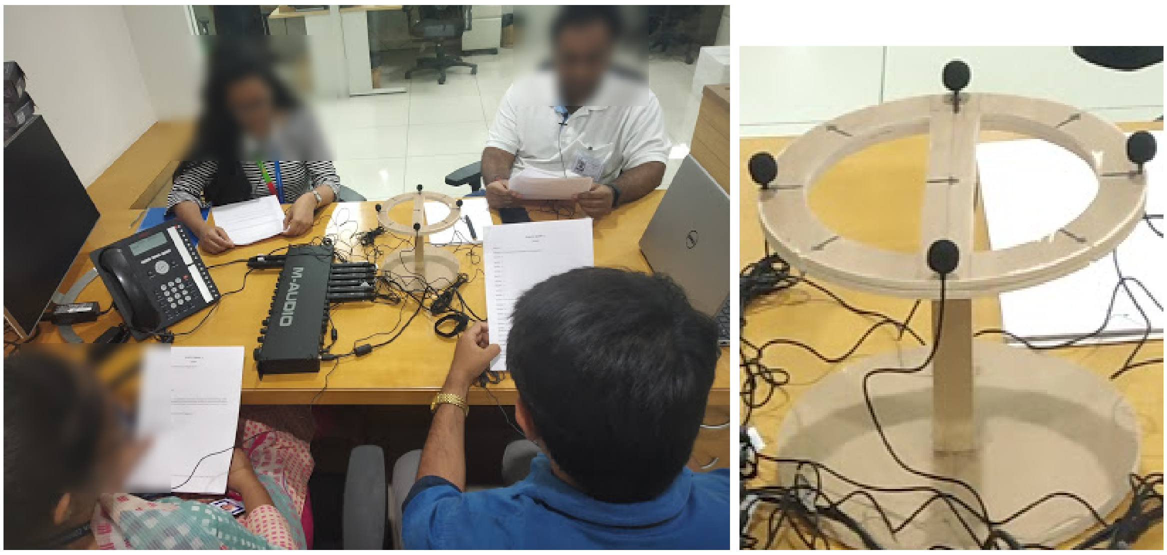
Table 1.2: Parameters to be selected for simulating meeting audio of a conversation session:Dataset B

Parameter	**Value**
No. Speakers	3, 5
No. of times a speaker occurs	20 - 30
Duration of each utterance	4s-8s
Duration of silence after each utterance	0.2s-0.6s
T60	0.2s-0.5s
SNR	15, 20 dB
Room Dimension	6m x 4m x 2.5m
Distance of speaker from array	1m, 2m
Session duration	7min-10min

The reverberation constant (T60), number of speakers in a conversation, SNR levels and near/far are varied so that for each selection of these parameters, there are equal number of conversation sessions. The order of speaker turns is obtained by randomly choosing a speaker from the given number of speakers associated with the session. Typical duration of such a conversation session is approximately 7min-10min. The dataset sessions are stored with ground truth transcripts and the corresponding speaker labels, taken from Libripseech. This dataset will serve as a typical multi-channel meeting conversation test set for comparison of the various diarization/ASR methods in the following sections.

1.3.2 TCS Recorded Data

This dataset consists of multi microphone audio recordings of meeting type sessions, recorded using microphone array shown in Figure 1.5b, recorded at TCS office, Mumbai.



(a) Picture of a recording room setup (b) Microphone mounted frame

Figure 1.5: The recording room set up in (a) and microphone array mounted on a frame used in (b)

The total number of array microphones mounted on the circular frame varies and each speaker in the meeting sessions has a lapel microphone. The array mounted microphones are placed around the rim of the circular frame and one microphone is typically placed at the center of the frame, like a circular array configuration. This circular arrangement has a radius of 5cm. The array is placed on the table as shown in Figure 1.4 with speakers sitting alongside the table. The location of the array is fixed and the speakers are stationary for almost all the session recordings. The speakers sit with a distance of around 0.7m - 2.5m from the array, depending upon where the speakers are placed around. The speakers converse in a non-overlapping fashion, i.e. only one speaker is active at a time.
Sessions recorded consist of mixed gender/all-female types of recordings. The sessions

consist of recordings of two types:

- Each speaker speaks one utterance only in the session. (like Dataset A)
- A conversation where speakers speaker multiple utterances in the session. (like Dataset B)

Other details of the TCS recorded dataset is shown in Table 1.3

Table 1.3: Summary of the TCS dataset

Session Details	**Values**
No. of Speakers	3 - 5
Total conversation data	Total 43min:
	Scripted conversations(29min)
	Natural conversation(14min)
Total single-speaker data	35min
Session duration	2min-10min
Number of different rooms	4
Array radius	5cm

The dataset is stored along with the documentation describing each session, room configuration with the transcripts corresponding to each recorded session. The recording setup consisted of 'M-Audio M-Track Eight USB Audio' interface connected to 'AKG C417 PP Professional Lavalier' microphones which synchronized all the connected array and lavalier microphones together. A software called 'Reaper' was used to record the audio, at the 48KHz sampling rate and then store the individual channel recordings on the computer.

Chapter 2

Source Localization

For beamforming, it is essential to obtain the estimates of the location of the speaker to apply spatial filtering techniques. The problem of finding the source location using sensor arrays has long been of great research interest given its practical importance in a great variety of applications. In these applications, source localization is more commonly referred to as the direction of arrival (DOA) or time difference of arrival (TDOA) estimation.

One possible way of obtaining the azimuth is to run a search algorithm along with different hypothesized locations and finding the directions where the hypothesis function obtains a maximum or minimum. Such types of search algorithms are called Steered Response Power (SRP) [10] search. Further, the beamformer is steered towards the obtained hypothesis angle, and the direction which receives maximum power is the source direction. Another way for estimating the source angle is to obtain the TDOAs between the microphone pairs and subsequently use this information to obtain the angle. Source Localization using TDOA estimating is the one which we will focus on.

2.1 Cross-correlation Based

One of the simplest approaches to obtain TDOA is to find the time shift where peaks appear in the cross-correlation of signals between two channels. Let $\mathrm{x}_1(\mathrm{t})$ and $\mathrm{x}_2(\mathrm{t})$ be the signals received at two different channels, then the cross-correlation (CC) R_{x1x2} can be expressed as:

$$R_{x1x2} = E[x_1(t)x_2(t)] \tag{2.1}$$

The time delay TD between two channels signals is the time shift for which cross-correlation is maximum given as:

$$TD = argmax_{\tau}(Rx_1x_2(\tau)) \tag{2.2}$$

CC method works when there is uncorrelated stationary noise but suffers from early reverberation. Here the delay computed from simple cross-correlation will also give rise to multiple peaks because of multiple relatively high signal power reflections of the source signal.

2.2 Generelized Cross-correlation Based

TDOA values are more accurate when obtained using Generalized Cross-Correlation (GCC) [11]. GCC based algorithms were introduced to increase the robustness of CC method towards noise and reverberation by applying an additional weighing factor to each frequency bin. GCC function $R_{y_iy_j}$ with the parameter as the delay between microphone pair {i,j} and every frame t can be expressed as:

$$\begin{aligned} R_{y_iy_j}(t,\tau) &= \int G_{y1y2}(t,f)e^{j2\pi f\tau}df \\ &= \int \psi(f)G_{x1x2}(t,f)e^{j2\pi f\tau}df \end{aligned} \tag{2.3}$$

Here $\psi(f)$ represents the weighting factor applied to each frequency bin of short time cross power density G$_{x1x2}$(t,f) of the input multi-channel audio signal. The TDOA TD$_{ij}$ is estimated by obtaining the delay value for which GCC function attains maximum value i.e.

$$TD_{ij}(t) = argmax_{\tau}(R_{y_iy_j}(t,\tau)) \tag{2.4}$$

The weighting function which produced satisfactory result as discussed below.

GCC - Smoothed Coherence Transform (SCOT)

The GCC function with SCOT weighing factor is given by:

$$R_{y_iy_j}(t,\tau) = \int \frac{G_{x_ix_j}(t,f)e^{j2\pi f\tau}df}{\sqrt{G_{x_ix_j}G_{x_ix_j}}} \tag{2.5}$$

GCC - Phase Transform (PHAT)

The GCC function with PHAT weighting factor is given by:

$$R_{y1y2}(t,\tau) = \int \frac{G_{x1x2}(t,f)e^{j2\pi f\tau}df}{G_{x1x1}G_{x2x2}} \tag{2.6}$$

Because of the normalization of the cross power spectral density by the weight term, the delay component in the cross-correlation is left as a residue inside the inverse Fourier Transform. GCC-PHAT performs better in the presence of reverberation and stationary noise and hence will be used further for beamforming.

2.3 TDOA Estimates from Beamformit

The tool Beamformit [1] uses a dual step Viterbi filtering of TDOA values and removing poor TDOA values. TDOA value does not show any useful information when it is computed over a silence (or mainly silence) region or when the SNR of either of the signals being compared is low, making them very dissimilar. The first problem could be addressed by using a speech/non-speech detector before any further processing, but prior experimentation indicated that further errors were introduced due to the detector. The the selected algorithm applies a simple continuity filter on the TDOA values for each frame index with $'t'$:

$$\begin{aligned} \mathbf{TD}^n_{i,ref}(t) =& \mathbf{TD}^n_{i,ref}(t-1) \quad if \quad max_\tau(R_{i,ref}(t,\tau)) < \text{thresh} \\ =& \mathbf{TD}^n_{i,ref}(t) \qquad\quad if \quad max_\tau(R_{i,ref}(t,\tau)) > \text{thresh} \end{aligned} \tag{2.7}$$

where $\mathbf{TD}^n_{i,ref}$ is a vector of N-best TDOA values corresponding to N highest peaks in the GCC-PHAT correlation for microphone pair (i,ref) and *thresh* is minimum correlation value, for i=1:M,i$\neq$ ref,ref being the reference microphone. The dual step Viterbi TDOA filtering stage is given in Fig 2.1.

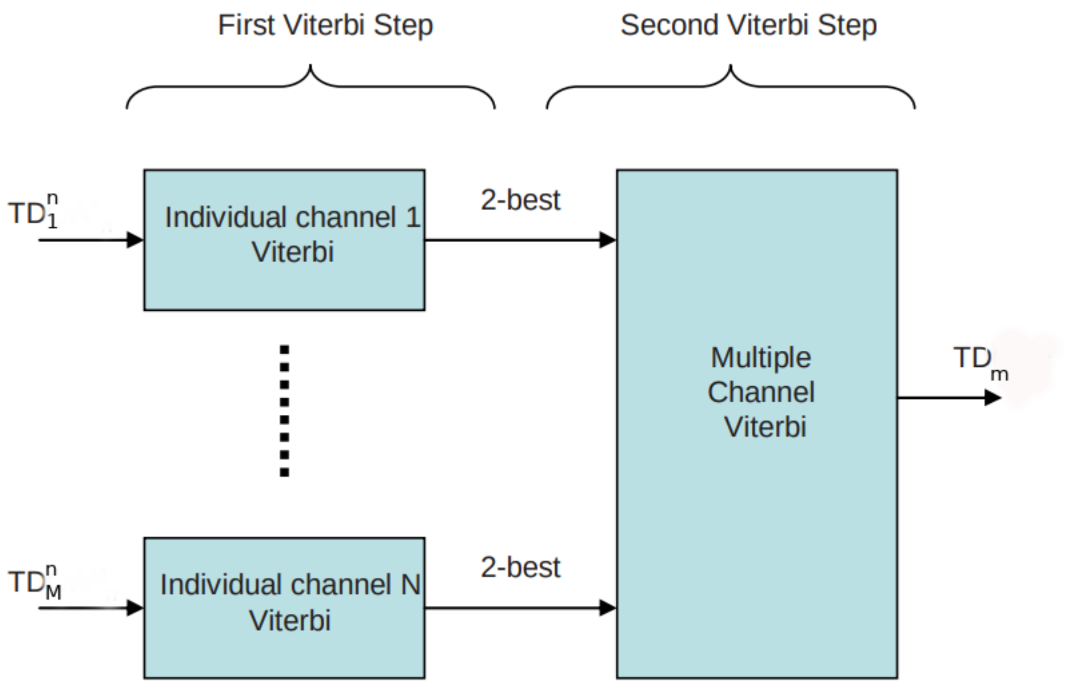
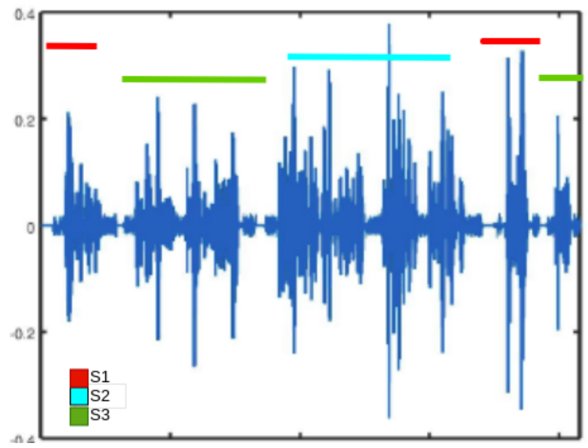


Figure 2.1: Dual step Viterbi [1] used in Beamformit

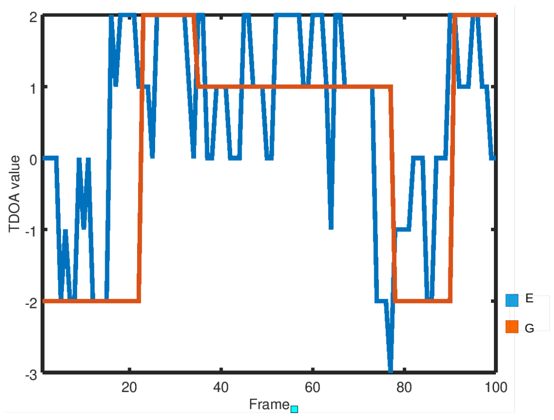
It gives one TDOA value for each channel w.r.t. the reference microphone, for every frame (typically used frame window size of 500ms). It is observed that the TDOA values across the frame do not change even in moderate reverberation conditions, unlike GCC-PHAT where the only peak valued index is considered as delay, the estimation of which is affected by noise and reverberation. The TDOA vector $\mathbf{TD}(t) = [\ TD_1,\ TD_2, \ldots TD_{M-1}\ ]$ obtained for every channel pair w.r.t reference channel and for every frame is the final TDOA output of Beamformit, which is later used as a diarization feature, as discussed in chapter 4.

2.4 Some Observations

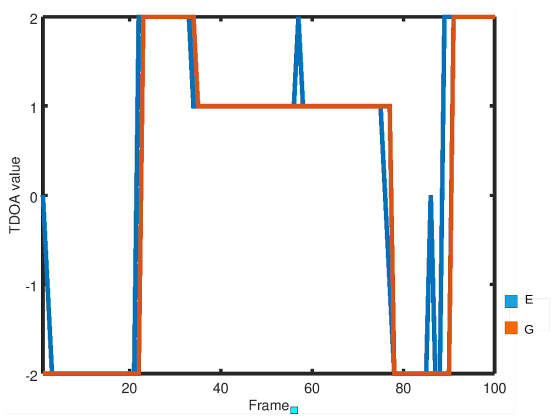
We compare the TDOA estimated obtained from GCC-PHAT to that of Beamformit which filters some more of poor GCC-PHAT estimates for various reverberation conditions (t60). Fig 2.2 shows the TDOA obtained for two sessions (both have different t60s) of duration ∽25s from Dataset A where three speakers are speaking and are placed at 30^o, 60^o and 150^o with respect to array axis. The window size for TDOA computation for both GCC-PHAT and Beamformit is 500ms with a hop of 250ms. It shows the ground truth TDOAs where the value 2 corresponds to 30^o, 1 to 60^o and -2 to 150^o. Here the angle is obtained using the delay between the microphone by $\cos^{-1}(\frac{\tau.c}{d.f_s})$ where d is the distance between microphone pair, τ is the delay between the microphone pair, c is the speed of sound in air and f_s is the sampling rate.



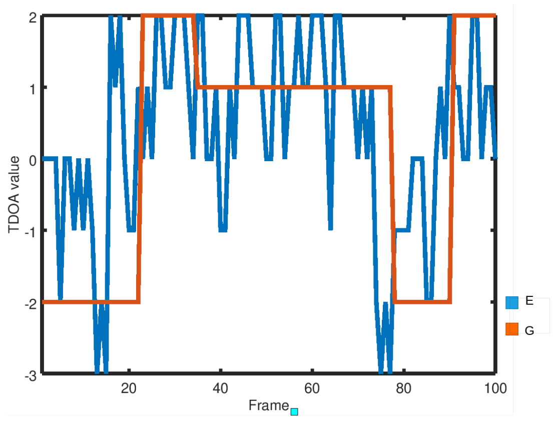
(a) Waveform of the one of channel showing active speaker labels



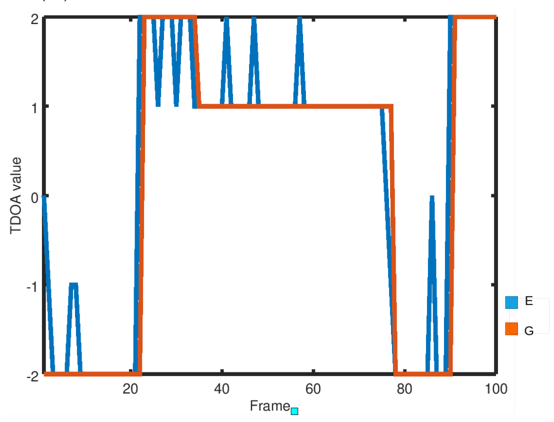
(b) GCC-PHAT TDOAs at t60 = 0.2



(d) Beamformit TDOAs at t60 = 0.2s



(c) GCC-PHAT TDOAs at t60 = 0.4s



(e) Beamformit TDOAs at t60 = 0.2s

Figure 2.2: Showing TDOA values obtained for one of the microphone pair using GCC-PHAT in (b),(c) and using Beamformit in (d),(e) for t60=0.2s,0.4s and for three speakers placed at 30^o, 60^o and 150^o. Legend E-Estimated , G-Ground Truth

It is observed that TDOAs obtained from Beamformit are more reliable as less variation is seen across frames as compared to GCC-PHAT TDOAs. TDOAs values from GCC-PHAT get worse for higher t60 of 0.4 while there are minor changes in TDOA values obtained from Beamformit.

Table 2.1: Showing mean angular error and variance of angular error on Dataset A for all speakers located at 30 ^{o}, for GCC-PHAT and Beamformit estimated TDOAs

T60(s)	Method	Mean(Angular Error in ^{o})	Variance(Angular Error in ^{o})
0.2	GCC-PHAT	10.3	5
	Beamformit	3.1	2.2
0.3	GCC-PHAT	18.0	6.2
	Beamformit	5.1	3
0.4	GCC-PHAT	25.4	7.1
	Beamformit	12.2	3.7

Table 2.1 shows Angular error obtained for different t60 for GCC-PHAT and Beamformit method. It is obtained of all speakers located at 30^o in Dataset A. Here the mean angular error is defined as :

M(Angular Error^o) = (Estimated Angle^o - Ground Truth^o Angle)/(Number of points or frames). Similarly, we can define the variance of angular error.

It is seen that for all t60s from 0.2 to 0.4, the mean and variance of angular error for Beamformit is lower as compared to GCC-PHAT. Also, the variance of angular error do not increase much for Beamformit at t60 increases relative to GCC-PHAT.

TDOA estimation was observed to be affected by the amount of reverberation in the multi-channel audio. This affects spatially filtering methods (Beamforming), discussed in the next chapter which rely on these TDOA estimates. Thus, doing single-channel dereverberation on the multi-channel audio was seen to improve the TDOA estimates, thus further improving the beamforming performance. Also, because of the reliable TDOA estimation of Beamformit and less variation of the estimated values for a given speaker utterance, these TDOA estimates are used as a diarization feature, in chapter 4.

Chapter 3

Acoustic Beamforming

Beamforming has a long history; it has been studied in many areas such as radar, sonar, seismology, communications, to name a few. It can be used for a lot of different purposes, such as detecting the presence of a signal, estimating the direction of arrival (DOA), and enhancing the desired signal from its measurements corrupted by noise, multiple sources, and reverberation. Traditionally, a beamformer is formulated as a spatial filter that operates on the outputs of a sensor array to form the desired beam (directivity) pattern. Such a spatial filtering operation can be further decoupled into two sub-processes: synchronization and weigh-and-sum [12]. The synchronization process is to delay (or advance) each sensor output by a proper amount of time so that the signal components coming from the desired direction are synchronized. The information required in this step is the TDOA, which, if not known a priori, can be estimated from the array measurements using time-delay estimation techniques. The weigh-and-sum step, as the name indicates, is to weigh the aligned signals and then add the results together to form one output. Although both processes play an important role in controlling the array beam pattern (the synchronization part controls the steering direction and the weight-and-sum process controls the beamwidth of the main lobe and the characteristics of the sidelobes), attention to beamforming is often paid to the second step on determining the weight coefficients.

The spatial-filter based beamformers were developed for narrowband signals that can be sufficiently characterized by a single frequency. That is, the response of the beamformer (like beamwidth, the attenuation factor) is a function of signal frequency. One way to design a broadband beamformer is to use a sub-band decomposition and obtain the narrowband beamformer for each sub-band of signal frequency [13]. In that way, a speech signal which is typically broadband can be enhanced using such a technique. The method of beamforming can be represented in a form on the filtering of each channel with different filters after delaying each channel with the estimated time difference delay as shown in Fig 3.1.

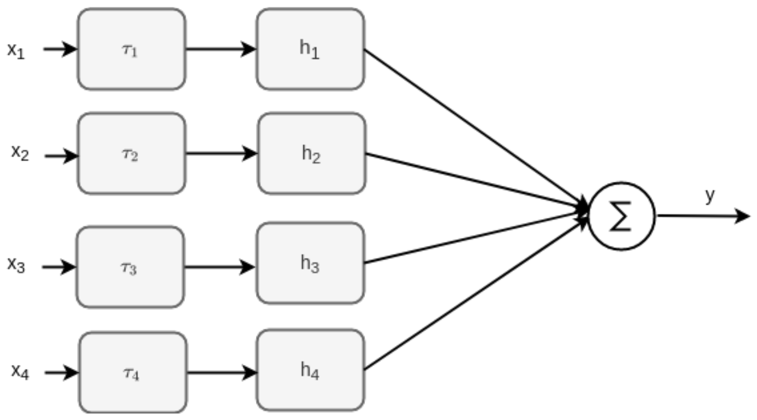


Figure 3.1: Beamforming Technique

Let $\mathbf{h}$(f) be a column vector with each element representing the transfer function of filter at the output of each channel (considering 4 channels for illustration). Then the output at the beamformer is given by:

$$\mathbf{y} = \mathbf{h}^T \mathbf{x}_a \tag{3.1}$$

where $\mathbf{x}_a = \{x_{1a},\ x_{2a}, \ldots x_{Ma}\}$, with each x_i being the delayed by the estimated delays from the localization algorithm from the set $\boldsymbol{\alpha} = \{\tau_1,\ \tau_2, \ldots \tau_M\}$ (for convenience the time parameter is omitted in the equation). Further, the subscript a denotes the delayed channel vector.

3.1 Delay Sum Beamforming (DSB)

The advantages of using an array to enhance the desired signal reception while simultaneously suppressing the undesired noise can be achieved by a DSB. The first step is to time-shift each sensor signal by a value corresponding to the TDOA between that sensor and the reference one. The next step is to sum the delayed signal giving the output of delay-sum beamformer, which according to the beamforming filter $\mathbf{h} = [\frac{1}{M}, \frac{1}{M}, \dots \frac{1}{M}]^T$.

$$y(t) = \frac{1}{M} \sum_{m=1}^{N} x_{ma}(t) + n_{ma}(t) \tag{3.2}$$

Since signal received at each channel is superposition of speech signal s(t-t$_i$) and n(t), for the actual delay t$_i$ the signal at the output of the beamformer is expressed as

$$y(t) = \frac{1}{M} \sum_{m=1}^{N} s_(t - (\tau_i - t_i)) + \frac{1}{M} n_{ia}(t) \tag{3.3}$$

with τ_i being the estimated delay. In the ideal case when t$_i$ $= \tau_i$, for all i=1:M channels, the beamformer mainlobe points in the direction of the source.

3.2 Adaptive Beamforming

Once the array geometry is fixed and the desired steering direction is determined, the characteristics of the beam pattern of a DSB, including the beamwidth, the amplitude of the sidelobes, and the positions of the nulls, would be fixed. This means that if we want to adjust the beam pattern, we have to make physical changes to the array geometry, which is virtually impossible once an array system is delivered. The fixed beamforming techniques can fully take advantage of the array geometry and source location information to optimize their beam pattern. However, the ability of a fixed-beamforming array system in suppressing noise and competing sources is limited by many factors, e.g., the array aperture. One way to achieve a higher SNR gain when the array geometry is fixed is through using the characteristics of both the source and noise signals, resulting in a wide variety of array processing algorithms called adaptive beamforming techniques, discussed in the following sections.

3.2.1 Generelized Eigenvalue (GEV) Beamformer

A more general form of a beamformer output is given in time domain as :

$$y(t) = s(t - t_i)\mathbf{h}^T\mathbf{a} + \mathbf{h}^T\mathbf{n}_a(t) \tag{3.4}$$

where $\mathbf{h} = [h_1,\ h_2, \dots h_M]$ are weights to be estimated and $\boldsymbol{\alpha}$ is the delay vector estimated from localization. In particular, taking h$_i$ = $\frac{1}{N}$, $\forall$ i, we get the DSB. Since the signal and noise are assumed to be uncorrelated, the correlation matrix of the vector signal y(t) can be expressed as $\mathbf{R}_{yy} = \sigma_2 \boldsymbol{\alpha}\boldsymbol{\alpha}^T + \mathbf{R}_{nn}$ where σ^2 is the source signal variance and $\mathbf{R}_{nn}$ is the noise correlation matrix.

With this general filter, the output SNR is written as:

$$SNR(\mathbf{h}) = \sigma_2 \frac{(\mathbf{h}^T \boldsymbol{\alpha})^2}{\mathbf{h}^T \mathbf{R}_{nn} \mathbf{h}} \tag{3.5}$$

can. One straightforward way of doing this is to find a filter h that would maximize the positive quantity SNR(h). This is equivalent to solving the generalized eigenvalue problem:

$$\sigma_2 \boldsymbol{\alpha}\boldsymbol{\alpha}^T = \lambda \mathbf{R}_{nn} \mathbf{h} \tag{3.6}$$

The solution to this is given by the beamforming vector which is the eigenvector corresponding to maximum eigenvalue, i.e

$$y(k) = \mathbf{h}^T_{\lambda_{max}} \mathbf{x}_a \tag{3.7}$$

This beamforming method is called GEV [13] beamforming. Since it tries to maximize the SNR, in some literature it is referred to as Maximum SNR beamforming.

3.2.2 Minimum Variance Distortionless Beamformer (MVDR)

As the name suggests, the MVDR beamformer [14] tries to minimize the output noise variance but subject to constraint that noise coming from the source direction is not high but constant given below:

$$min_h \mathbf{h}^T \mathbf{R}_{nn} \mathbf{h} \qquad \text{subject to} \qquad \mathbf{h}^T \boldsymbol{\alpha} = c \tag{3.8}$$

The solution to this optimization problem is given by:

$$\mathbf{h}^{MVDR} = \frac{\mathbf{R}_{nn}^{-1} \boldsymbol{\alpha}}{\boldsymbol{\alpha}^T \mathbf{R}_{nn}^{-1} \boldsymbol{\alpha}} \tag{3.9}$$

Thus the final output of the MVDR beamformer is given by:

$$y(k) = \mathbf{h}_{mvdr}^T \mathbf{x}_a \tag{3.10}$$

3.3 Neural Network Mask Based Beamforming

One way to compute the spatial noise co-variance matrix is to do a time average as :

$$\mathbf{R}_{nn} = \sum_{t=1}^{T} \mathbf{x}_a(t,f)\mathbf{x}_a(t,f)^H \tag{3.11}$$

Assuming that noise properties do not change over the entire audio signal, the chunk of frames of the audio before the onset of speech is used to compute this co-variance matrix. Note that the covariance matrix is a function of frequency.

Another method of obtaining this matrix is to use a neural network mask [15]. We have considered both MVDR and GEV beamformer here since both these methods require co-variance matrix estimation. For both beamformers the co-variance matrix R$_{NN}$ is obtained from the estimated masks **M(t,f)** which is then element-wise multiplied with **x**(t.f) **x**^{H}(t,f) .

$$R_{nn} = \sum \mathbf{M}_v(t,f).(\mathbf{x}(t.f)\mathbf{x}^H(t,f)) \qquad \text{where v=\{X=speech,N=noise\}} \tag{3.12}$$

To obtain this mask, we use noisy speech and its correspondingly labeled clean speech.

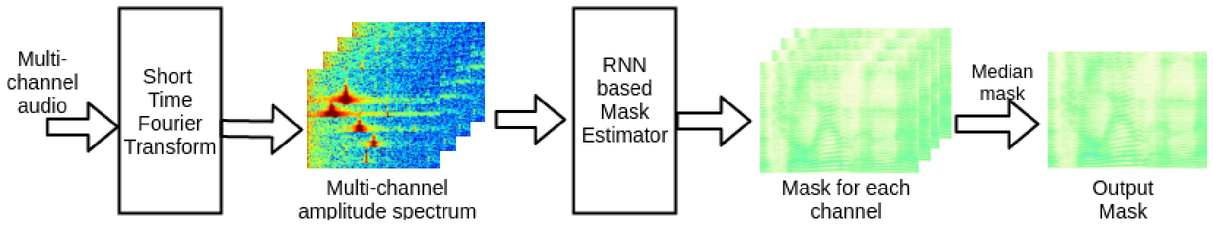


Figure 3.2: Mask estimation using RNN using multi-channel audio taken from [15]

An RNN based model is used to obtain mask spectrogram for speech for each channel. The mask obtained for each channel take values in {0,1}. Median value is obtained across channels for each time-frequency point to get the final output mask. If the obtained speech mask is M$_x$ then the noise mask is 1-M$_x$. The mask obtained is used to compute the noise co-variance matrix as eq 3.12. This obtained co-variance matrix is then used to compute the beamforming vector as in eq 3.6 and 3.9.

3.4 Observations

This analysis here was to compare the performance of the beamforming methods. The comparison was done on the simulated testset: Dataset A (one-utterance of each speaker type dataset mentioned in section 1.2). Here, each single-channel was first dereverberated using WPE as TDOA improvement was observed after dereverberation and then fed to the beamformer. TDOAs are computed using GCC-PHAT method for the analysis window of 20ms with 10ms hop. The mask estimation neural network was trained on CHiME4 data [16]. The estimated average performance of these beamforming enhancements is evaluated using cepstral distance (CD), perceptual evaluation speech quality (PESQ) and SDR of ′BSS Source Eval′. [17] for all sessions in Dataset A as shown in Table 3.1.

.

Table 3.1: Performance measures evaluated on Dataset-A for various beamforming methods discussed

Enhancement Measures			
BF-Method	CD	PESQ	SDR
None	5.7	1.2	4
DSB	5.6	1.6	4.2
GEV	4.7	1.3	4.2
MVDR	4.8	1.8	5.0
NN-GEV	4.3	1.9	5.1
NN-MVDR	4.2	1.9	5.2

Other evaluations include a comparison of these methods using ASR measures. ASR models are trained on the augmented clean-beamforming enhanced data and then used to obtain WER will be discussed in Chapter 7.

Chapter 4

Speaker Diarization

Speaker diarization for recordings made in meetings consists of identifying the number of participants in each meeting and creating a list of speech time intervals for each participant. Since many speech processing technologies, such as in automatic speech recognition or speaker recognition, assume the presence of only one speaker, diarization can be an important front end in scenarios where the single-speaker assumption can be violated. The role of diarization is to annotate recordings of multiple speakers in a way that specifies which speaker has spoken from what time to what time. It divides the input recording into segments or chunks as per the start and end of the utterance and then allots speaker label to each segment.
A typical diarization system consists of three main components: (1) Speech segmentation; the audio is divided into individual segments assuming that there is only one speaker in each such segment and tries to remove silence portions (2) Feature extractions; specific audio features like MFCC, speaker-specific features like i-vectors or embeddings from neural networks are obtained for the segments (3) Speaker Clustering, where the features extracted are clustered as per some distance measure and assigned speaker labels. In cases, where the number of speakers is not known, some methods can itself estimate the number of speakers.

The most common approaches for diarization of audio consist of using conventional acoustic features like MFCC, HMM-GMM based models trained on these features to obtain likelihoods and agglomerative clustering based on the obtained likelihoods [18]. Other diarization approaches depend on using spatial information of the speaker as a feature to distinguish between speakers in a conversation. These methods estimate the DOA of the speaker and then use a DOA classifier to assign labels to each speaker [19]. These DOA or TDOA features carry information about the location of the current speaker and they have been used as complementary features to conventional MFCC. Many state-of-the-art diarization systems for meeting recordings are based on the HMM/GMM framework and the combination of spectral (MFCC) and TDOA features. The combination happens at the model level, weighting the log-likelihoods of independent GMM models estimated on each feature stream [20] [21] [22]. Recent efforts have been successful in learning better representations automatically using i-vectors, while others use feed-forward or recurrent neural networks (RNN). Embeddings obtained from neural network [23] [24] models used for speaker recognition and verification tasks seem to have outperformed state-of-the-art techniques like the i-vector. These embeddings capture speaker characteristics and are not dependent on what the speaker spoke, i.e, they are text-independent.

One aspect of the segmentation-based approaches above is that the resulting diarization marks will be restricted to begin and end according to the segmentation boundaries. To remedy this, the second stage of diarization, often called re-segmentation, can be added [25]. In re-segmentation, the results of the clustering are used to initialize a frame-level diarization system that then iterates to refine the boundaries of the speaker turns. Most re-segmentation is performed in the acoustic feature space with a Hidden Markov Model (HMM) or by using a neural network. In this work, we don't use re-segmentation after clustering.

The diarization system proposed here combines the two features i.e neural network embeddings obtained from a TDNN based speaker recognition model [23] and the TDOA features [1]. The new feature is a concatenation of the TDOA feature and extracted embedding (known as an x-vector feature), as in the block diagram in Figure 4.1.

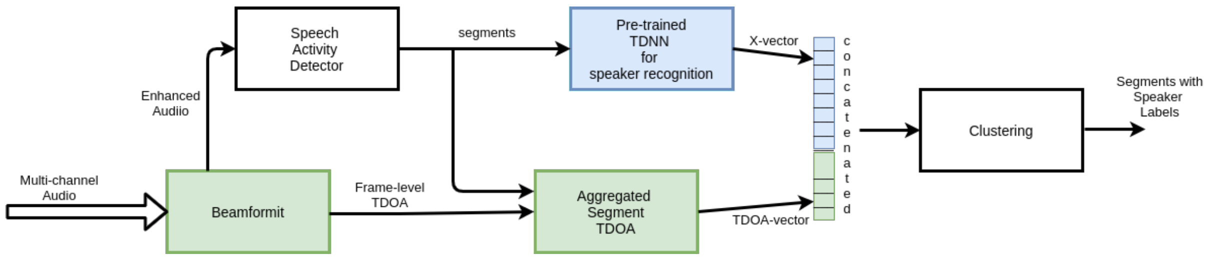


Figure 4.1: Proposed Diarization System

4.1 Proposed Diarization System

The proposed system combines the spatial information and speaker information obtained from the multi-channel audio for the task of diarization. The following sections discuss each of the diarization step, i.e segmentation, the method of combining features and clustering.

4.1.1 Speech Activity Detection (SAD)

As a first step in the diarization process, non-speech frames are identified and removed. Speech activity detector (SAD) outputs segments with start and end time in seconds. Each segment is seen to contain only one speaker, for the audio of no overlapping speakers. The multi-channel audio is first enhanced using the techniques discussed earlier and this enhanced output is given to the SAD. It is a TDNN+LSTM model trained with the CHiME-6 data with the alignments obtained by a GMM [26]. The advantage of using a TDNN network is, it can learn a wider context, wherein this case each layer doubles the width, with a total duration context of +- 320ms for each frame-level decision. The decoded SAD output (every frame level decision is softmax probability over silence/speech) gives the start and end time of each segment. Say the segment obtained for the audio signal are represented by set $S = \{s_1,\ s_1, \dots s_N\}$, for the obtained N segments. The next stage of feature extraction followed by clustering is done per segment.

4.1.2 X-vector Embeddings

To obtain speaker characteristic information, we use the x-vector neural network embedding as the diarization feature [27]. It is a pre-trained speaker recognition model on Voxceleb1 data and the output of the before classification layer is used as this embedding. The input is extracted MFCC features which are 24-dimensional filter banks with a frame-length of 25ms, mean-normalized over a sliding window of up to 3 seconds, and are then fed as input to TDNN network as shown in the Table 4.1. For each segment in S, x-vector set $X = \{\mathbf{x}_{s_1},\ \mathbf{x}_{s_2}, \dots \mathbf{x}_{s_N}\}$ is obtained.

Layer	Layer context	Total context	Input x Output
frame 1	[t-2,t+2]	5	120 x 512
frame 2	{t-2,t+2}	9	1536 x 512
frame 3	{t-3,t+3}	15	1536 x 512
frame 4	{t}	15	512 x 512
frame 5	{t}	15	512 x 1500
stats pooling	[0,T}	T	1500T x 3000
segment 6	{0}	T	3000 x 128
segment 7	{0}	T	128 x 128
softmax	{0}	T	128 x N

Table 4.1: Architecture of TDNN based speaker recognition model used in [27]

4.1.3 TDOA Features

TDOAs are obtained for each analysis frame of the input multi-channel audio using GCC-PHAT between pairs of microphones with one of them as a reference (ref). If there are $'m'$ channels, then a m-1 TDOA valued vector is formed for every analysis frame TD$_{ref,j}$(t) using all other microphones j= 1:M, j≠ref and then used as a feature vector for diarization. One problem in this TDOA estimation using GCC-PHAT lies in the peak picking of the cross-correlation, where the index of the peak is chosen for the TDOA value. With room reverberation, the phase of the source signal is affected and hence the peak will not always correspond to the true delay between the microphone pair. Also, it is sometimes observed that as the analysis window changes, the TDOA value is seen to vary even though the source position has not changed. This poses a continuity problem in the TDOA values obtained for the segment where the speaker is fixed. These problems are addressed by Beamformit which (1) Computes N-best TDOA values for N highest peaks for each analysis frame and (2) Smoothing of the varying nature of these TDOA values computed for successive analysis frames by finding the least cost path which is a two-pass Viberti TDOA refinement; one across the individual channel, and another pass across all channels as in Fig 2.1. For the final best path, frame-wise TDOA values of dimension

M-1, one for each channel, are taken for the entire segment of the SAD output, and then maximum occurring TDOA value is computed across all frames for each channel. Thus a single segment is represented by an aggregated TDOA value given by:

$$agTD_j(s) = mode_t(TD(_j(t))) \tag{4.1}$$

for every microphone $j = 1 : M, j \neq ref$. Thus the vector **agTD**(s$_i$) of length M-1, is the TDOA feature vector used for diarization.

4.1.4 Feature Combination and Clustering

The idea here is to use the complementary information between the x-vector and TDOA, to aid in diarization. Since there is no direct relation between the speaker characteristic embeddings and the spatial TDOA feature from Beamformit (referred as TDOA-B here) vector, a way to capture complementary information from both these feature types is by combining both the feature vectors into to a single vector. The obtained segment level x-vector x$_{s_i}$ and TDOA-B vector agTD$_{s_i}$ are combined to form a xTDOA concatenated vector :

$$\mathbf{xTD}_{s_i} = [\mathbf{x}_{s_i} \quad \mathbf{agTD}_{s_i}] \tag{4.2}$$

for all i in segment set S. Each dimension of x-vector is mean-standard deviation normalized across the segments in the entire conversation and then concatenated with the corresponding TDOA-B vector of that segment as shown in eq 4.2. The total dimension of this new vector is 127+M-1.

We investigated two clustering techniques for the xTDOA feature namely K-means which clusters the feature directly (distance-based clustering) and cosine similarity(CS) with agglomerate hierarchical clustering (AHC:for similarity matrix clustering).

For the K-means, the number of clusters are initialized to the number of speakers in the conversation. It was observed that the Euclidean distance between the features was best in terms of classifying the features. CS metric distinguishes features based on the angular separation between them rather than the absolute distance, as given:

$$cos_{s_i,s_j}(\theta) = \frac{\mathbf{xTD}_{s_i}.\mathbf{xTD}_{s_j}^T}{||\mathbf{xTD}_{s_i}||\mathbf{xTD}_{s_j}||} \tag{4.3}$$

for segment pair {i,j} and then a similarity matrix is constructed, for every segment pair in the conversation (for N segments, a $N \times N$ matrix is formed). The elements of this matrix are then clustered using AHC.

4.2 Diarization Evaluations

4.2.1 Evaluation metric

Speaker diarization systems are usually evaluated by the diarization error rate (DER). DER consists of three components: false alarm (FA) duration, missed detection (Miss) duration, and speaker confusion (SC) duration, among which FA and Miss are caused by SAD errors.

$$\text{DER} = (\text{FA}+ \text{ Miss}+ \text{ SC})/\text{Total reference duration} \tag{4.4}$$

Since a SAD is employed in our implementation and the output of this SAD is an oracle segmentation output, we exclude FA and Miss from our evaluations. Therefore, we report Speaker error rate (SER) as given:

$$\text{SER} = \text{SC}/\text{Total segment durations detected by SAD} \tag{4.5}$$

4.2.2 Experiments

As per Fig 4.1, multi-channel enhanced audio (using Beamformit here) is used to obtain segments by the SAD. A minimum speech duration of 300ms and minimum silence of 30ms are set while decoding the SAD out-put. 40-dimensional MFCCs are obtained for every 250ms frame with 10ms shift and are fed as input to the pre-trained SAD model. Beam search is done on the softmax output to obtain segments of speech with start and end-time. For obtaining TDOA-B feature, Beamformit is run on the multi-channel audio to obtain the per frame TDOA for a frame duration of 500ms with 250ms hop. The TDOA vector obtained from Beamformit for every frame of 500ms is aggregated over the segment as in eq 4.1. This feature is investigated for diarization referring it as TDOA-B. Also, the enhanced audio from Beamformit with the SAD segment output is used to obtain the x-vector using the speaker-recognition model mentioned in Table 4.1. Concatenation of this segment level x-vector and TDOA is done for the xTDOA method as in eq 4.2. We observe complementary segment level information between the two methods: x-vector and TDOA-B to give an idea of how much improvement over we can observe, as depicted in Table 4.2 after combining the feature types.For each correctly classified/misclassified x-vector/TDOA method, time duration $T(hrs) = \sum n_j t_j$ for n$_j$= 0,1 and t$_j$ duration of the segment.

Total duration of misclassified segments by x-vector method (0.31 + 1.69 hrs) is more than that misclassified by TDOA (0.31 + 0.91 hrs),(the best being being TDOA method with total misclassification of 0.31 + 0.91 = 1.22 hrs). Here we use the best performing TDOA, x-vector, and TDOA + x-vector method for the analysis. The total duration of misclassified segments by x-vector method (0.31 + 1.69 hrs = 2 hrs) is more than that misclassified by TDOA (the best being TDOA method with total misclassification of 1.2 hrs). With the xTDOA method, the total misclassification duration was observed to be *0.84* hrs which is an improvement over the segments which were misclassified.

Table 4.2: The total duration of correctly classified/misclassified segments (w.r.t duration) for x-vector and TDOA-B method evaluated on the same 8.5 hrs of multi-channel data. (total duration of speech segments was 7.2 hrs)

X-vector / TDOA-B	Misclassified	Correctly Classified
Misclassified	0.31	0.91
Correctly Classified	1.69	5.14

We compare the xTDOA diarization feature system with two individual baselines of x-vector and TDOA-B in Table 4.3. The clustering baseline method is CS with AHC while the x-vevctor baseline is probabilistic linear discriminant analysis (PLDA) [24] with AHC. x-vector: method showed an average SER of 21.2%. For near and far speaker placements, the error does not change much (maybe obvious as the x-vector diarization system is independent of spatial information). For higher t60, the error is more than that of lower t60. Also for some scenarios, more 5 speaker conversation has slightly more error than 3 speakers.

TDOA-B: With an average SER of 16.2%. For low t60(300ms), on average SER's

Table 4.3: SER in % for a closed conversation data with using Dataset B, evaluated for x-vector, TDOA-B(Beamformit) and xTDOA diarization method

T60	SNR(dB)	No. of Spks	Spk placement	SER(%)		
				x-vector	TDOA-B	xTDOA
300ms	15	3	Near	20.4	18.1	11.2
			Far	20.1	16.9	10.1
		5	Near	22.9	17.1	12.1
			Far	22.0	15.5	11.2
	20	3	Near	17.2	17.2	10.9
			Far	17.0	15.6	**9.0**
		5	Near	18.7	17.5	11.7
			Far	18.4	16.1	10.4
600ms	15	3	Near	22.9	20.9	14.2
			Far	22.6	19.3	12.4
		5	Near	23.2	20.1	13.3
			Far	23.7	19.8	12.2
	20	3	Near	18.9	18.7	12.1
			Far	19.2	18.2	**12.0**
		5	Near	20.3	17.2	13.6
			Far	20.6	16.7	12.4

are comparatively lower as compared to 600ms. Some improvement is also seen for 20dB SNR as compared to 15dB. For far cases, there is around 1.5-2% improvement. xTDOA: For low T60(300ms), SER's are comparatively lower as compared to higher T60(600ms) as seen from Table 4.3. As compared to the previous TDOA only method, SER for a conversation with 3 and 5 speakers is roughly the same. Also as the distance of the speaker increases, an average improvement of 1.5% to 2% is seen SER reduction is noticed. Also as obvious, 20dB SNR showed lower SER as compared to 15dB.

SER for these individual feature methods namely the TDOA-B, x-vector, and the combined xTDOA for various clustering techniques are reported in Table 4.4. We also compare the baseline x-vector PLDA clustering as in [23]. It is seen that K-means means does fairly well in clustering TDOA-B as compared to other metrics like CS+AHC, while xTDOA method shows more improvement with CS+AHC.

Table 4.4: Comparison of the TDOA-B method, x-vector method and the combined xTDOA method for various clustering techniques

Diarization method	Clustering method	SER
TDOA-B	K-means	15.6
	CS + AHC	16.7
X-vector	PLDA + AHC	21.6
	CS + AHC	23.6
xTDOA	K-means	16.2
	CS + AHC	10.7

Chapter 5

User Interface for Analysis and Synthesis

5.1 Multi-channel Audio to Text Tool

This tool [28] is an implementation of the complete enhancement-diarization-ASR system. It combines the techniques investigated for each sub-block namely the multi-channel enhancement and the diarization, discussed in earlier chapters.

It is used to generate speaker labeled transcripts using an input multi-channel audio. It is built for Linux systems along with dependencies like Octave/MATLAB, Python, and Kaldi toolkit used for diarization/ASR back-end. It is a shell script that can be called from the UNIX terminal. It has a configuration file imported in the script specifying the arguments that we can use while running the script.

5.1.1 Using the tool

The syntax of the command while running is

`./multictext.sh <input-file> <number-of-speakers> <output-directory>`.

`<input-file>`: The path of the input multi-channel file.

`<number-of-speakers>`: Number of speakers in the conversation.

`<output-directory>`: where the output transcripts are stored.

The details of how to set up the tool and run are given in [28]. The configuration file specifies the type of enhancement, diarization method and ASR model to be used while calling the setup. Table 5.1 shows the arguments which can be changed inside the configuration file. The options and their actions are part of the tool.

Table 5.1: Configuration file details for the tool

Option	Description	Values
-denoise	Does signal-channel denoising on the multi-channel audio	wiener, spec-sub
-dereverb	Does single-channel dereverberation of the multi-audio or the denoised audio if denoising is specified	wpe, nmf
-localize	GCC based localization to compute the time difference of arrival (TDOA) used as steering vector for beamforming	gcc_phat, gcc_scot
-beamform	Does multi-channel enhancement. Does it on the enhanced audio using single-channel denoising and/or dereverberation, if specified	dsb, mvdr_ta, mvdr_nn, gev_ta, gev_nn
-diarize	The type of diarization system used	xvector, tdoa, xtdoa
-model_dir	Please give the relative model path	eg exp/chain_cleaned/ tdnn_sp_1a
-graph_dir	Please give the relative graph directory path	eg exp/chain_cleaned/ tdnn_sp_1a/graph

Below is the description of the actions of the methods under each argument specified in the configuration file:

- *Single channel dereverberation :* If specified, it does single-channel dereverberation on the multi-channel audio (hence precedes the beamforming stage). The motivation of doing so is dereverberating each channel was observed to give reliable TDOA estimates further improving beamforming performance. The enhanced multi-audio is stored temporarily in the folder.

- *Single channel denoising :* If specified, it does single-channel denoising of the multi-channel audio. Similar to dereverberation, it precedes beamforming but unlike WPE dereverberation, it is not seen to significantly improve the beamforming output. The enhanced multi-audio is stored temporarily at folder.

- *Seq (Sequence)*: Specify the sequence for single-channel enhancement. nr: denoising followed by dereverberation; rn: dereverberation followed by denoising.

- *Beamform :* Type of beamforming method used. The multi-channel audio is either used directly or enhanced output of the previous single-channel stages is used as input for beamforming. Here the beamforming using the output from the localization stage. The output enhanced audio is stored at the folder.

- *Localization :* Specifies the type of GCC localization method for TDOA estimation used for beamforming. If the ′Beamformit′ type of beamforming is used, then it implicitly computes the TDOA estimates.

- *Diarize :* Specifies the diarization feature used. First the SAD produces the segments using the enhanced audio and the segmented audio is for diarization type feature extraction. If *tdoa* or *xtdoa* is the option value, then ′Beamformit′ is used TDOA feature extraction. If diarize is *xvector* then it uses the x-vector speaker recognition model. The output labels per segment are stored at <output-directory>specified when running the script.

- Specifies the path of the acoustic model and graph model to be used while decoding. The models are placed inside the 'exp' folder. The model directory structure should be in the formats in which Kaldi requires.

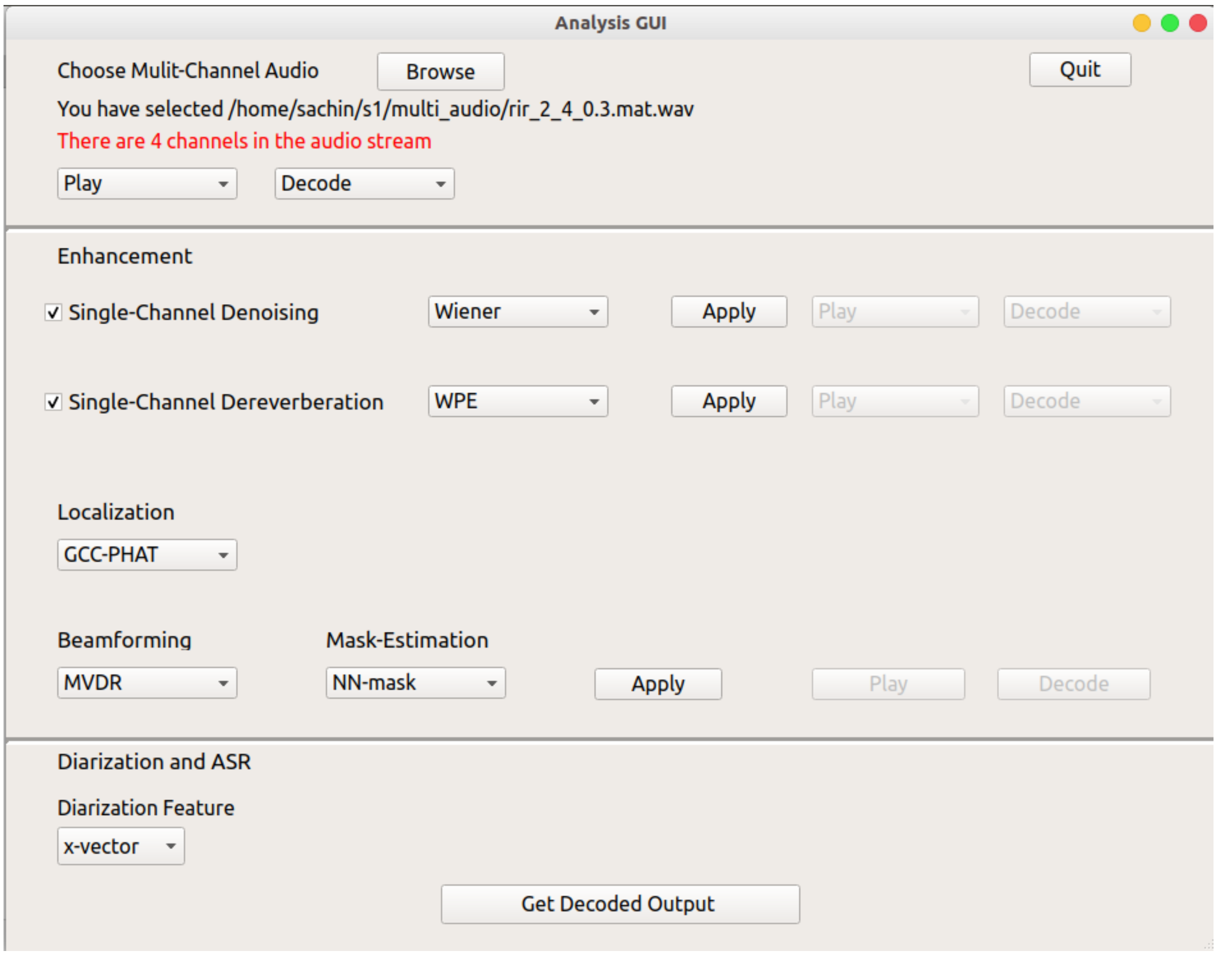
All the folder paths are relative with *<kaldi-root>/egs/toolasr/dia*.

An output produced by the tool for a sample audio file is displayed as

```
SPEAKER 1 : listen is anything going to happen here so what are we supposed to be
doing here
SPEAKER 2 : robin has lords into mighty he should short here shortly we can directly
start without the director person
SPEAKER 3 : speaker three maybe you say a few words for introduction things which
reader of aint i dot you you dot future already know
SPEAKER 2 : this meeting is called to order second that
SPEAKER 3 : i am speaker three i am the incorporator of this city
SPEAKER 1 : right now we have a problem the name of this place is you university
new newtork academy on wednesday i went to the university of state of taxes and was
told that we could not incorporate with the name college or university without beginning
existence for two years
SPEAKER 3 : the problem is that what we should we do what we can to incorporate
outside of taxes
```

5.2 Analysis Graphical User Interface

This is a GUI version of the previous script based tool. This Python based GUI uses the same back-end that the setup uses (link at [28]). Like the script based tool, the GUI has two main stages: enhancement and diarization/ASR. Each of the intermediate stages in Figure 1.3 can be run from the interface. Also, this interface will enable decoding at each intermediate enhancement stage to show the decoded text.



Choose Mulit-Channel Audio

You have selected /home/sachin/s1/multi_audio/rir_2_4_0.3.mat.wav

There are 4 channels in the audio stream

☑ Single-Channel Denoising: Wiener

☑ Single-Channel Dereverberation: WPE

Localization: GCC-PHAT

Beamforming: MVDR

Mask-Estimation: NN-mask

Diarization Feature: x-vector

Figure 5.1: GUI window with each step of the system block for analysis

The interface has 3 section:

- Choosing multi-channel file
- Running Enhancements
- Running ASR

5.2.1 Choosing the multi-channel file

The multi-channel file that can be chosen from the sample folder $'multi_{a}udio'$ in the $PWD or any other multi-channel audio. After clicking the *Browse* button shown in Fig 5.2, a dialog box will open to choose the multi-channel file. Choose the multi-channel file you wish to decode and click *Open*.

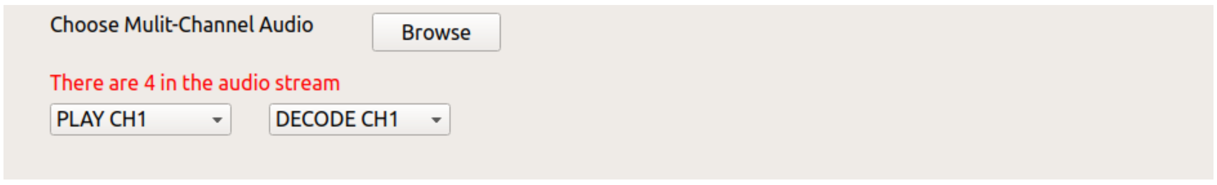


Figure 5.2: Function 1 of GUI

A button to plot a selected RIR is displayed which opens a new window similar to one shown in Figure 5.6. There is an option to play a particular channel of the file selected which will open a media player (by default VLC media player, if installed). Another option is decoding can be done on one particular channel which opens a text output of the channel audio in e new window.

5.2.2 Enhancements

It starts with single-channel de-noising, single-channel dereverberation, DoA estimation, and beamforming shown in Fig 5.3.

Starting with single-channel de-noising, the drop-down list has two techniques namely: {Weiner and Spectral Subtraction}. You can similarly play and decode each single-channel file the same as that for input multi-channel files as mentioned before. There is an option for running single-channel de-noising and can be toggled with a checkbox.

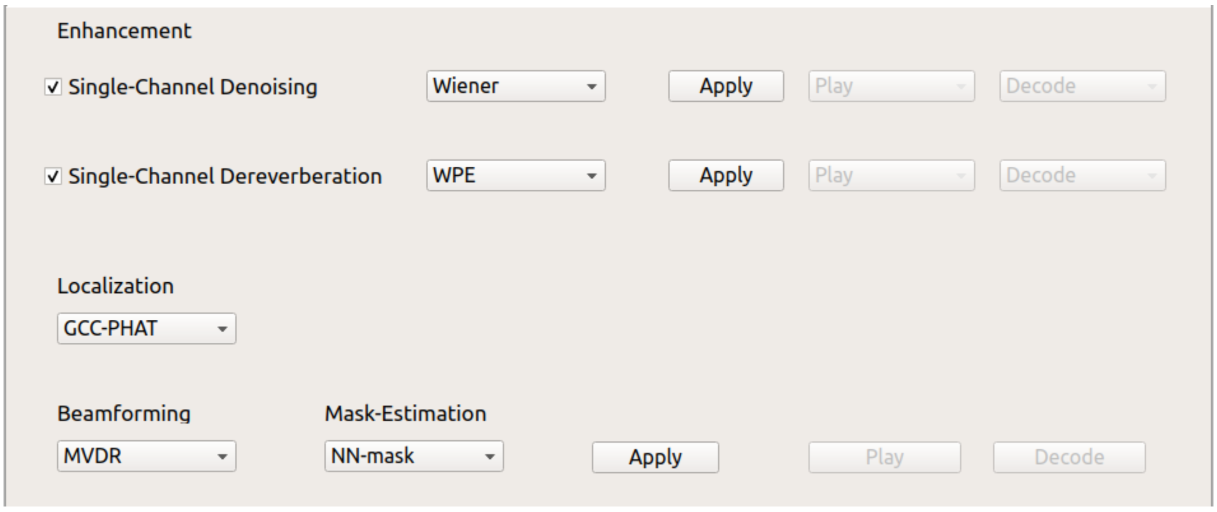


Figure 5.3: Function 2 of GUI

This is succeeded by single-channel dereverberation if it is enabled. For dereverberation, we have used {WPE and NMF}, appearing in the drop-down. We can also choose to play or decode each of the dereverberated channels.

This stage is followed by localization with GCC-{PHAT,SCOT} available as options. Beamforming using the TDOA estimates obtained from the localization(expect for Beamformit Tool). Options available in beamforming are: {DSB, GEV, MVDR, NN-GEV, NN-MVDR}. Finally, after beamforming, a single-channel enhanced file obtained can be played or decoded.
Note that after clicking on the 'Enhance' button only, the enhancement will run. If single-channel denoising/dereverberation is selected, then and only then they will be used to enhance each channel before feeding it to source localization and beamforming (Although both functionality for denoising and then doing dereverberation is an option in the GUI, just doing dereverberation is seen to have improvements in beamforming and TDOA estimation but doing denoising before dereverberation is seen to degrade the ASR performance after beamforming)

5.2.3 Diarization and ASR

This stage uses the enhanced output from the beamforming from the chosen method used in the previous stage to do diarization and ASR. The drop-down in Fig 5.4 lets you choose the feature used for diarization, i.e x-vector, TDOA, or xTDOA method before clustering and after click 'Get Decoded Output' will generate the conversation text using the speaker labels generated from diarization and ASR decoded output from ASR system back-end. After clicking on 'Get Decoded Output', a new dialog box will open to showing the decoded text conversation. The format of the output is the same as the one generated by the script-based tool.

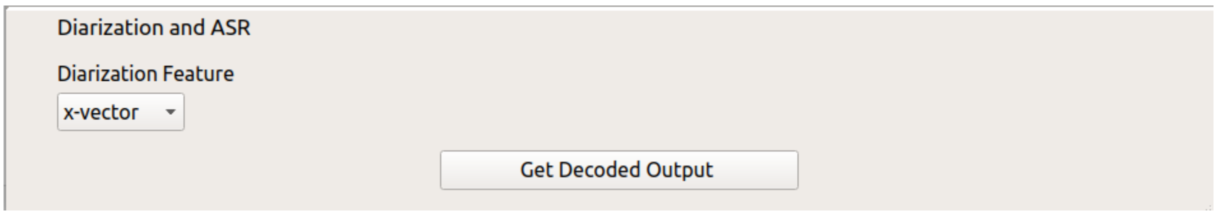


Figure 5.4: Section 2 of GUI

5.2.4 Directory Structure for Files

Linux machine is required to run this task as most of the ASR models are being tested run on the Linux machine.

The following toolkits are required:

- Kaldi
- Octave-dev
- Python 3

The folder from [28] to be placed in `<Your kaldi path >/egs` folder. The log of each stage is displayed in the 'log' console window.

Other Paths

After de-noising, the enhanced file is stored at `./de-noised` with
`<file name>_<method>_<channel no>.wav` as file name.
After dereverberation, the enhanced file is stored at `./dereverb` with
`<file name>_<method>_<channel no>.wav` as file name.
After Beamforming, the enhanced file is stored at `./outbeamform` with
`<file name>_<method>.wav` as file name.
After diarization, the speaker label file and the conversation text is stored respectively as
`<output-dir>/<file name>_<method>_labels`
`<output-dir>/<file name>_<method>_txt`

5.3 Synthesis Interface

This GUI is useful for generating multi-channel audio by selecting room configuration (speaker and microphone array position, room dimensions, T60) (link in [29]). It has two stages as shown in Figure 5.5. The first stage is where we can build a configuration file for generating RIR for our analysis by varying all the allowing parameters. The second stage uses the configuration file generated and then using the source files generates the multi-channel data.

5.3.1 Generate RIR

The RIR generation is based on Habets RIR [8]. It uses the image method, proposed by Allen and Berkley in 1979, is probably one of the most commonly used methods in

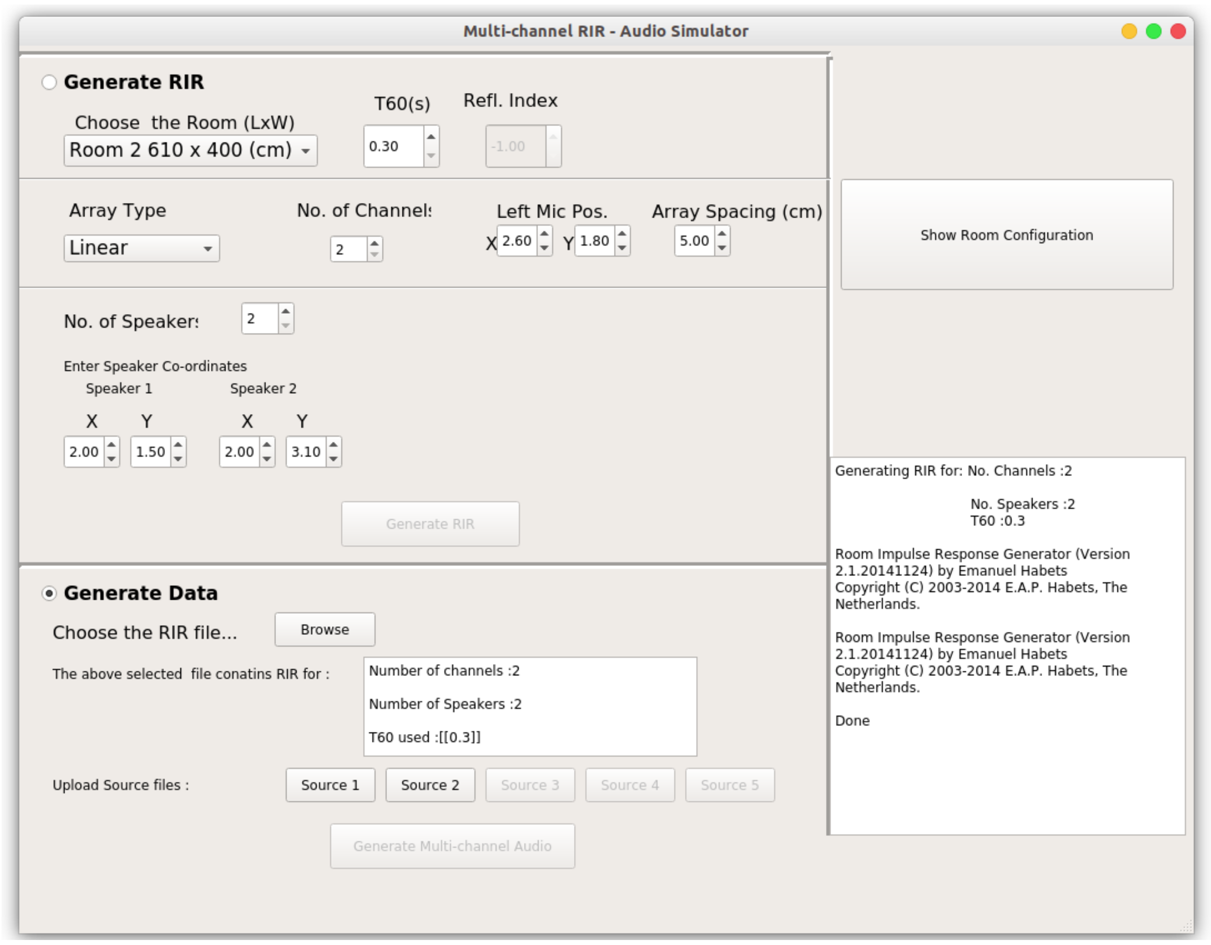
the acoustic signal processing community to create synthetic room impulse responses. A mex-function, which can be used in MATLAB, was developed to generate multi-channel room impulse responses using the image method. This function enables the user to control the reflection order, room dimension, and microphone directivity.
The room configuration used here is almost the same as that of TCS meeting rooms. The parameters that can be varied are :

- Type of room
- Reverberation time constant (T60)
- Type of microphone array (Linear, Circular)
- Number of microphone-channels
- Number of speakers
- Positions of speakers and the array center

The positions of the speaker and array center have to be specified by inputting the X and Y coordinates in meters with rooms one corner fixed as origin whereas array spacing is specified in centimeters. The 'Generate RIR' button will build the RIR mat file. The log window (one on the right with white background) shows the status of the process.

5.3.2 Generate Multi-channel data

The task here is taking the already generated RIR mat file and convolve in the time domain with the source file(s) to generate the multi-channel data. The source files have to be single-channel files of the single speaker of any duration.



Choose the Room (LxW): Room 2 610 x 400 (cm)
T60(s): 0.30
Refl. Index: -1.00
Array Type: Linear
No. of Channels: 2
Left Mic Pos. X 2.60 Y 1.80
Array Spacing (cm): 5.00
No. of Speakers: 2

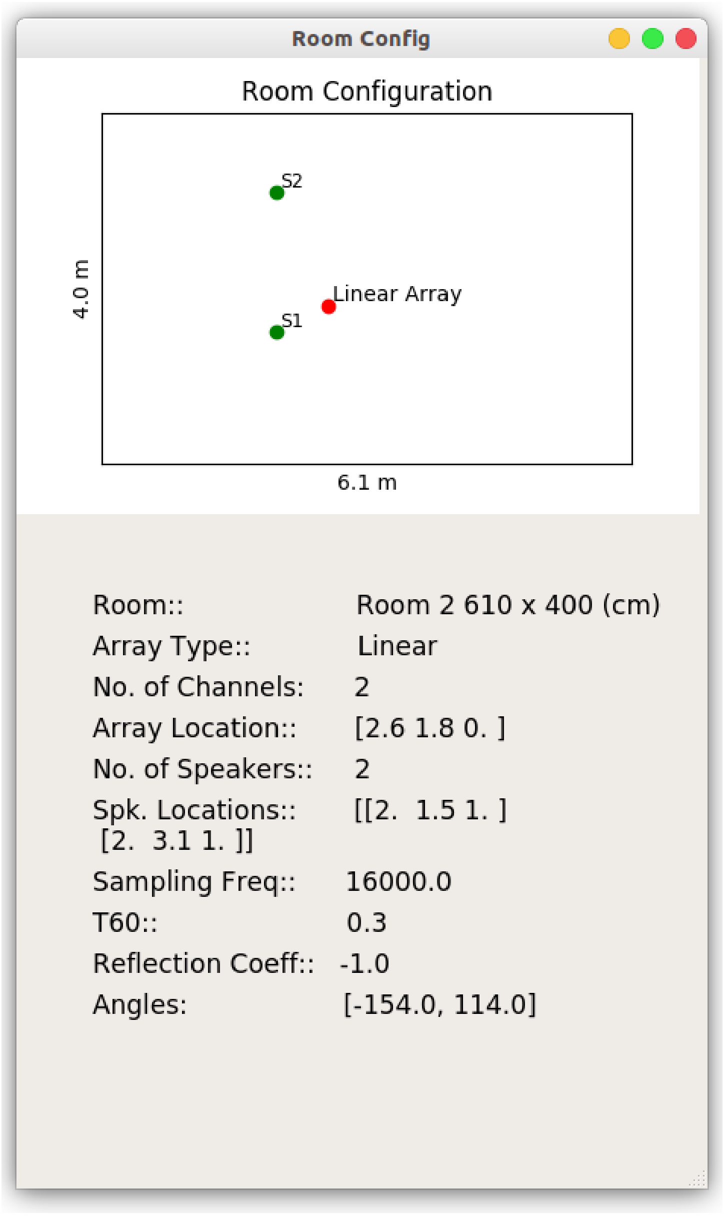
	Speaker 1		Speaker 2	
	X	Y	X	Y
	2.00	1.50	2.00	3.10

Choose the RIR file...

The above selected file conatins RIR for :
Number of channels :2
Number of Speakers :2
T60 used :[[0.3]]

```
Generating RIR for: No. Channels :2

                    No. Speakers :2
                    T60 :0.3

Room Impulse Response Generator (Version
2.1.20141124) by Emanuel Habets
Copyright (C) 2003-2014 E.A.P. Habets, The
Netherlands.

Room Impulse Response Generator (Version
2.1.20141124) by Emanuel Habets
Copyright (C) 2003-2014 E.A.P. Habets, The
Netherlands.

Done
```

Figure 5.5: Multi-channel RIR generator used to generate RIR and multi-channel data using generated RIR



```
Room::                  Room 2 610 x 400 (cm)
Array Type::            Linear
No. of Channels:        2
Array Location::        [2.6 1.8 0. ]
No. of Speakers::       2
Spk. Locations::        [[2.  1.5 1. ]
 [2.  3.1 1. ]]
Sampling Freq::         16000.0
T60::                   0.3
Reflection Coeff::      -1.0
Angles:                 [-154.0, 114.0]
```

Figure 5.6: Room configuration of the simulated RIR

The 'Generate Multi-channel Data' button will generate the multi-channel file while the button 'Show Room Configuration'(enabled after choosing the RIR mat file) will open a window as shown in Figure 5.6 showing the RIR details like locations of speaker, array and room dimensions. Also, other parameters set while generating RIR will be listed.

5.3.3 Setup, Directory Structure and Other requirements

This GUI run on Windows and Linux with the required packages installed. The software along with the `README` is found at [29] The following software packages are required to be installed to run the GUI's:

- Python 3 or >3
- Octave-cli , Octave-dev
- PyQt5, soundfile for Python3

The directory 'rir' contains the generated RIR files, which are .mat file contain all the information of the parameters which one has chosen during RIR generation while the directory 'multi-channel' contains the generated multi-channel files. Here the audio files stored are in '.wav' format. This folder also contains the transcripts for each multi-channel files with text for each source speaker file if it's available while generating multi-channel data.

The command-line tool along with Analysis and Synthesis GUIs can be used to experiment with and analyze algorithms developed for the purpose of multi-channel audio enhancement and diarization.

Chapter 6

ASR Experiments

ASR is a technology that enables the recognition of spoken language into a textual representation by computers. It can now be found in a large variety of consumer electronics from cars to mobile phones. These technologies often rely on statistical models like Hidden Markov Models although state-of-the-art ASR systems use end-to-end neural networks (use audio in the time domain to obtain the text) with the additional need of large data to train these models. The audio to be decoded by the ASR is typically first enhanced and then MFCC features are extracted from this audio. MFCC is a feature widely used in automatic speech and speaker recognition. They were introduced by Davis and Mermelstein in the 1980s, and have been state-of-the-art ever since. MFCCs are obtained for every frame of the audio (overlapping frames are used). Let **O** sequence of MFCC features corresponding to a speech signal. That is, O = $\{\mathbf{o}_1, \mathbf{o}_2, \mathbf{o}_3,..\mathbf{o}_T\}$, where $\mathbf{o}_i$ refers to a d-dimensional acoustic feature vector and T is the length of the sequence or the total number of frames, for i=1,2..T. If W one of all the possible word sequences, then the problem the automatic speech recognition(ASR) decoder solves is to find the most likely word sequence W* as given below:

$$\mathbf{W}^* = argmax_W(P(\mathbf{O}|\mathbf{W})P(\mathbf{W})) \tag{6.1}$$

Here P(**O**|**W**) is modelled by acoustic model and P(**W**) by the language model.

6.1 Acoustic Modelling

The acoustic model learns a mapping between the acoustic or observation vectors to a unique phone which is a fundamental unit of acoustic speech. In conventional ASR systems, they are modeled by Hidden Markov Model (HMM). The mappings are related to the parameters of this HMM model, which are the transition and observation probabilities of the HMM states.

Hidden Markov models are used to model the acoustic observations (feature vectors) at the sub-word level, such as phonemes. It is typical for each phoneme to be modeled with 3 states, to separately model the beginning, middle, and end of the phoneme. Each state has a self-transition and a transition to the next state. Each state in the HMM had a probability distribution defined by a Gaussian Mixture Model (GMM) as referred to as the observation probability. The simplest modeling is to model the observation probabilities by continuous distributions like Gaussian distribution, upon which the model is called GMM-HMM [30].

Another way of modeling the observation probabilities is to use some other model's parameters to learn these distributions than using GMMs. In-state of the art acoustic models, DNN/TDNN [31] [32] or RNN [33] based neural nets are used to model these observation probabilities. These models are trained using i-vectors and not MFCCs since these i-vectors try to capture speaker characteristics better. Such acoustic models are called "hybrid" systems or DNN-HMM systems to reflect the fact that the observation probability estimation formerly done by GMMs is now done by a DNN, but the rest of the HMM framework, in particular the HMM state topologies and transition probabilities, are still used.

6.2 Hybrid Acoustic models

The simplest and most common neural network used for acoustic modeling is the conventional fully connected feed-forward neural network shown in Fig 6.1. Although we are training a DNN-HMM to predict the label for each frame of input, it is very beneficial for classification to provide a context window of frames to the network as input which a time-delay neural network (TDNN) [32] utilizes. The neural network acoustic models compute posterior probabilities p$(s|x_t)$ over phoneme labels(s). These state-level posterior probabilities must be converted to state likelihoods p($x_t|s$) for decoding using an HMM. This can be done by an application of Bayes' rule:

$$p(x_t|s) = \frac{p(s|x_t)p(x_t)}{p(s)} \tag{6.2}$$

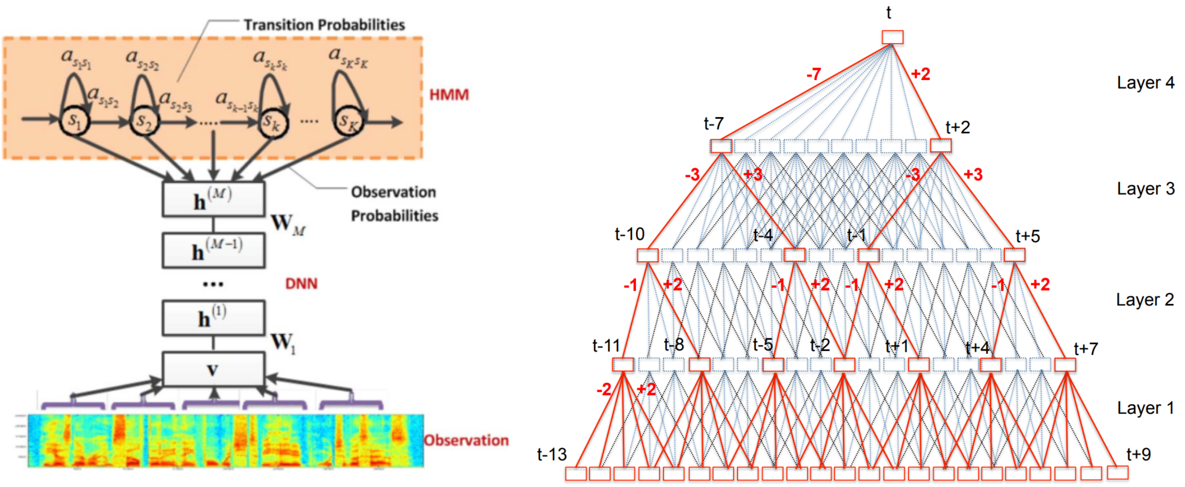


Figure 6.1: Figure on the left is DNN-HMM [31] and on the right is TDNN-HMM [32] . The input to these models is acoustic features like MFCC and the label is the stage of the acoustic feature provided from the GMM-HMM alignments, for this reason, they are called 'hybrid' models

Note that because the prior over the observations p(x_t) is constant over all HMM phoneme states s, it contributes a constant factor to all likelihood scores so it can be dropped. Thus, the likelihood p($x_t|s$) is computed by dividing the network's posterior probabilities by state probability. This state prior probability p(s) can be easily estimated by counting the occurrences of each phoneme state in the training data.

6.3 Language Model

Models that assign probabilities to sequences of words are called language models or LMs. We use the simplest model that assigns probabilities LM to sentences and sequences of words, the n-gram. An n-gram is a sequence of N n-gram words: a 2-gram (or bigram) is a two-word sequence of words like “Give your”, “your mobile”, or ”mobile phone”, and a 3-gram (or trigram) is a three-word sequence of words like “Give your mobile”, or “your mobile phone”.

We compute probabilities of entire sequences P(w_1,w_2,...,w_n) using the chain rule of probabilities as :

$$P(w_1^n) = P(w_1)P(w_2|w_1)P(w_3|w_1^2)P(w_n|w_1^{n-1}) \tag{6.3}$$

where w_1^j is a sequence { w_1, w_2,..w_j }. For a 2-gram LM, the context sequence is of length 1, i.e. only depends of the previous work given by P(w_j|w_{j-1}) and for a 3-gram its P(w_j|w_{j-1}, w_{j-2}).

6.4 Evaluations

Model Description: Acoustic models based on GMM-HMM and neural network hybrid models were used for evaluating ASR performance. The acoustic model was trained on simulated multi-channel data (A total of 20hrs of audio) chosen from $'100 hours of train'$ LibriSpeech audio. Following are the different versions of GMM-HMM, DNN-HMM, and TDNN-HMM models acoustic models trained on different acoustic conditions:

1. Trained on clean data (original)

2. Trained on simulated multi-channel data (clean + one of the channels)

3. Trained on clean, simulated multi-channel and dereverberated data. (clean + one of the channels+ dereverberated channel)

That is, models (2) and (3) are trained on more than one type of data (multi-condition training). The model training was done using Kaldi TDNN models. For decoding on TCS-set, we use a pre-trained Aspire [34] acoustic model. It is trained on Fisher English that has been augmented with impulse responses and noises to create multi-condition training. The language model used here was a 4-gram model, trained on the joint set of train and test transcripts.

6.4.1 Effects of different enhancements on ASR

Here the evaluations are done on two test sets: Dataset A (single-utterance per speaker conversation set) and TCS dataset as discussed in section 1.2 and 1.3 respectively. We use Dataset A, a 5.1 hour simulated multi-audio as a test-set, for evaluating and comparing the beamforming methods v/s these different multi-conditional trained model types (superscript to model names means the refer to multi-conditions used to train)

Table 6.1: WER in % at different enhancement stages in the enhancement block in Fig 1.3 for different AMs:1-trained on clean, 2-trained on +clean+reverberated, 3-trained on clean+reverberated+dereverberated

Model	Clean	Reverb+Noise	De-noised & De-Reverb	De-Reverb & BF(NN-MVDR)
GMM-HMM^1	7.18	13.45	13.56	**10.46**
GMM-HMM^2	7.19	14.35	13.23	**12.45**
GMM-HMM^3	7.19	15.56	14.25	**11.25**
DNN-HMM^1	7.14	17.13	**12.13**	12.50
DNN-HMM^2	7.34	16.63	**11.13**	12.39
DNN-HMM^3	7.58	12.27	13.13	**10.3**
TDNN-HMM^1	8.34	18.63	13.13	**11.39**
TDNN-HMM^2	9.10	16.56	**10.21**	10.27
TDNN-HMM^3	8.27	14.25	12.5	**9.13**

As observed, NN-MVDR gave a better WER performance considering all types of acoustic models, 2-3% more error than the clean on clean data. Also, it is observed that single-channel ASR on denoised and dereverberation is not helping the ASR as in DNN-HMM and TDNN-HMM models, the WER values are almost same as that of noisy speech WER values (around 1% difference). Models trained on reverb and both clean and reverberated data gave lower WER than that trained on clean data for NN based TDNN and DNN model. Also among all the models tested, TDNN-HMM^3 gave lower WER's, one possible reason being it does a better job looking get wider temporal observation sequence than others and also because of the multi-conditional training of it's AM.

Table 6.2 shows evaluations on different beamforming methods only for TCS-set and Dataset A using the TDNN-HMM^3 AM. First WPE single-channel dereverberation is done on the multi-channel audio and then beamforming is performed. NN-MVDR beamforming showed better performance as compared to other methods. One reason why the WER numbers were TCS dataset on were higher as compared to simulated Dataset A, was because of train-test mismatch as it was decoded using Aspire AM trained on Fisher English recordings, while the test-set TCS records were Indian English.

Table 6.2: WER in % for different beamforming methods using TDNN-HMM[3] AM for Dataset A and Aspire AM for TCS dataset

BF-Method	Dataset A	TCS-set
None	16.1	29.9
DSB	14.2	29.3
Beamformit	14.1	28.1
GEV	13.3	26.7
MVDR	13.2	25.7
NN-GEV	9.8	25.9
NN-MVDR	**9.1**	**25.0**

For the neural network based models, training and decoding was done using Nvidia 1080 Ti with 1 GPU using Kaldi toolkit.

6.4.2 Effects of Diarization on ASR

As discussed in section 6.1.1 where the hybrid models try to model the observation probabilities using a neural network model like DNN/TDNN , the input to these AM models are i-vectors. Normally without speaker labels, i-vectors are extracted per utterance of a speaker considering that every utterance in the recording session belongs to different speakers. A minor difference when we have utterance level speaker labels is that these per utterance speaker label information is used for extracting i-vector (i-vector feature-length normalization is done for all utterances corresponding to a given speaker). This normalization of i-vectors aids using speaker labels aided in better estimation of the observation probabilities. For ASR evaluations using speaker information from diarization, we use Dataset B and TCS-dataset. The multi-channel audio file is first single-channel dereverberated using WPE and then compared for two beamforming techniques: NN-MVDR (best performing) and Beamformit (performance of other b-f methods GEV was roughly same as MVDR, hence only MVDR is illustrated). The enhanced audio is then used by the diarization system: SAD is used to extract segments and feature-based diarization methods like TDOA, x-vector and xTDOA are applied to obtain speaker labels. This speaker information is finally used to do ASR decoding using the respective AMs.

Table 6.3: WER in % for different diarization methods for NN-MVDR and Beamformit beamforming methods decoded using TDNN-HMM^3 AM for Dataset A and Aspire AM for TCS dataset

Enhancement	**Diarization**	**Dataset B**	**TCS-dataset**
NN-MVDR	None	13.4	25.8
	TDOA	10.3	25.9
	x-vector	11.3	26.1
	xTDOA	10.5	25.8
Beamformit	None	15.2	27.8
	TDOA	13.4	27.1
	x-vector	14.3	27.2
	xTDOA	13.9	27.9

We use the best performing TDNN-HMM^3 acoustic model for obtaining ASR performance on simulated data Dataset B and Aspire model to test on TCS-dataset. For diarization, clustering is done using CS + AHC. It is seen that without any diarization, the WER is around 13.4% where there is an improvement after diarization, although there is no significant difference of WER among the types of feature-based diarization methods. Also for the TCS dataset, no significant improvement was actually seen after diarization as compared to 28.1% in Table 6.2. One of the reasons why the diarization output didn't aid much is because the SAD was not able to detect the single speaker segments in the audio.

Chapter 7

Summary and Future Work

This work focused on improving the ASR performance of audio recorded in a meeting type room using a microphone array. Multi-channel enhancement techniques based on adaptive beamforming showed promising results especially neural network mask based beamforming enhancement outperforming conventional beamforming with respect to WER numbers, on both LibriSpeech simulated and TCS natural recordings. Although on the TCS recordings the WER numbers were relatively higher because of mainly train-test mismatch/unavailability of data to train. With the help of multi-condition training, the WER improved using a DNN/TDNN-HMM acoustic model for simulated data where mask based beamforming MVDR showed the best improvement. A part of the work also focused on speaker diarization which aided in the transcription of the meeting session with speaker labels. Diarization feature-based methods such as TDOA-B and the fused xTDOA feature showed lower confusion rates as compared to x-vector system, although near field condition affected TDOA-B diarization system performance for the simulated data. Over the TDOA-B method, xTDOA overall showed atleast 3% lower SER with cosine similarity and agglomerative clustering. With diarization aided speaker labels, there was a slight improvement in WER number (roughly 2%) as compared to no diarization SER number for simulated data.

The performance of the speaker diarization system discussed here strongly depends on the SAD, apart from the feature and the clustering method used. Improving the SAD such that it is robust enough so only the speech regions are correctly detected and silence is suppressed, is one task to focus on, for better diarization. The feature that capture spatial information like the TDOA and the one that captures speaker characteristic like the MFCC or embeddings from speaker recognition models need to be associated in a better way rather than just doing simple concatenation which was proposed in this work. Also, this work was restricted to recordings where there were more than one speaker in a meeting session but not all speakers were active at the same time which is not always the case. Such scenarios might need separating the speakers into individual speaker for speech recognition, which too comes with its own challenges. Thus improving the ASR on such recordings is a challenging task, an example of this can be seen in CHiME6 challenge [35].

References

[1] X. Anguera, C. Wooters, and J. Hernando. Acoustic beamforming for speaker diarization of meetings. *2014 IEEE Spoken Language Technology Workshop (SLT). IEEE,*, page 2011–2022, 2007.

[2] Chen, Szu-Jui, Aswin Shanmugam Subramanian, Hainan Xu, and Shinji Watanabe. Building state-of-the-art distant speech recognition using the CHiME-4 challenge with a setup of speech enhancement baseline. *2014 IEEE Spoken Language Technology Workshop (SLT). IEEE*, page arXiv:1803.10109, 2018.

[3] Benjamin Cauchi and et al. Joint dereverberation and noise reduction using beamforming and a single-channel speech enhancement scheme. *Proc. REVERB challenge workshop. Vol. 1.*, 2014.

[4] McWhirter, J. G., and T.J. Shepherd. An efficient systolic array for MVDR beamforming. *2014 IEEE Spoken Language Technology Workshop (SLT). IEEE,*, pages 11–20, 1988.

[5] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach. NARA-WPE: A python package for weighted prediction error dereverberation in numpy and tensorflow for online and offline processing. pages 1–5, 2018.

[6] Nikhil Mohanan, Rajbabu Velmurugan, and Preeti Rao. Speech dereverberation using NMF with regularized room impulse response. *In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4955–4959, 2017.

[7] Navneet Upadhyay and Abhijit Karmakar. Speech enhancement using spectral subtraction-type algorithms: A comparison and simulation study. *Procedia Computer Science*, 54:574 – 584, 2015.

[8] Allen, Jont B., and David A. Berkley. Image method for efficiently simulating small-room acoustics. *The Journal of the Acoustical Society of America 65.4 (1979)*, pages 943–950, 1979.

[9] Panayotov, Vassil, and et al. Librispeech: an ASR corpus based on public domain audio books. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[10] Do, Hoang, Harvey F. Silverman, and Ying Yu. A real-time SRP-PHAT source location implementation using stochastic region contraction (SRC) on a large-aperture microphone array. *In 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, vol. 1,*, pages I–121, 2007.

[11] M. Azaria and D. Hertz. Time delay estimation by generalized cross correlation methods. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):280–285, 1984.

[12] Ronald Mucci. A comparison of efficient beamforming algorithms. *IEEE Transactions on Acoustics, Speech, and Signal Processing 32.3*, pages 548–558, 1984.

[13] E. Warsitz and M. R. Haeb-Umbach. Blind acoustic beamforming based on generalized eigenvalue decomposition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 15(5):1529–1539, July 2007.

[14] Habets, Emanuël AP, and et al. The MVDR beamformer for speech enhancement. *Speech Processing in Modern Communication. Springer, Berlin, Heidelberg*, pages 225–254, 2010.

[15] J. Heymann, L. Drude, and R. Haeb-Umbach. Neural network based spectral mask estimation for acoustic beamforming. pages 196–200, 2016.

[16] Emmanuel Vincent and et al. The 4th CHiME speech separation and recognition challenge. *URL: http://spandh. dcs. shef. ac. uk/chime challenge Last Accessed on 1 August, 2018.*

[17] Gribonval R. Vincent, E. and C. Févotte. Performance measurement in blind audio source separation. *IEEE Trans. Audio, Speech and Language Processing, 14(4)*, pages 1462–1469, 2006.

[18] Gerald Friedland and et al. The ICSI RT-09 speaker diarization system. *IEEE Transactions on Audio, Speech, and Language Processing 20.2*, pages 371–381, 2011.

[19] Araki, Shoko, and et al. A DOA based speaker diarization system for real meetings. *2008 Hands-Free Speech Communication and Microphone Arrays*, 2008.

[20] Xavier Anguera, Chuck Wooters, Jose M. Pardo, and Javier Hernando. Automatic weighting for the combination of tdoa and acoustic features in speaker diarization for meetings. *In 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, pages IV–241, 2007.

[21] Deepu Vijayasenan, Fabio Valente, and Hervé Bourlard. Multistream speaker diarization of meetings recordings beyond MFCC and TDOA feature. *Speech Communication 54.1 (2012)*, pages 55–67, 2012.

[22] José Pardo, Xavier Anguera, and Chuck Wooters. Speaker diarization for multiple distant microphone meetings: Mixing acoustic features and inter-channel time differences. *Ninth International Conference on Spoken Language Processing*, 5, 2006.

[23] Gregory Sell and et al. Diarization is hard: Some experiences and lessons learned for the jhu team in the inaugural dihard challenge. pages 2808–2812, 2018.

[24] Zili Huang, L. Paola García-Perera, Jesús Villalba, Daniel Povey, and Najim Dehak. JHU diarization system description. *In IberSPEECH (pp. 236-239)*, 2018.

[25] Garcia-Romero, Daniel, and et al. Speaker diarization using deep neural network embeddings. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[26] SAD. *Speech activity detector for diarization*, CHiME 6, 2020. `https://kaldi-asr.org/models/12/0012_sad_v1.tar.gz`.

[27] David Snyder, Daniel Garcia-Romero, Sell Gregory, Daniel Povey, and Daniel Khudanpur. X-vectors: Robust DNN embeddings for speaker recognition. *In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5329–5333, 2018.

[28] Sachin Nayak. *Enhancement-Diarize-ASR Tool*, 2020. `https://github.com/iitbdaplab/toolasr`.

[29] Sachin Nayak. *Multi-channel audio simulator*, 2019. `https://github.com/iitbdaplab/multi-channel_audio_simulator`.

[30] Lawrence R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[31] George E. Dahl. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing, 20(1)*, pages 31–42, 2011.

[32] Daniel Povey Peddinti, Vijayaditya, and Sanjeev Khudanpur. A time delay neural network architecture for efficient modeling of long temporal contexts. *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[33] H. Sak, Andrew Senior, and F. Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pages 338–342, 2014.

[34] Vijayaditya Peddinti and et al. JHU ASpIRE system: Robust LVCSR with TDNNs, ivector adaptation and RNN-LMS. *ASRU*, 2015.

[35] Shinji Watanabe, Michael Mandel, Jon Barker, and Emmanuel Vincent. CHiME-6 challenge: Tackling multispeaker speech recognition for unsegmented recordings. *ArXiv*, abs/2004.09249, 2020.

Acknowledgements

I would take this opportunity to thank a few people who have helped me to make this project a success. First and foremost, I would like to thank Prof. Preeti Rao and Prof. Rajbabu Velmurugan for providing me an opportunity to work on this project. Their motivation, guidance and positivity has inspired me throughout this project work. I am especially thankful to Tata Consultancy services (TCS) for providing the funding to support my M.Tech. studies at IIT Bombay. Besides my supervisors, I would like to thank Dr. Sunil Kopparapu and Dr. Ashish Panda from TCS, who in spite of their busy work schedule, spent their valuable time in various discussions, project meetings and other project-related work. I am thankful to my friend Mr. Nikhil M., Ph.D. student of IIT Bombay, for his valuable suggestions and insights in understanding various concepts. I would also like to express special thanks to my friends and all those who directly or indirectly helped me in my project. Finally, I would like to thank to my parents for their constant support and prayers throughout these three years of study.