

IMPLEMENTATION OF A DIGITAL HEARING AID AS A SMARTPHONE APPLICATION

*A dissertation
submitted in partial fulfilment of
the requirements for the degree of*

Master of Technology

by

Saketh Sharma

(14307R030)

under the supervision of

Prof. P. C. Pandey



**Department of Electrical Engineering
Indian Institute of Technology Bombay**

June 2017

Left blank

Indian Institute of Technology Bombay
Department of Electrical Engineering

M. Tech. Dissertation Approval

This dissertation entitled “**Implementation of a Digital Hearing Aid as a Smartphone Application**” by **Saketh Sharma** (Roll No. 14307R030) is approved, after the successful completion of *viva voce* examination, for the award of the degree of **Master of Technology in Electrical Engineering** with specialization in **Electronic Systems**.

Supervisor: <i>P. C. Pandey</i>20/06/2017	(Prof. P. C. Pandey)
Examiners: <i>Preeti Rao</i>	(Prof. Preeti Rao)
 <i>Arun Pande</i>	(Dr. Arun Pande)
Chairman: <i>Preeti Rao</i>	(Prof. Preeti Rao)

Date: 30 June 2017

Place: Mumbai

Left blank

DECLARATION

I declare that this written submission represents my ideas in my own words and where ideas or words are taken from others, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and I have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

saketh sharma

Saketh Sharma

(Roll No. 14307R030)

Date: 30 June 2017

Place: Mumbai

Left blank

ABSTRACT

Persons with sensorineural hearing loss suffer from degraded speech perception caused by frequency-dependent elevation of hearing thresholds, reduced dynamic range, abnormal loudness growth, and increased temporal and spectral masking. Several signal processing techniques are used in hearing aids to alleviate the effects of sensorineural hearing loss and to improve speech perception. As an alternative to existing ASIC-based hearing aids which are expensive to develop, a digital hearing aid is implemented as a smartphone application.

The implementation provides user-configurable processing for (i) background noise suppression and (ii) dynamic range compression and frequency-selective amplification. To reduce the effect of increased masking, signal processing for background noise suppression is incorporated. Noise spectrum estimation using dynamic quantile tracking and speech enhancement using spectral subtraction and geometric approach are implemented to improve speech perception. Enhancement of speech corrupted with different types of additive stationary and non-stationary noise showed improvement in speech quality to be equivalent to an SNR advantage of 3 – 6 dB. To compensate for reduced dynamic range and frequency-dependent elevation of hearing thresholds, a sliding-band dynamic range compression technique is used. It does not introduce perceptible distortions associated with the single-band and multi-band compression techniques. Both processing blocks are implemented for real-time processing using single FFT-based analysis-synthesis.

Implementation as a smartphone application has been carried out using Nexus 5X with Android 7.1 Nougat OS. A touch-controlled graphical user interface enables the user to fine tune the processing parameters in an interactive and real-time mode. The processing delay is approximately 45 ms, making it suitable for face-to-face communication.

Left blank

CONTENTS

ABSTRACT	i
CONTENTS	iii
LIST OF FIGURES	v
LIST OF TABLES	vii
LIST OF SYMBOLS	ix
LIST OF ABBREVIATIONS	xi
1 INTRODUCTION	1
1.1 Overview	1
1.2 Objective	2
1.3 Outline	2
2 SIGNAL PROCESSING FOR HEARING AIDS	3
2.1 Introduction	3
2.2 Signal Processing to Reduce the Effect of Decreased Dynamic Range	3
2.3 Signal Processing to Suppress Background Noise	4
2.4 Signal Processing to Reduce the Effect of Increased Spectral and Temporal Masking	5
2.5 Smartphone as a Hearing Aid Platform	6
2.6 Scope of the Project	7
3 HEARING AID IMPLEMENTATION AS A SMARTPHONE APP	9
3.1 Introduction	9
3.2 Audio I/O Framework	10
3.3 Processing Framework	13
3.4 Graphical User Interface	14
3.5 Test Results	14
3.6 Summary	15
4 SLIDING-BAND DYNAMIC RANGE COMPRESSION	17
4.1 Signal Processing Technique	17
4.2 Implementation Details	20
4.3 Test Results	21
4.4 Summary	25
5 NOISE SUPPRESSION USING DYNAMIC QUANTILE TRACKING	27
5.1 Introduction	27

5.2	Noise Estimation	27
5.3	Speech Enhancement	29
5.4	Implementation	31
5.5	Test Results	32
5.6	Summary	37
6	SUMMARY AND CONCLUSION	39
	APPENDIX A	41
	REFERENCES	43
	ACKNOWLEDGEMENTS	49
	AUTHOR'S RESUME	50

LIST OF FIGURES

Figure 3.1	Implementation of the hearing aid application.	10
Figure 3.2	Audio I/O Framework.	12
Figure 3.3	Screenshot of the homescreen of the app.	14
Figure 3.4	Audio interface to the 4-pin TRRS headset port of the mobile handset.	15
Figure 4.1	Relation between input power (dB) and output power (dB) for i th frame and band centered at k th spectral sample.	18
Figure 4.2	Spectral modification for compensation of increased hearing thresholds and decreased dynamic range using sliding-band dynamic range compression, adapted from [13].	19
Figure 4.3	Screenshot of the settings screen for sliding-band dynamic range compression.	20
Figure 4.4	Example of processing for constant gain and compression: (a) input; constant amplitude tone with frequency linearly swept from 100 Hz to 10 kHz in 30 s, (b) GUI parameters set for constant gain of 12 dB, (c) processed output for constant gain, (d) GUI parameters set for constant gain of 12 dB and compression ratio of 2, and (e) processed output for constant gain and compression	22
Figure 4.5	Example of processing for frequency-dependent gain and compression: (a) input; constant amplitude tone with frequency linearly swept from 100 Hz to 10 kHz in 30 s, (b) GUI parameters set for frequency-dependent gain, (c) processed output for frequency-dependent gain, (d) GUI parameters set for frequency-dependent compression, and (e) processed output for frequency-dependent compression	23
Figure 4.6	Example of processing for dynamic range compression: (a) input; amplitude modulated tone of 1 kHz frequency and (b) processed output for CR = 2.	23
Figure 4.7	Example of processing for dynamic range compression: (a) processing parameters (b) input; amplitude modulated VHSES speech, and (c) processed speech with parameters as shown in (a).	24
Figure 5.1	Estimation of the noise spectral samples using dynamic quantile tracking technique based on range estimation (adapted from [9]).	28
Figure 5.2	Screenshot of the settings screen for noise suppression.	32
Figure 5.3	PESQ scores for the enhanced noisy speech, speech: VHSES, enhancement methods: spectral subtraction and geometric approach, processing parameters: $\alpha = 2$, $\beta = 0.01$, $\gamma = 1$, and $\lambda = 1/256$.	33
Figure 5.4	PESQ scores for the enhanced noisy speech, speech: NOIZEUS, enhancement method: geometric approach, processing parameter: $\lambda = 1/256$.	34
Figure 5.5	Non stationary noise suppression using dynamic quantile tracking based speech enhancement. (a) unprocessed noisy signal with modulated white noise at 0 dB, -6 dB and 0 dB SNR, (b) processed using spectral subtraction, and (c) processed using geometric approach.	36

Figure 5.6 Non stationary noise suppression using dynamic quantile tracking based 36
speech enhancement. (a) unprocessed noisy signal with babble, white and
car noise, (b) processed using spectral subtraction, and (c) processed using
geometric approach

LIST OF TABLES

Table 5.1	SNR improvement (dB) for PESQ score of 2 using spectral subtraction (SS) and geometric approach (GA) for different types of noises from AURORA database; speech: VHSES.	35
Table 5.2	Improvement in PESQ scores by noise suppression using dynamic quantile tracking and geometric approach for different types of noises; speech: 30 sentences from NOIZEUS.	35

Left blank

LIST OF SYMBOLS

Symbols	Explanation
$BW(k)$	Auditory critical bandwidth centered at k th spectral sample
$CR(k)$	Compression ratio at k th spectral sample
$\hat{D}(i,k)$	Noise estimate at k th spectral sample of i th frame
$d_n(i,k)$	Change in n th quantile estimate at k th spectral sample of i th frame
$f(k)$	Frequency in kHz corresponding to k th spectral sample
F_s	Sampling frequency
$G(i,k)$	Gain at k th spectral sample of i th frame
$G_{LdB}(k)$	Gain at linear region (no compression) in dB at k th spectral sample
$G_{TdB}(i,k)$	Target gain in dB at k th spectral sample of i th frame
i	Frame index
k	Spectral sample index
L	Window length
N	DFT size
$\hat{P}(i,k)$	Estimate of peak at k th spectral sample of i th frame
$P_{IdB}(i,k)$	Input power in dB at k th spectral sample of i th frame
$P_{IdBLL}(k)$	Input loud level in dB at k th spectral sample
$P_{IdBSL}(k)$	Input soft level in dB at k th spectral sample
p_n	Probability corresponding to n th quantile
$P_{OdB}(i,k)$	Output power in dB at k th spectral sample of i th frame
$P_{OdBCL}(k)$	Output comfortable level in dB at k th spectral sample
$P_{OdBLL}(k)$	Output loud level in dB at k th spectral sample
$P_{OdBSL}(k)$	Output soft level in dB at k th spectral sample
$\hat{q}_n(i,k)$	n th quantile estimate at k th spectral sample of i th frame
$\hat{R}(i,k)$	Estimate of range at k th spectral sample of i th frame
S	Window shift
T_a	Attack time
T_r	Release time
$\hat{V}(i,k)$	Estimate of valley at k th spectral sample of i th frame

$X(i,k)$	Input at k th spectral sample of i th frame
$Y(i,k)$	Output at k th spectral sample of i th frame
α	Over-subtraction factor for spectral subtraction
β	Noise floor factor for spectral subtraction
γ	Power exponent for spectral subtraction
γ_a	Attack time constant
γ_r	Release time constant
$\Delta_n^+(i,k)$	Increment in n th quantile estimate at k th spectral sample of i th frame
$\Delta_n^-(i,k)$	Decrement in n th quantile estimate at k th spectral sample of i th frame
κ	Smoothing factor for <i>a priori</i> SNR calculation
λ	Convergence factor for quantile tracking
ρ	Smoothing factor for <i>a posteriori</i> SNR calculation
σ	Fall time factor for peak and valley detection
τ	Rise time factor for peak and valley detection
$\psi(i,k)$	<i>a posteriori</i> SNR
$\xi(i,k)$	<i>a priori</i> SNR

LIST OF ABBREVIATIONS

Abbreviation	Explanation
ADC	Analog-to-digital converter
API	Application programming interface
ASIC	Application-specific integrated circuit
CL	Comfortable level
CPU	Central processing unit
CR	Compression ratio
CVR	Consonant-to-vowel ratio
DAC	Digital-to-analog converter
DFT	Discrete Fourier transform
DMA	Direct memory access
DSO	Digital storage oscilloscope
DSP	Digital signal processor
FFT	Fast Fourier transform
GA	Geometric approach to spectral subtraction
GUI	Graphical user interface
I/O	Input-output
IFFT	Inverse fast Fourier transform
JNI	Java native interface
LL	Loud level
NAL	National Acoustics Laboratory prescription procedure
OS	Operating system
PCM	Pulse code modulation
PESQ	Perceptual Evaluation of Speech Quality
POGO	Prescription of Gain/Output
SL	Soft level
SNR	Signal-to-noise ratio
SS	Spectral subtraction
TRRS	Tip-ring-ring-sleeve connector
UI	User interface
VHSES	Vowel Hindi sentence English sentence database
VoIP	Voice over internet protocol

Left blank

Chapter 1

INTRODUCTION

1.1 Overview

Sensorineural hearing loss is associated with loss of sensory hair cells in cochlea or degeneration of auditory nerve. It may be inherited genetically or may be caused by excessive noise exposure, aging, infection, or use of ototoxic drugs. It is characterized by frequency-dependent elevation of hearing thresholds, abnormal growth of loudness known as loudness recruitment, increased temporal and spectral masking, and widening of auditory filters leading to degraded speech perception [1] – [6]. Several signal processing techniques have been reported for improving the speech perception by patients suffering from sensorineural hearing loss.

Frequency selective amplification and dynamic range compression are the primary processing techniques used in hearing aids [6]. Single band dynamic range compression leads to reduced high frequency audibility and multiband dynamic range compression may lead to perceptible distortion due to transition of speech formants across band boundaries. These problems can be addressed by using sliding-band dynamic range compression, reported by Tiwari and Pandey [7]. The compression parameters can be tuned to fit the frequency dependent thresholds and loudness recruitment curves of the patient.

Persons with sensorineural loss experience great difficulty in understanding speech in noisy environment. Reduction of noise helps in improving speech audibility and quality. The noise suppression technique for use in a hearing aid should have low algorithmic delay and low computational complexity. Spectral subtraction [8], a single-channel speech enhancement technique that uses estimate of noise spectrum, is suitable for such applications. Dynamic quantile tracking based noise estimation [9] is reported to track stationary and non-stationary noise efficiently, and can be used for real-time noise suppression.

Hearing aids are generally designed using ASICs due to power and size constraints, leading to prohibitive costs in development and testing of new processing techniques. Use of smartphone-based application (app) as a hearing aid can provide user-configurable settings and a greater flexibility to hearing aid users and developers. It would provide a low-cost alternative for expensive hearing aids.

1.2 Objective

The objective of the project is to implement sliding-band dynamic range compression [7] and dynamic quantile tracking based noise suppression [9] in a smartphone app to enable the use of smartphone as a hearing aid with real-time control of settings. This involves development of real-time audio I/O and processing framework, efficient implementation of the algorithms to meet the requirement of acceptable signal delay and designing a user-friendly graphical user interface (GUI) to configure the processing parameters. The app also serves as a platform for incorporating other processing techniques. The app is implemented and tested for real-time operation using ‘LG Nexus 5X’ running ‘Android 7.1’.

1.3 Outline

A literature survey on the signal processing techniques used in hearing aids and smartphone based hearing aid related applications is provided in Chapter 2. The design and implementation of the hearing aid app is presented in Chapter 3. The signal processing technique, implementation details, and test results for sliding-band dynamic range compression and noise suppression using dynamic quantile tracking are presented in Chapter 4 and 5, respectively. Summary and conclusion are provided in the last chapter.

Chapter 2

SIGNAL PROCESSING FOR HEARING AIDS

2.1 Introduction

Sensorineural hearing loss is characterized by several problems leading to degradation of speech perception. Frequency-dependent increase in hearing thresholds is the primary problem leading to decreased audibility of soft sounds. The thresholds of loudness discomfort do not increase by the same amount as thresholds of hearing, leading to reduced dynamic range [6]. Sensorineural hearing loss also results in increased spectral and temporal masking and severely affects speech perception, particularly in the presence of background noise [6].

Hearing aids are assistive devices designed to improve speech perception in the hearing impaired by alleviating the problems associated with the sensorineural hearing loss. The hearing aids are required to be small enough to be used in day-to-day life. This puts a constraint on size and power, leading to limited processing capability in the hearing aids. Hence, the signal processing techniques to be used in hearing aids should have low computational complexity. Since they should also provide enhancement in real-time to enable the use of hearing aid in face-to-face communication, they should have low signal delay. The following sections provide review on the signal processing techniques that can be used in hearing aids.

2.2 Signal Processing to Reduce the Effect of Decreased Dynamic Range

Frequency-dependent increase in hearing thresholds is the primary problem leading to inaudibility of the soft sounds [6]. Linear frequency-selective amplification alone cannot solve this problem since the increase in hearing thresholds is accompanied with decrease in dynamic range. To map a wider dynamic range input into the limited dynamic range of hearing impaired listener, compression techniques are used.

Dynamic range compression involves changing the gain based on the input signal level for presenting the sounds comfortably within the limited dynamic range of the hearing impaired listener [10]. Analog hearing aids generally provide single-band compression. In this type of processing, the gain is calculated as a function of the signal power over its entire bandwidth [6]. As speech signal is dominated by low frequency components, this

compression distorts the perceived temporal envelope of the signal and often makes the high frequency components inaudible.

Most of the digital hearing aids provide multi-band dynamic range compression which involves dividing the input signal in multiple bands and applying a gain in each band which is a function of the signal power in that band [11]. It avoids the problems associated with single band compression, but results in decreased spectral contrast in the speech signal. Further, different gains in adjacent bands may distort spectral shape of a formant spanning the band boundaries, particularly during formant transitions.

With the objective of reducing the temporal and spectral distortions associated with the compression techniques commonly used in the hearing aids, a sliding-band compression technique has been reported by Tiwari and Pandey [7], [12] for use in digital hearing aids. It calculates a frequency-dependent gain function, wherein the gain for each frequency sample is calculated as a function of the signal power in the auditory critical band centered at it. It keeps compression related distortions below perceptible levels for improving speech perception.

2.3 Signal Processing to Suppress Background Noise

Presence of background noise significantly reduces the speech quality and intelligibility. Background noise affects speech intelligibility drastically in case of sensorineural hearing loss due to limited dynamic range and increased spectral smearing. Noise suppression can be used in hearing aids to increase the SNR and thereby improve the speech perception.

Spectral subtraction [8], [13], [14], a single-channel speech enhancement technique, is suitable for hearing aid application, since it has low computational complexity. It uses processing steps of calculating short-time spectrum of the input speech, dynamically estimating the noise spectrum, subtracting the estimated noise spectrum from the input speech spectrum, and re-synthesizing the output speech signal.

Spectral magnitude subtraction is based on the assumption that speech and noise are uncorrelated. This assumption is not valid for short-time windowed noisy speech. The error caused by this assumption leads to excessive musical noise [13]. A geometrical approach for spectral subtraction (GA) has been proposed by Lu and Loizou [15]. A gain dependent on cross terms due to phase difference between noisy speech and noise estimate is calculated for suppression. It is reported to have no audible musical noise.

Dynamic estimation of the noise spectrum is critical for effective noise suppression. As the interfering noise is usually non-stationary, the noise spectrum has to be estimated dynamically. Under-estimation of noise leads to excessive residual noise and over-estimation results in perceptible distortions leading to degraded quality and intelligibility.

Voice activity detection based methods track noise during the silence regions and the tracked noise is assumed to be stationary during speech segments [16]. They may not work satisfactorily in low-SNR conditions, particularly when noise is non-stationary [13]. Several statistical techniques for dynamic estimation of noise without using voice activity detection has been reported. These are based on statistical assumptions on noise and signal.

Minimum-tracking algorithms are based on the assumption that the minimum power level in a particular frequency bin over past frames corresponds to the noise level in that bin [17], [18]. These algorithms cannot track rapid changes in noise statistics. They often under-estimate the noise and require SNR dependent over-subtraction. Time-recursive averaging algorithms estimate noise as a weighted average of previous noise estimates and present noisy speech. The weights are changed dynamically based on *a posteriori* SNR or speech-presence probability [19] – [21].

Some techniques use a quantile of the noisy speech spectrum in each frequency bin as the noise estimate [22] – [26]. They are based on the observation that the speech energy in a particular frequency bin is low in most of the frames and high only in 10–20% frames. These algorithms require frequency-dependent quantiles for estimation of non-stationary noises [23], [24], [26]. Several histogram-based techniques that estimate noise as the maximum of the distribution of energy values in each frequency bin have been reported [27], [28]. The quantile-based techniques are not suitable for use in hearing aids due to large memory requirement and high computational complexity involved in storing and sorting the spectral samples. The histogram-based techniques pose even higher implementation challenges as they require estimation of multiple quantiles.

A technique for noise spectrum estimation based on dynamic quantile tracking as an approximation to the sample quantile, without involving storage and sorting of past samples has been reported by Tiwari and Pandey [9], [29]. Histogram is tracked dynamically by tracking quantiles and its peak is used as the adaptive quantile for estimating the noise at each spectral sample for speech enhancement.

2.4 Signal Processing to Reduce the Effect of Increased Spectral and Temporal Masking

Sensorineural hearing loss is characterized by widening of auditory filters leading to increased intra-speech spectral masking. Several signal processing techniques have been proposed to reduce the effect of intra-speech masking. Multi-band frequency compression techniques [30] – [32] present the speech signal in relatively narrow bands by compressing the spectral samples in each band towards the band center. Increasing the ratio of intensity of consonant

segments to vowel segments (CVR) has been reported to improve recognition of consonants in noisy environment [33].

Binaural dichotic presentation techniques reduce the effect of internal masking by presenting alternate band information to left and right ear. Use of constant bandwidth filters has been reported to show no considerable improvements in speech recognition scores [34], [35]. Auditory critical bandwidth based comb filters showed 5 – 9.5% increase in recognition scores. Time varying comb filters with sweep cycle duration of 40 – 80 ms and 8 – 16 shifts has been reported to give maximum improvement in recognition scores for subjects with severe symmetrical hearing loss [36]. For perceptual balance in perceived loudness, the comb filters should have complementary magnitude response in linear scale [37].

2.5 Smartphone as a Hearing Aid Platform

Hearing aids are generally designed using ASICs due to power and size constraints. The prohibitive costs in designing an ASIC-based hearing aid and in development and testing of new processing techniques call for an alternative platform. Development of application software (app) for smartphones is emerging as a convenient and inexpensive platform for rapid implementation and testing of signal processing techniques for audiometry and hearing aids.

Android and iOS are the two most commonly used mobile platforms, with market shares of 87.6% and 11.7%, respectively [38]. As Android-based phones have a large user base, building an Android app would reach a large number of users. One of the major drawbacks in implementing audio app for Android-based devices is that they currently have audio I/O latency of 20 – 200 ms as compared to 4 – 20 ms for iOS-based devices [39].

There are many apps currently available in mobile platforms which aim to provide hearing related utilities. The functionality of the apps can be broadly classified into three types i.e., audiometry apps, hearing aid control apps, and hearing aid apps.

2.5.1 Audiometry Apps

These apps provide diagnosis on the hearing loss by performing audiometric tests. Pure tone audiometry is the most common audiometric test implemented in apps, wherein a single tone from the set of standard frequencies is presented with increasing amplitude, and the thresholds of hearing is obtained. Calibration of the headphones is one of the major challenges in accurately determining the hearing profile.

‘Audiocal’ [40] is an iOS application with pure tone audiometry and a diagnosis on the extent of hearing loss. The calibration is done for standard iPod headphones manually. uHear [41] provides a plot of hearing thresholds along with the threshold values at the test frequencies. The hearing loss diagnosis is obtained using a detailed questionnaire to be

answered by the patient. ‘Hearing Test’ [42], ‘Q+’ [43], ‘uSound’ [44], and ‘Petralex’ [45] provide threshold curves in Android platform. Since there is a huge variability in headphones for Android phones, these apps obtain relative threshold curves, but cannot be used to diagnose the extent of hearing loss.

2.5.2 Hearing Aid Control Apps

Many hearing aid manufacturers provide apps to control hearing aids using Android or iOS smartphone. These apps personalize the listening experience of users by allowing them to adjust settings on their hearing aids using their smartphones without removing their hearing aids. GN ReSound [46], Phonak [47], Unitron [48], Seimens [49] provide apps to control the parameters such as noise environment, loss compensation mode, compression modes, which will be used to configure the hearing aid via Bluetooth. The hearing aid control apps currently available do not permit fine tuning of the processing parameters in real-time.

2.5.3 Hearing Aid Apps

Hearing aid apps to compensate for sensorineural hearing loss are being developed as a low-cost and more flexible alternative to hearing aids. ‘Hearing Aid with Replay (Lite)’ [50] provides single-band compression and gain control. It also enables user to replay last 15s audio. ‘uSound’ [44] has provisions for audiometric test and provides single-band compression with parameters based on the test. ‘Petralex’ [45] for Android/iOS provides audiometric test, single-band compression with four modes of compression ratios, and amplification profiles of standard fitting procedures such as NAL [51], Berger and POGO [52]. It also provides acoustic feedback cancellation. ‘Q+’ [43] provides compression along with feedback noise reduction. The Android version of these apps, except ‘Petralex’, have latency above 120 ms which is not acceptable for conversational speech.

2.6 Scope of the Project

A digital hearing aid is implemented as a smartphone app with dynamic quantile tracking based noise suppression [9] and sliding-band dynamic range compression [7]. The app consists of a real-time audio I/O framework, FFT-based processing framework, and a touch-controlled GUI. The app also acts as a platform to implement and test additional signal processing techniques, which employ signal processing through spectral modification. The implementation details of the app are described in the next chapter.

Left blank

Chapter 3

HEARING AID IMPLEMENTATION AS A SMARTPHONE APP

3.1 Introduction

A hearing aid application should have low latency for effective use in face-to-face communication. It is reported that the audio delay should be less than 125 ms to avoid detectable audio-visual desynchrony [53]. After examining the processing capability and audio I/O delay of several currently available Android-based handsets [39], implementation of the app has been carried out and tested using ‘Nexus 5X’ with Android 7.1 Nougat OS. The device has hexacore CPU operating at 1.8 GHz [54]. A signal bandwidth of 4 – 5 kHz is generally considered adequate for hearing aid application; hence a sampling frequency of 12 kHz may be used. Android-based handsets provide a default audio sampling rate of 48 kHz/44.1 kHz which may be decreased using the re-sampler block at an expense of increase in the input-output latency due to intermediate buffers, antialiasing filters and smoothening filters. Considering the high computational load associated with sampling frequency of 48 kHz and increased latency due to re-sampler block with 12 kHz, a sampling frequency $F_s = 24\text{ kHz}$ is used. The downsampling is carried out by dropping alternate samples of the acquired signal instead of using the re-sampler block. This method reduces the latency associated with re-sampling, but at the risk of some aliasing related signal degradation.

The application was developed using a combination of C++ and Java, with Android Studio 2.2.0 as the development environment. Figure 3.1 shows block diagram of the implementation of hearing aid app. The application can be divided into three functional blocks i.e., audio I/O framework, processing framework, and GUI. The audio I/O framework handles low latency data-transfers along with buffering. The processing framework provides an analysis-modification-synthesis platform for implementation of FFT-based signal processing techniques. The processing parameters and status of the app are communicated between user and processing framework using the GUI.

The input analog signal acquired from microphone of the headset is amplified by input amplifier and is converted to digital samples by ADC. The digital samples are processed

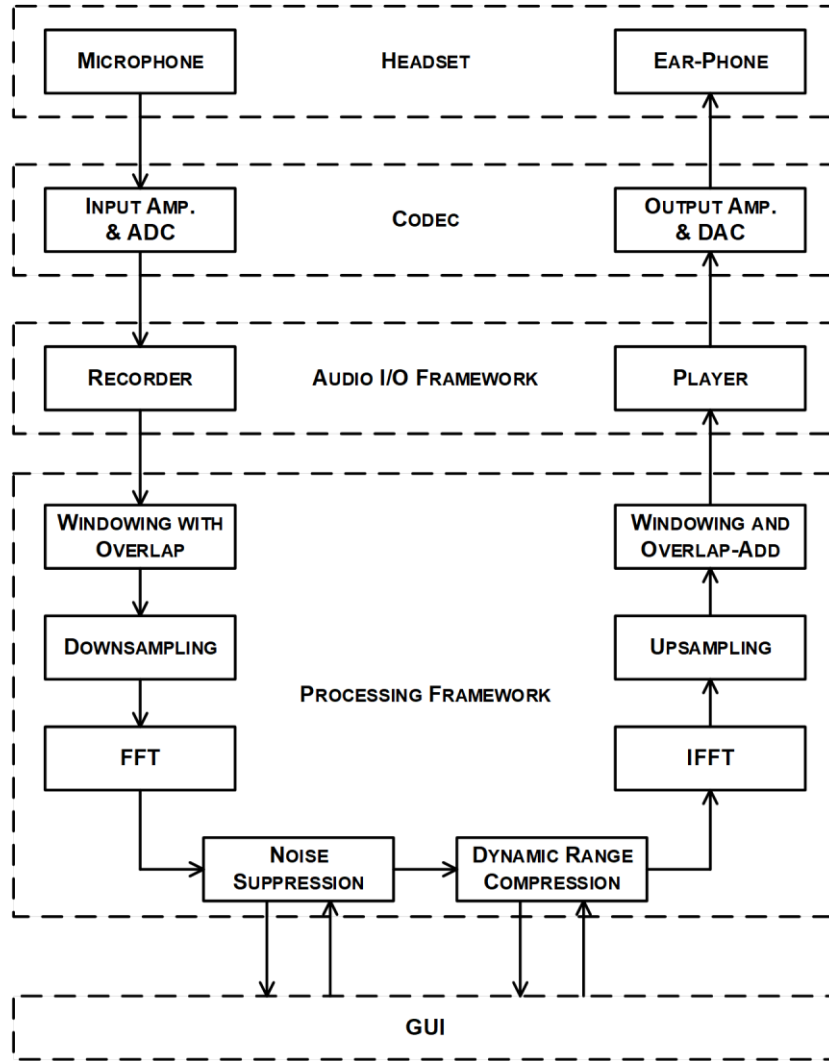


Figure 3.1 Implementation of the hearing aid application.

and converted back to analog signal using DAC. The analog output from DAC is amplified by output amplifier and is available through earphones of the headset. The functional blocks of the app are described in following sections.

3.2 Audio I/O Framework

The I/O latency of currently available Android Java APIs is beyond the acceptable value of 125 ms. Hence, Open SLES library [55], a hardware-accelerated audio API for I/O handling, is used for the implementation of the audio I/O framework in native C++.

The audio I/O and processing framework are created within a service to provide uninterrupted audio I/O even when the user switches to another app, or when the app is not in foreground. The service is flagged as a foreground service, by providing an ‘ongoing’ notification to the user, to avoid termination of the service by OS in low memory conditions.

Once in foreground, the app either binds the GUI to the running service or starts the audio service if not started earlier.

Using Open SLES library [55], an audio engine with recording and playback capabilities is created. It holds references to a recorder object, a player object, and associated memory blocks. The recorder object handles audio input and the player object handles audio output. Recorder and player objects are created with required sampling rate F_s , shift length S , encoding format and mono/stereo specifications. Android supports PCM encoding of the acquired audio in three formats, viz. unsigned integer, signed integer, and float. The float format is not supported for low latency audio in Android 7.1. Therefore, signed integer format is used with 16 bits per sample. As the TRRS headset port provides mono channel for microphone and stereo for earphones, recorder is initialized with mono and player is initialized with stereo specification. Each memory block used by recorder and player is of $2S$ -word length corresponding to 5 ms of audio data. The size of each $2S$ -block word is given by $2S \times \text{No. of bits per sample} \times \text{No. of channels}$ bits.

Ring buffers with a fixed number of blocks are commonly used for real-time audio I/O in dedicated DSPs. However in a multitasking environment, variability in scheduling and processor frequency and lags in callbacks may cause unavailability of blocks to record/play (called as buffer underrun). In these situations, the already recorded block may get overwritten by the newly acquired samples corrupting the data. Similarly, unavailability of the processed block for the player would cause periodic glitches in the audio output. Recovery from this scenario requires resetting the entire audio I/O framework. To avoid it, fixed buffers with higher length could be used to ensure the availability of blocks for record/play, but at the cost of increased latency. The optimum length of the buffers depends on the processing load and processing capabilities of the handset. To provide handset-independent solution to buffer underruns, the proposed implementation uses extensible queues instead of fixed length buffers. The maximum number of memory blocks permissible in the queue is increased only at buffer underruns, providing glitch-free audio I/O with minimum possible latency.

Usage of blocking calls (function calls that suspend the calling function until the expected event occurs) to the OS, such as memory allocation, mutex locks, etc., should be minimal to avoid buffer underruns in real-time applications. To minimize blocking calls, the memory required to handle audio I/O and processing are pre-allocated. The data transfer between queues in audio I/O and processing framework are carried out using pointers to pre-allocated memory such that no memory allocation is required when the hearing aid app is running.

Both recorder and player have internal queues to keep track of the free, recorded, and played memory blocks and the blocks that are currently being recorded or played, as shown in

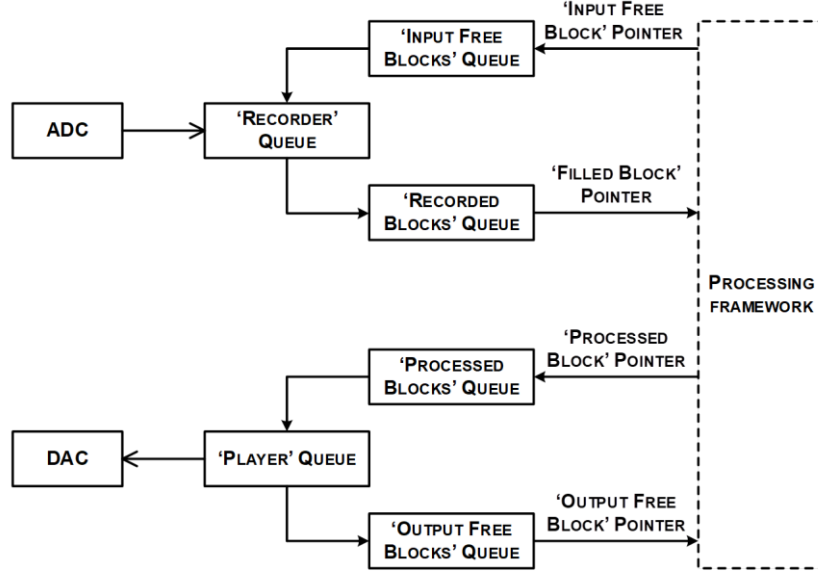


Figure 3.2 Audio I/O Framework.

Figure 3.2. The input and output free buffer queues hold pointers to the free memory blocks corresponding to input and output sides, respectively. The recorder queue maintains the pointers to memory blocks which are ready to be filled. The recorded blocks queue holds pointers to the memory blocks that are recorded but not yet processed. The processed blocks queue maintains pointers to the memory blocks that are processed but not yet output. The player queue holds pointers to the memory blocks ready to output. The input and output free block queue length can vary from 1 – 16, and all other queue lengths can vary between 1 – 4. During initialization, 16 2S-word blocks of memory are allocated to each of recorder and player and the pointers to these blocks are pushed into corresponding free queues.

During initialization of the recorder, pointer to a free block of memory is popped out of the input free blocks queue and pushed into the recorder queue. The data acquired from ADC are filled into the memory block pointed by the head of the recorder queue using DMA. After a memory block is filled, a callback is obtained. The pointer to the filled block is popped out the of the recorder queue and pushed into the recorded blocks queue. A memory pointer is popped out of the recorded blocks queue by processing framework and pushed into the input free blocks queue after copying the data.

During initialization of the player, pointer to a 2S-word empty block is pushed into the player queue and playback is initiated. A callback is obtained after a memory block is played. The pointer to the played block is popped out of the player and pushed into the output free blocks queue. A processed block is popped from the processed blocks queue and pushed into the player queue for playing. A memory pointer is popped out of the output free blocks queue by processing framework and pushed into the processed queue after filling it with the processed data.

3.3 Processing Framework

An FFT-based analysis-modification-synthesis framework is implemented with frame length of $L = 480$. It corresponds to 20 ms frame for $F_s = 24$ kHz, such that at least 2 pitch periods of speech signal are captured in a window. The frame shift S is selected to be 120, which corresponds to 5 ms shift and hence 75% frame overlap. This puts a limit of 5 ms on the processing time required per frame for real-time operation. The FFT length N should be larger than the sum of frame length L and overall impulse response of the processing modules to ensure alias-free filtering. Considering the high computational load with increasing N , $N = 1024$ is selected. As the processing modules to be used in the application modify the magnitude spectrum without modifying phase spectrum, the resulting sequence of spectrum may not correspond to a valid continuous time domain sequence. A modified Hamming window [56] is used to estimate a valid spectrum from the modified spectrum.

The processing framework is implemented in a separate thread, called ‘processor thread’ to ensure that audio I/O callbacks are not missed. The processor thread is spawned by the Android Java API so as to give a priority level of `URGENT_AUDIO` which ensures higher scheduling probability among other threads within the process. To avoid the additional latency due to CPU core-switches, the thread affinity is set to the CPU core that the thread is initially scheduled to run.

The processor thread executes the signal processing steps consisting of short-time spectral analysis, spectral modification, and re-synthesis using a FFT-based analysis-synthesis blocks as shown in Figure 3.1. The input samples, real and imaginary parts of spectral samples, and the output samples are stored as 32-bit floating point arrays. To have 75% overlap between the analysis frames, the $2S$ -word input block pointed by the head of recorded queue of audio I/O framework is pushed into a shift register of $8S$ -word length. Thus, input window with $2L$ samples ($8S$ words) is formed using the samples of the just-filled block and the samples from previous three blocks stored in the shift register. These $2L$ samples are multiplied by $2L$ -point modified-Hamming window. The resultant sequence is downsampled by 2 to get L -point sequence and N -point FFT is calculated after zero-padding. Open source ‘Kiss FFT’ library [57] has been used for floating-point FFT computation. Provision is provided to apply multiple mono/stereo processing blocks in series after calculating FFT. For each channel, modified spectrum is obtained by multiplying first $(N/2 + 1)$ complex spectral samples with the corresponding gains. The other samples of modified spectrum are obtained by taking complex conjugate. The $2N$ -point IFFT of the complex spectra are calculated after zero padding and the resulting sequences are multiplied with the $2L$ -point modified-Hamming window to get the modified output frames. The output signals are re-synthesized using overlap-add. A $2S$ -word output is obtained by interleaving the output signals corresponding to

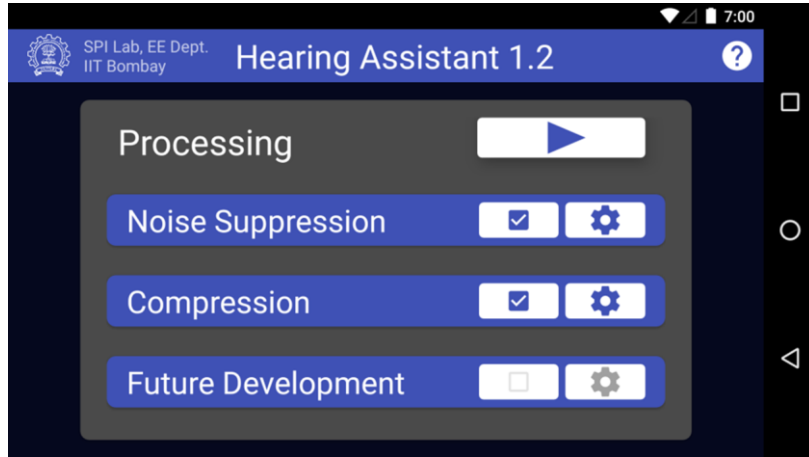


Figure 3.3 Screenshot of the homescreen of the app.

each channel. The output is copied to the memory block pointed by the head of output free queue of audio I/O framework and pushed to processed queue.

3.4 Graphical User Interface

The app is developed with a GUI enabling the user to easily control and configure the hearing aid parameters. The screenshot of the home screen is shown in Figure 3.3. Help button in the title-bar provides the usage tips as shown in Appendix A. The play/stop button is for playback control of audio. Each signal processing module is represented by a card in the home screen, with ‘on/off’ checkbox and ‘settings’ button. The ‘on/off’ checkbox of a module toggles the corresponding processing technique. The ‘settings’ button of each module opens the corresponding setting screen consisting of tunable processing parameters described in respective chapters.

The parameters set using GUI, and the status parameters obtained from the processing modules are exchanged via Java native interface (JNI). It acts as a bridge between the Java virtual environment in which the application is running and the native C++ runtime. The parameters set using GUI are immediately communicated to the processing module and therefore are reflected in the processed output in time less than that corresponding to 2S-word (5 ms). The set parameters are saved in application data memory for future use. The status of the processing modules is polled every 20 ms to update the GUI.

3.5 Test Results

The implementation of the app was tested on the handset model ‘Nexus 5X’ with Android 7.1 OS. Qualitative evaluation was carried out using the headset of the handset for speech input through its microphone and audio output through its earphone. The objective evaluation of framework was carried out using a 4-pin TRRS (tip-ring-ring-sleeve) connector to the headset

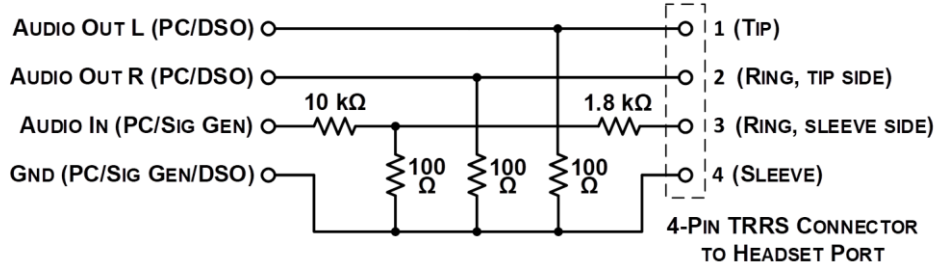


Figure 3.4 Audio interface to the 4-pin TRRS headset port of the mobile handset.

port of the handset, as shown in Figure 3.4. It has a resistive attenuator for attenuating the input audio signal to a level compatible with the microphone signal level and output resistance of $1.8\text{ k}\Omega$ for it to be recognized as external microphone. The two output channels have $100\text{ }\Omega$ load resistances. The input audio signal can be from a PC sound card or function generator.

The total audio latency of the application (signal delay comprising the algorithmic delay and input-output delay) was measured using a 1 kHz tone burst of 200 ms as the input and using a DSO. It was found to be approximately 45 ms . The algorithmic delay due to 20 ms frame length with 75% overlap corresponds to 25 ms (1.25 times the frame length). The additional delay is due to audio input-output latency of the handset hardware, buffering operations in the OS, and delays in the anti-aliasing and smoothening filters.

The time taken for computation per frame is measured using ‘Android device monitor’, a profiling tool for Android OS. The average CPU time per frame taken by the processor thread for audio loopback (inclusive of FFT and IFFT operations) without any processing modules is measured to be 0.45 ms , indicating availability of 4.5 ms CPU time per frame for signal processing operations.

3.6 Summary

The hearing aid application is implemented with low latency audio I/O. An audio I/O framework is implemented with internal extensible queues to provide continuous and glitch-free audio output. A processing framework with a provision of implementing multiple FFT-based processing techniques is implemented. The processing framework has algorithmic delay of 25 ms . As the processing of each frame has to get completed within the frame-shift interval, the computational delay for real-time processing has to be less than 5 ms . The average CPU time taken for the loopback without any processing is measured to be 0.45 ms . Thus processing modules with combined computational requirement of up to 4.5 ms can be incorporated. The signal processing techniques implemented in this work are described in the following chapters.

Left blank

Chapter 4

SLIDING-BAND DYNAMIC RANGE COMPRESSION

4.1 Signal Processing Technique

The processing for sliding-band compression [7], [12] uses a DFT-based analysis-synthesis comprising the steps of short-time spectral analysis, spectral modification, and signal re-synthesis. To compensate for increased hearing thresholds and reduced dynamic range, a frequency-dependent gain function is calculated in accordance with the desired levels for ‘soft’, ‘comfortable’, and ‘loud’ sounds (referred to as SL, CL, LL, respectively). This gain function is used for modification of the short-time input complex spectrum.

For each spectral sample k , the spectral modification is carried out using a piece-wise linear relation between the input power and the output power on dB scale as shown in Figure 4.1. The relationship is specified by the values of $P_{OdBSL}(k)$, $P_{OdBCL}(k)$, and $P_{OdBLL}(k)$ which are the output signal levels corresponding to soft, comfortable, and loud sounds, respectively, for the hearing aid user and by the values of $P_{IdBSL}(k)$ and $P_{IdBLL}(k)$ which are the input signal levels corresponding to soft and loud sounds, respectively, for a normal-hearing listener.

The lower segment of the input-output relationship, marked as ‘CR = 1’ in Figure 4.1, corresponds to linear gain, providing the amplification needed for soft level sounds and the value of the gain is given as

$$G_{LdB}(k) = P_{OdBSL}(k) - P_{IdBSL}(k) \quad (4.1)$$

This gain is applied unless the output exceeds the comfortable level. Thus the input-output relationship in this segment for i th frame is given as the following:

$$P_{OdB}(i, k) = P_{IdB}(i, k) + G_{LdB}(k) , \quad P_{IdB}(i, k) < P_{OdBCL}(k) - G_{LdB}(k) \quad (4.2)$$

The central segment of the relationship, marked as ‘CR > 1’ in Figure 4.1, corresponds to compression with compression ratio $CR(k)$ given as

$$CR(k) = \frac{P_{IdBLL}(k) - P_{OdBCL}(k) + G_{LdB}(k)}{P_{OdBLL}(k) - P_{OdBCL}(k)} \quad (4.3)$$

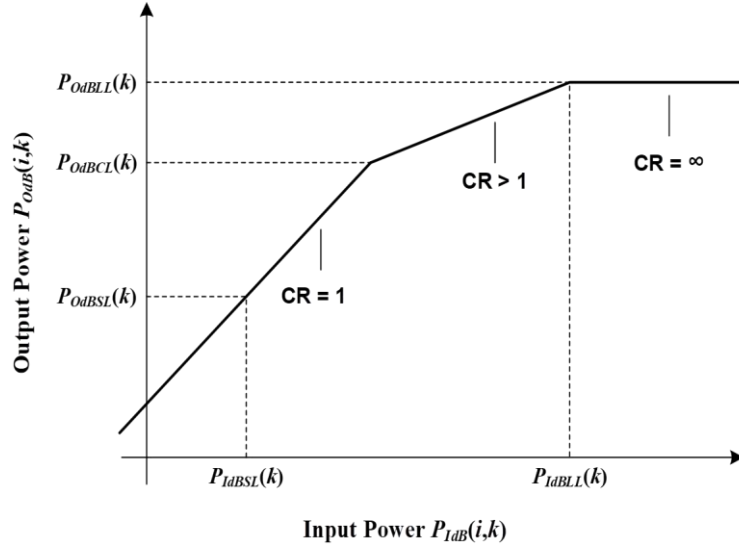


Figure 4.1 Relation between input power (dB) and output power (dB) for i th frame and band centered at k th spectral sample.

This relationship is applicable for the output lying between comfortable and loud levels, and the input-output relationship is given as the following:

$$P_{OdB}(i,k) = P_{OdBCL}(k) + \frac{P_{IdB}(i,k) - P_{OdBCL}(k) + G_{LdB}(k)}{CR(k)}, \quad (4.4)$$

$$P_{OdBCL}(k) - G_{LdB}(k) \leq P_{IdB}(i,k) \leq P_{IdBLL}(k)$$

The upper segment of the relationship, marked as ‘CR = ∞ ’, corresponds to limiting of the output power and is applied when the output exceeds loud level. The relationship is given as the following:

$$P_{OdB}(i,k) = P_{OdBLL}(k), \quad P_{IdB}(i,k) > P_{IdBLL}(k) \quad (4.5)$$

In accordance with the input-output relationship as represented by the three segments and given in (4.2), (4.4), and (4.5), the target gain for the spectral sample k in i th frame is given as

$$G_{TdB}(i,k) = \begin{cases} G_{LdB}(k), & P_{IdB}(i,k) < P_{OdBCL}(k) - G_{LdB}(k) \\ \frac{G_{LdB}(k) - \{P_{IdB}(i,k) - P_{OdBCL}(k)\} \{CR(k) - 1\}}{CR(k)}, & P_{OdBCL}(k) - G_{LdB}(k) \leq P_{IdB}(i,k) \leq P_{IdBLL}(k) \\ P_{OdBLL}(k) - P_{IdB}(i,k), & P_{IdB}(i,k) > P_{IdBLL}(k) \end{cases} \quad (4.6)$$

Block diagram of the spectral modification is shown in Figure 4.2. For each spectral sample k in i th frame, the input level $P_{IdB}(i,k)$ is calculated as the sum of squared magnitude of the spectral samples in the band centered at k and with bandwidth corresponding to auditory critical bandwidth given as

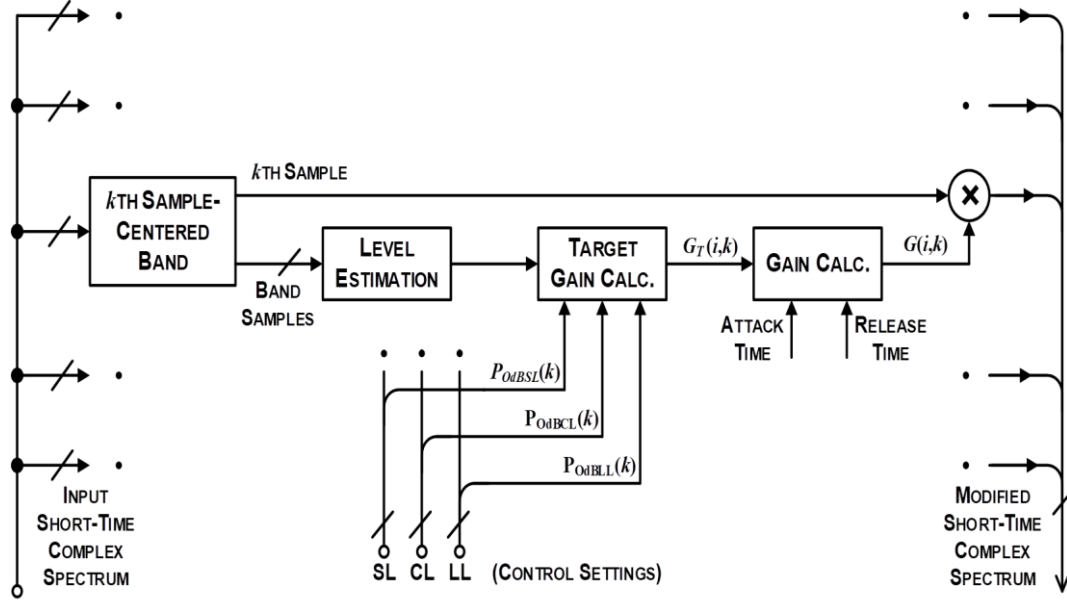


Figure 4.2 Spectral modification for compensation of increased hearing thresholds and decreased dynamic range using sliding-band dynamic range compression, adapted from [13].

$$BW(k) = 25 + 75 \left(1 + 1.4 (f(k))^2 \right)^{0.69} \quad (4.7)$$

where $f(k)$ is the frequency corresponding to k th spectral sample in kHz.

For spectral modification, the target gain is converted to linear scale. The gain applied to the k th spectral sample in the i th frame is obtained using the desired attack and release times by updating the gain from the previous value towards the target value, as given in (4.6), and is given as

$$G(i, k) = \begin{cases} \max(G(i-1, k) / \gamma_a, G_T(i, k)), & G_T(i, k) < G(i-1, k) \\ \min(G(i-1, k) \cdot \gamma_r, G_T(i, k)), & G_T(i, k) \geq G(i-1, k) \end{cases} \quad (4.8)$$

The number of steps during the attack and release phases are controlled using gain ratios $\gamma_a = (G_{max}/G_{min})^{1/s_a}$ and $\gamma_r = (G_{max}/G_{min})^{1/s_r}$, respectively. Here G_{max} and G_{min} are the maximum and minimum possible values of target gain. The number of steps during attack s_a and the number of steps during release s_r are selected to set the attack time as $T_a = S_a \cdot S / f_s$ and release time as $T_r = S_r \cdot S / f_s$, where f_s is the sampling frequency and S is the number of samples for frame shift. A fast attack avoids the output level from exceeding the uncomfortable level during transients, and a slow release avoids the pumping effect or amplification of breathing.

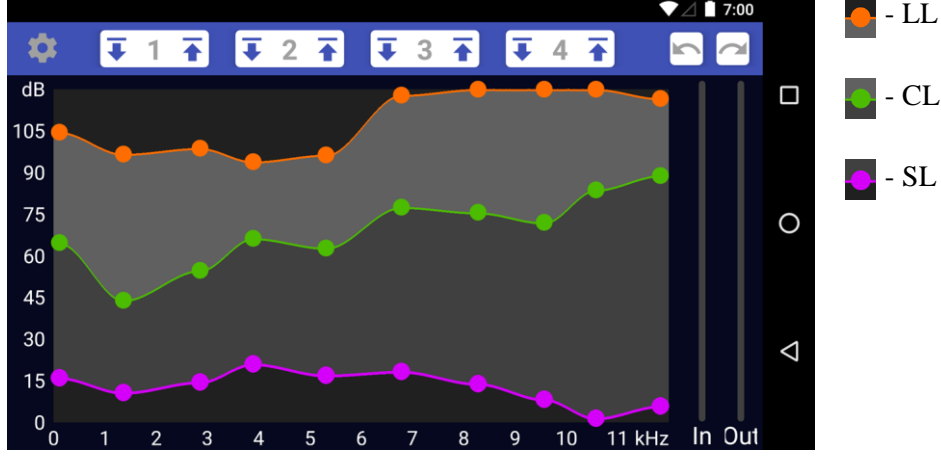


Figure 4.3 Screenshot of the settings screen for sliding-band dynamic range compression.

4.2 Implementation Details

The processing module is implemented with the sampling rate of $F_s = 24$ kHz. The memory required to store the previous gains, current SL, CL, and LL parameters, and power in band centered at each frequency bin is allocated while initialization. The bandwidths for bands centered at each bin as given in (4.7) are pre-calculated and stored. Attack time step S_a and release time step S_r are set to 1 and 15 respectively. The peak audio input value for the device is taken as 120 dB and P_{dBLL} is set to 120 dB. P_{dBSL} is set to 0 dB, and hence the dynamic range of normal hearing user is assumed to be 120 dB in the implementation.

The dynamic range compression can be turned on/off using the checkbox provided in homescreen of the app. To graphically control the SL, CL, and LL values, a custom UI component is designed as shown in Figure 4.3. It can be accessed using settings button of the module. The UI consists of three touch-controlled curves to set the values of SL, CL, and LL across frequencies. Control points called as thumbs are provided to adjust the curves. Each curve consists of 10 thumbs. Provision is provided to store and retrieve up to 4 parameter settings. The UI also consists of undo and redo button to access recent thumb movements.

A thumb can be moved both horizontally and vertically by dragging. i th thumb of each curve has the same horizontal position such that values for all three curves are specified at same frequency bin. Vertical dragging is constrained by the upper and lower curves whereas horizontal dragging is unconstrained.

When a touch-event occurs, the set of thumbs which are closest with respect to x-coordinate of touch is obtained, and then among them, the closest thumb with respect to y-coordinate is selected and dragged to the touch position. This provides the flexibility to select and move the thumbs without dragging it from the initial position. With each change in the thumb position, the curve values for all the intermediate frequencies are obtained by a

smooth curve fitting through the thumbs. Cubic spline fitting is used to ensure that the curve between two adjacent control points is monotonic using Fritsch–Carlson method of spline interpolation [58]. The obtained curve values are plotted on screen and communicated to the processing module using JNI.

The position of the thumbs and the interpolated curves are saved in the application data memory on closing the custom view, so that the settings are retained for the next session. Parameters can be saved in memory to store the settings for different usage environments. The thumb positions are stored in the program memory by long-touch of corresponding upload button. Long-touch of load button restores the saved thumb positions from the program memory store and communicates the parameters to processing module. A vibration is provided as a feedback for successful save/load.

Each thumb movement record is saved as a <thumb id, initial coordinates, final coordinates> entry in a queue to provide undo feature. The queue length is fixed to 10 and hence 10 undo operations can be carried out in succession. The movement record that is reverted back using undo is pushed into another queue to provide redo. Any thumb movement other than undo/redo will clear the redo queue.

The input and output power levels are displayed using vertical colour-filled bars. The sliding-band smoothened input spectrum is overlaid on the screen to provide an idea of the spectral content of input, which may help the user to fine tune the curves. The values for display of power levels and spectra are obtained from processing modules using JNI by polling for values every 25 ms.

4.3 Test Results

The processing module was tested on the handset model ‘Nexus 5X’ with Android 7.1 OS. Qualitative evaluation was carried out using the headset of the handset for speech input through its microphone and audio output through its earphone. The objective evaluation of the processing was carried out using a 4-pin TRRS (tip-ring-ring-sleeve) connector to the headset port of the handset, as described in section 3.5. A PC sound card was used for applying the audio input and acquiring the processed output.

The gain and compression characteristics were tested using a sinusoidal input with constant amplitude and frequency swept from 100 Hz to 10 kHz over 30 s. An example of the processing for frequency-wise constant gain and constant compression along with parameters set using GUI is shown in Figure 4.4. For constant gain, SL was set to 12 dB at all frequency and CL and LL were set to the maximum. For constant compression, SL was set to 12 dB, CL to 80 dB and LL to 106 dB, corresponding to compression ratio of 2. The output waveform for constant compression does not have any temporal distortions which may be observed in multi-band compression.

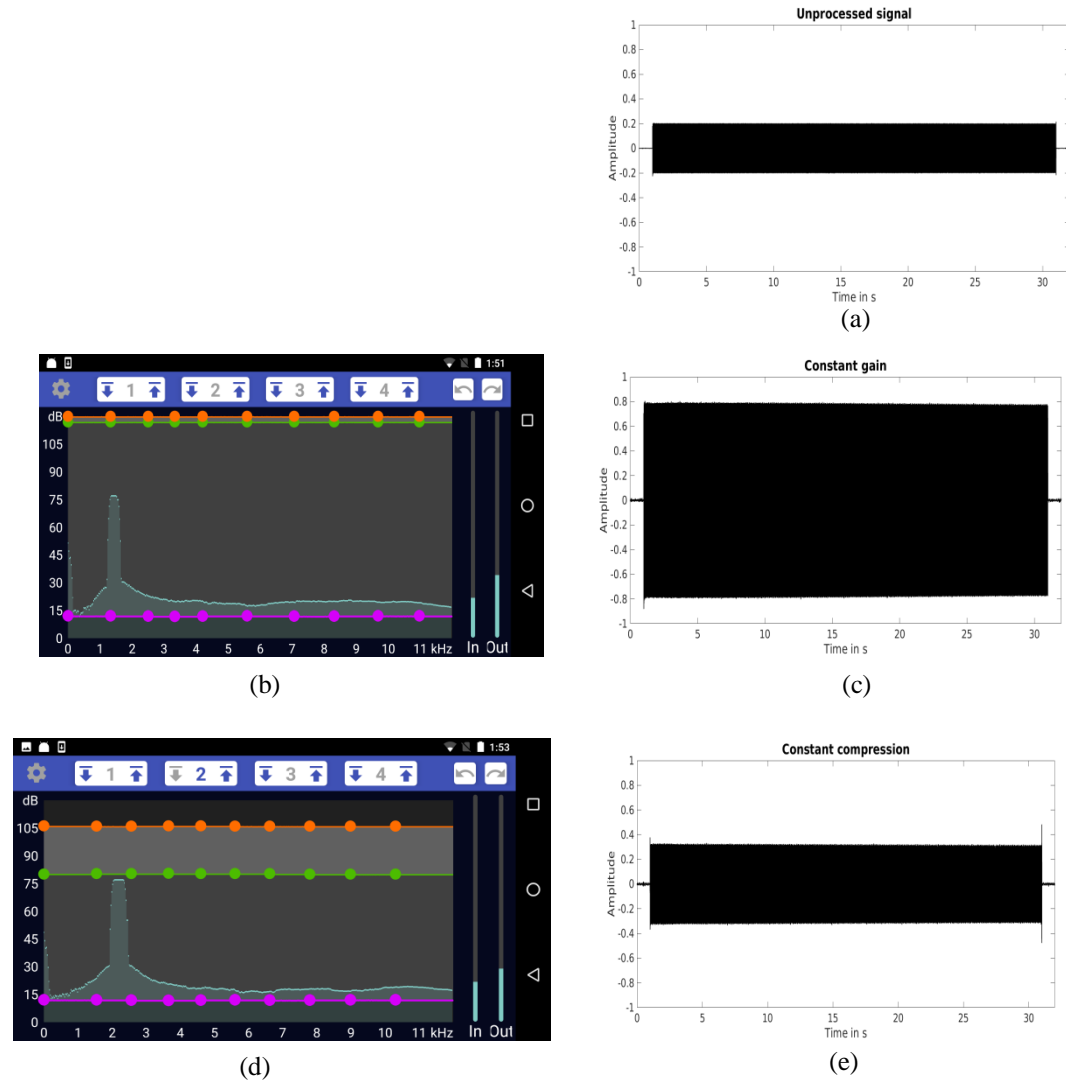


Figure 4.4 Example of processing for constant gain and compression: (a) input; constant amplitude tone with frequency linearly swept from 100 Hz to 10 kHz in 30 s, (b) GUI parameters set for constant gain of 12 dB, (c) processed output for constant gain, (d) GUI parameters set for constant gain of 12 dB and compression ratio of 2, and (e) processed output for constant gain and compression.

An example for frequency-dependent gain and compression for the same input is shown in Figure 4.6. The processed output for frequency-dependent gain shows changes in amplitude of the swept tone in accordance with the soft-level curve. The output for frequency-dependent compression shows amplitude variations in accordance with the comfortable-level curve, but with lower modulation depth.

An example for dynamic range compression with amplitude modulated input is shown in Figure 4.5. Input is an amplitude-modulated tone of 1 kHz and processing parameters are set as shown in Figure 4.4 (d) with compression ratio of 2. The processing gives higher gains at lower values of the input level. Spikes in the amplitude envelope of the output signal in response to step changes in the amplitude envelope of the input signal, as seen in the figure, are typical of the dynamic range compression with a finite frame shift and

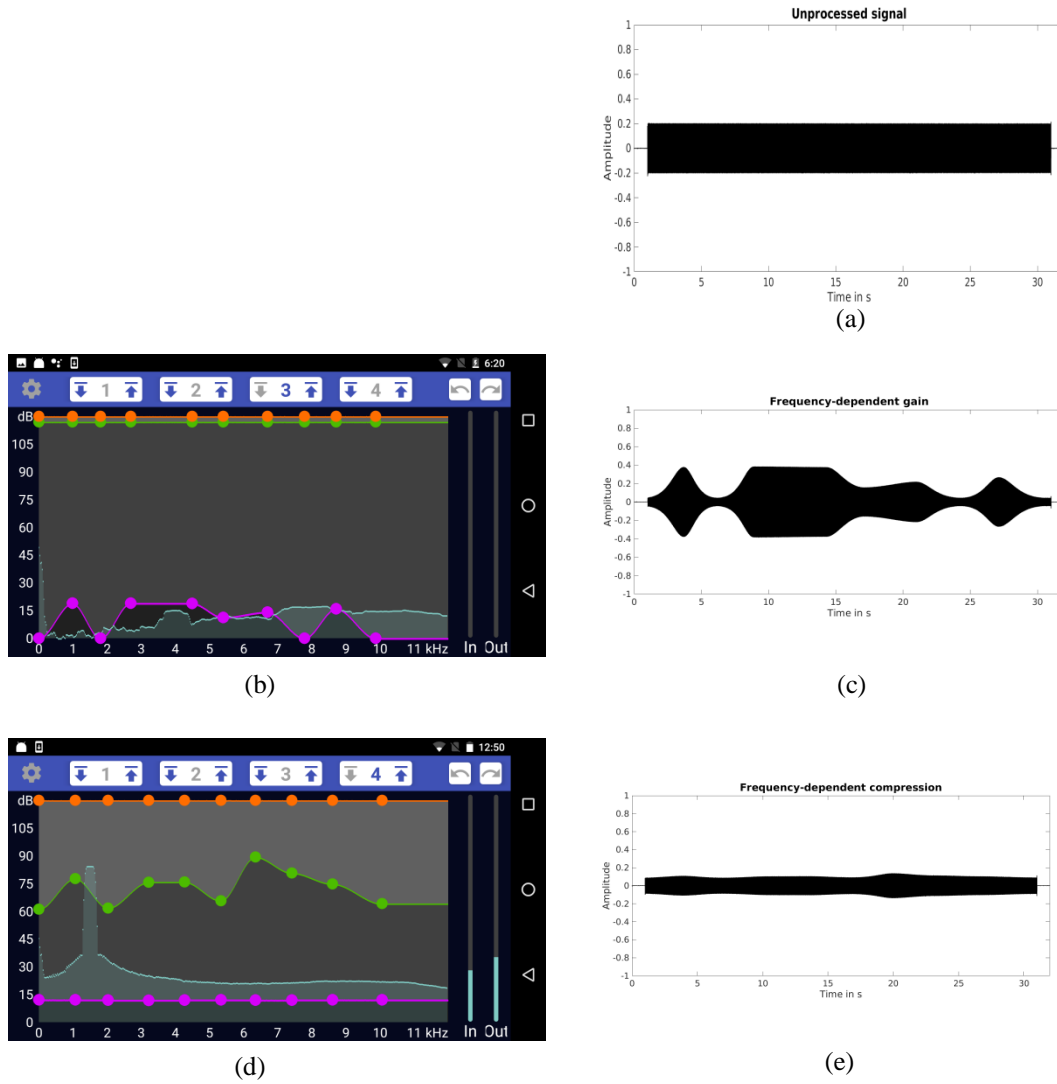


Figure 4.5 Example of processing for frequency-dependent gain and compression: (a) input; constant amplitude tone with frequency linearly swept from 100 Hz to 10 kHz in 30 s, (b) GUI parameters set for frequency-dependent gain, (c) processed output for frequency-dependent gain, (d) GUI parameters set for frequency-dependent compression, and (e) processed output for frequency-dependent compression.

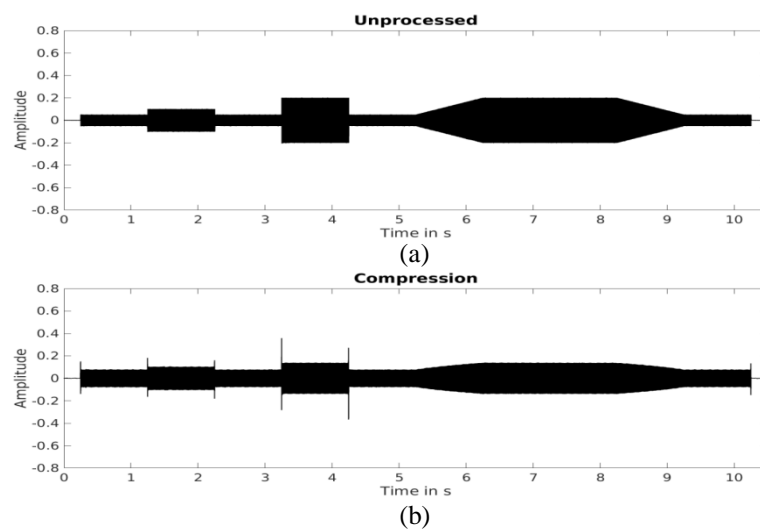
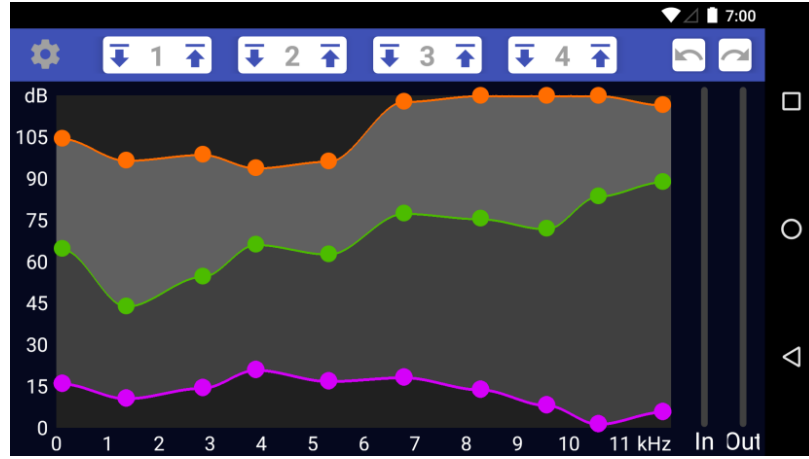
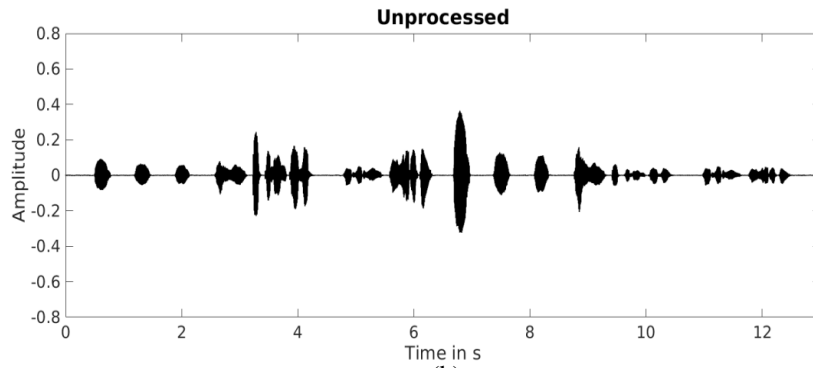


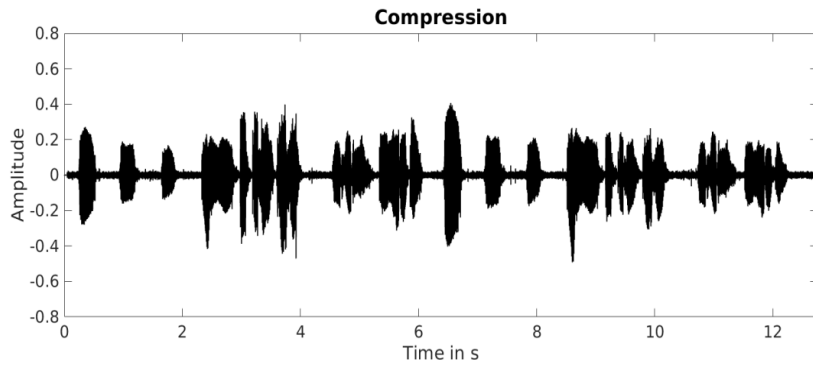
Figure 4.6 Example of processing for dynamic range compression: (a) input; amplitude modulated tone of 1 kHz frequency and (b) processed output for $CR = 2$.



(a)



(b)



(c)

Figure 4.7 Example of processing for dynamic range compression: (a) processing parameters (b) input; amplitude modulated VHSES speech, and (c) processed speech with parameters as shown in (a).

can be eliminated by using one-sample frame shift but with a significantly increased computation load.

The app was tested for speech modulated with different types of amplitude envelopes. An example of the processing is shown in Figure 4.7. An amplitude modulated concatenation of speech segment is given as input. The speech input consists of three isolated vowels, a Hindi sentence, and an English sentence, (-/a/-/i/-/u/-“aaiye aap ka naam kya hai?” – “where were you a year ago?”) referred to as VHSES material. Informal listening test was carried out with different speech materials, music, and environmental sounds with large variation in the

sound level as inputs. The outputs exhibited the desired amplification and compression without introducing perceptible distortions.

The average time taken for computation per frame is measured as the difference of the average CPU time taken per block with and without compression module. The average CPU time taken per block with compression module is measured to be 1.3 ms. Hence the average CPU time taken for compression is 0.85 ms in 'Nexus 5X', when the app is in foreground.

4.4 Summary

The sliding-band dynamic range compression technique is implemented with an interactive GUI to enable fine-tuning of the processing parameters in real-time. The processing has been validated by graphical assessment of the processed output and informal listening. The average processing time overhead for the compression module is 0.85 ms.

Left blank

Chapter 5

NOISE SUPPRESSION USING DYNAMIC QUANTILE TRACKING

5.1 Introduction

The signal processing for noise suppression involves two steps, viz. noise estimation and speech enhancement. For hearing aid application, both noise estimation and speech enhancement methods should have low computational complexity. Dynamic quantile tracking based noise estimation [9] is reported to have low computational complexity and hence it is used along with spectral subtraction for noise suppression.

5.2 Noise Estimation

The most frequent energy value, obtained as the maximum of histogram, in individual frequency bins is reported to be related to the noise level in the specified frequency bins [20]. The noise estimation technique dynamically estimates histogram and the histogram peak is used as the adaptive quantile for estimating the noise at each spectral sample. The histogram is estimated by dynamically tracking multiple quantile values ($q_1, q_2, q_3, \dots, q_M$) for a set of probabilities ($p_1, p_2, p_3, \dots, p_M$), chosen to be evenly spaced. The desired quantile corresponding to the peak of the histogram is obtained by finding p_n for which the difference between neighbouring quantile values is minimum. The estimate of noise spectrum, $\hat{D}(i, k)$ at i th frame and k th spectral sample is obtained as

$$\hat{D}(i, k) = \operatorname{argmin}_{\hat{q}_n(i, k)} [\hat{q}_n(i, k) - \hat{q}_{n-1}(i, k)]; \quad i = 2, 3, \dots, M \quad (5.1)$$

For estimating each of the quantiles, dynamic quantile tracking using range estimation [5], [23] is used. In this technique, an estimate of the quantile of a data stream is obtained without storage and sorting of past samples. The quantile $\hat{q}_n(i, k)$ is estimated by applying an increment or a decrement on its previous estimate. The increment and decrement are selected to be appropriate fractions of the range such that the estimate after a sufficiently large number of input frames matches the sample quantile. As the underlying distribution of the spectral samples is unknown, the range also needs to be dynamically estimated.

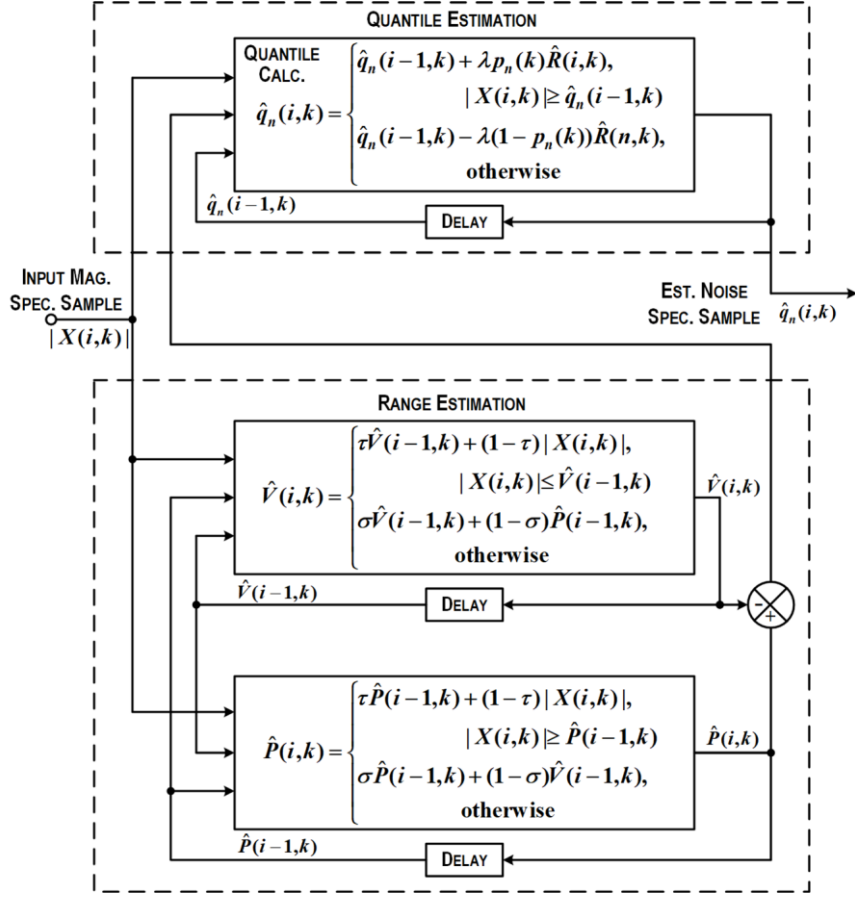


Figure 5.1 Estimation of the noise spectral samples using dynamic quantile tracking technique based on range estimation (adapted from [9]).

The computation steps are shown as a block diagram in Figure 5.1. At k th spectral sample, $q_n(i,k)$ is tracked as the $p_n(k)$ - quantile of the magnitude spectrum $|X(i,k)|$ as

$$\hat{q}_n(i,k) = \hat{q}_n(i-1,k) + d_n(i,k) \quad (5.2)$$

where $d_n(i,k)$ is the change in estimate given as

$$d_n(i,k) = \begin{cases} \Delta_n^+(i,k), & |X(i,k)| \geq \hat{q}_n(i-1,k) \\ -\Delta_n^-(i,k), & \text{otherwise} \end{cases} \quad (5.3)$$

The values of $\Delta_n^+(i,k)$ and $\Delta_n^-(i,k)$ should be such that the ratio $\Delta_n^+(i,k) / \Delta_n^-(i,k) = p_n(k) / (1 - p_n(k))$ to ensure that the quantile estimate approaches the sample quantile and sum of the changes in the estimate approaches zero, i.e. $\sum d_n(i,k) \approx 0$.

Therefore, $\Delta_n^+(i,k)$ and $\Delta_n^-(i,k)$ may be selected as

$$\Delta_n^+(i,k) = \lambda p_n(k) \hat{R}(i,k) \quad (5.4)$$

$$\Delta_n^-(i,k) = \lambda (1 - p_n(k)) \hat{R}(i,k) \quad (5.5)$$

where \hat{R} is estimate of the range (difference between the maximum and minimum values of the sequence of spectral values in a particular frequency bin) and λ is a convergence factor which controls the step size during tracking. Value of λ should be selected for an appropriate trade-off between ripple in the estimated quantile value and the number of steps needed for convergence as described in [5].

The range is estimated using dynamic peak and valley detectors. The estimates of the peak $\hat{P}(i, k)$ and the valley $\hat{V}(i, k)$ are updated, using the following first-order recursive relations:

$$\hat{P}(i, k) = \begin{cases} \tau \hat{P}(i-1, k) + (1-\tau) |X(i, k)|, & |X(i, k)| \geq \hat{P}(i-1, k) \\ \sigma \hat{P}(i-1, k) + (1-\sigma) \hat{V}(i-1, k), & \text{otherwise} \end{cases} \quad (5.6)$$

$$\hat{V}(i, k) = \begin{cases} \tau \hat{V}(i-1, k) + (1-\tau) |X(i, k)|, & |X(i, k)| \leq \hat{V}(i-1, k) \\ \sigma \hat{V}(i-1, k) + (1-\sigma) \hat{P}(i-1, k), & \text{otherwise} \end{cases} \quad (5.7)$$

The constants τ and σ are selected in the range $[0, 1]$ to control the rise and fall times of the detection. As the peak and valley samples may occur after long intervals, τ should be small to provide fast detector responses to an increase in the range and σ should be relatively large to avoid ripples. The range is tracked as

$$\hat{R}(i, k) = \hat{P}(i, k) - \hat{V}(i, k) \quad (5.8)$$

The dynamic quantile tracking to estimate quantile $q_n(i, k)$ as given by (5.2), (5.3), and (5.8) can be written as the following:

$$\hat{q}_n(i, k) = \begin{cases} \hat{q}_n(i-1, k) + \lambda p_n(k) \hat{R}(i, k), & |X(i, k)| \geq \hat{q}_n(i-1, k) \\ \hat{q}_n(i-1, k) - \lambda (1 - p_n(k)) \hat{R}(i, k), & \text{otherwise} \end{cases} \quad (5.9)$$

To estimate the histogram, quantiles corresponding to the set of probabilities $(p_1, p_2, p_3, \dots, p_M)$, are obtained using (5.9) with a common range tracked using (5.6), (5.7), and (5.8). The noise spectrum estimate, $\hat{D}(i, k)$, is obtained using (5.1) as the peak of dynamically tracked histogram.

5.3 Speech Enhancement

Spectral subtraction is used for suppression of additive noise. The processing consists of noise spectrum estimation, enhanced magnitude spectrum calculation, and estimating enhanced complex spectrum without explicit phase estimation. The technique described in previous sub-section is used for estimating the noise magnitude spectrum $\hat{D}(i, k)$. Two techniques, known as generalized spectral subtraction (SS) and geometric approach (GA), have been implemented and tested.

5.3.1 Generalized Spectral Subtraction

In generalized spectral subtraction as reported in [9], the enhanced magnitude spectrum $|Y(i, k)|$ is computed using the noise estimate $\hat{D}(i, k)$ as

$$|Y(i, k)| = \begin{cases} \beta^{1/\gamma} |\hat{D}(i, k)|^\gamma, & |X(i, k)| < (\alpha + \beta)^{1/\gamma} \hat{D}(i, k) \\ [|X(i, k)|^\gamma - \alpha (\hat{D}(i, k))^\gamma]^{1/\gamma}, & \text{otherwise} \end{cases} \quad (5.10)$$

For reducing the broadband peaks in the residual noise, the over-subtraction factor is selected as $\alpha > 1$. Spectral floor factor β is used for masking the warbling or musical noise. The exponent factor γ may be selected as $\gamma = 2$ for power subtraction or $\gamma = 1$ for magnitude subtraction.

The complex spectrum is obtained by combining the enhanced magnitude spectrum with the original noisy phase as

$$Y(i, k) = |Y(i, k)| e^{j\angle X(i, k)} \quad (5.11)$$

In order to avoid phase calculation, the complex spectrum is calculated using

$$Y(i, k) = \frac{|Y(i, k)|}{|X(i, k)|} X(i, k) \quad (5.12)$$

The spectral subtraction can be rewritten as

$$Y(i, k) = G_{ss}(i, k) X(i, k) \quad (5.13)$$

where, the gain $G_{ss}(i, k)$ is given by

$$G_{ss}(i, k) = \begin{cases} \left(1 - \alpha \left(\frac{|\hat{D}(i, k)|}{|X(i, k)|} \right)^\gamma \right)^{1/\gamma}, & |X(i, k)| < (\alpha + \beta)^{1/\gamma} \hat{D}(i, k) \\ \beta^{1/\gamma} \left(\frac{|\hat{D}(i, k)|}{|X(i, k)|} \right), & \text{otherwise} \end{cases} \quad (5.14)$$

5.3.2 Geometric Approach

Generalized spectral subtraction is based on the assumption that clean speech and interfering noise are uncorrelated. This assumption is not valid for short-term analysis with 20 – 30 ms segments [9]. A geometric approach to spectral subtraction is reported in [11], wherein the speech enhancement is carried out by multiplying a SNR-dependent gain function to the input spectrum. The gain $G_{GA}(i, k)$ is given by

$$G_{GA}(i, k) = \sqrt{\frac{1 - \frac{(\psi(i, k) + 1 - \xi(i, k))^2}{4\psi(i, k)}}{1 - \frac{(\psi(i, k) - 1 - \xi(i, k))^2}{4\xi(i, k)}}} \quad (5.15)$$

where, $\psi(i,k)$ is the smoothened *a posteriori* SNR and $\xi(i,k)$ is the smoothened *a priori* SNR. The value of $\psi(i,k)$ is calculated using instantaneous *a posteriori* SNR and previous estimate of *a posteriori* SNR as

$$\psi(i,k) = \rho\psi(i-1,k) + (1-\rho)\left(\frac{|X(i,k)|}{|\hat{D}(i,k)|}\right)^2 \quad (5.16)$$

where ρ is the smoothing constant used in order to reduce rapid fluctuations in estimated *a posteriori* SNR. The value of $\xi(i,k)$ is approximated using enhanced magnitude of previous frame and estimated *a posteriori* SNR of current frame using decision-directed approach as

$$\xi(i,k) = \kappa\left(\frac{|Y(i-1,k)|}{|\hat{D}(i-1,k)|}\right)^2 + (1-\kappa)\left(\sqrt{\psi(i,k)} - 1\right)^2 \quad (5.17)$$

where, κ is the weighting factor for present and past SNR measurements. $\left(\sqrt{\psi(i,k)} - 1\right)^2$ is the lower bound for $\xi(i,k)$ obtained using geometrical approach. The enhanced spectrum $Y(i,k)$ is obtained as

$$Y(i,k) = G_{GA}(i,k)X(i,k) \quad (5.18)$$

Geometric approach requires much higher number of computations compared to spectral subtraction as *a posteriori* and *a priori* SNR estimates are to be calculated for each bin to obtain gain for noise suppression.

5.4 Implementation

The noise suppression module is implemented using dynamic quantile tracking based noise estimation. Twelve quantiles with p-values 0.20, 0.25, ... 0.75 are tracked to get a histogram representation in order to track its peak. The peak and valley tracking are carried out using $\tau = 0.1$ and $\sigma = (0.9)^{1/1024}$ is used for peak and valley tracking. Generalized spectral subtraction with noise floor [13] and spectral subtraction using geometric approach [15] are implemented in ‘Hearing Assistant 1.2.1’ and ‘Hearing Assistant 1.2.2’, respectively. The module acts as a pre-processing module for sliding-band dynamic range compression module.

The settings screen of the noise suppression module for ‘Hearing Assistant 1.2.1’ is shown in Figure 5.2. The values of α and β are input by the user as tunable parameters named as suppression level and noise floor, respectively, using a seekbar. The values of α and β are constrained to $1 \leq \alpha \leq 5$, $0 \leq \beta \leq 0.1$. For ‘Hearing Assistant 1.2.2’, the settings button in homescreen is disabled, as no parameters are required from the user.

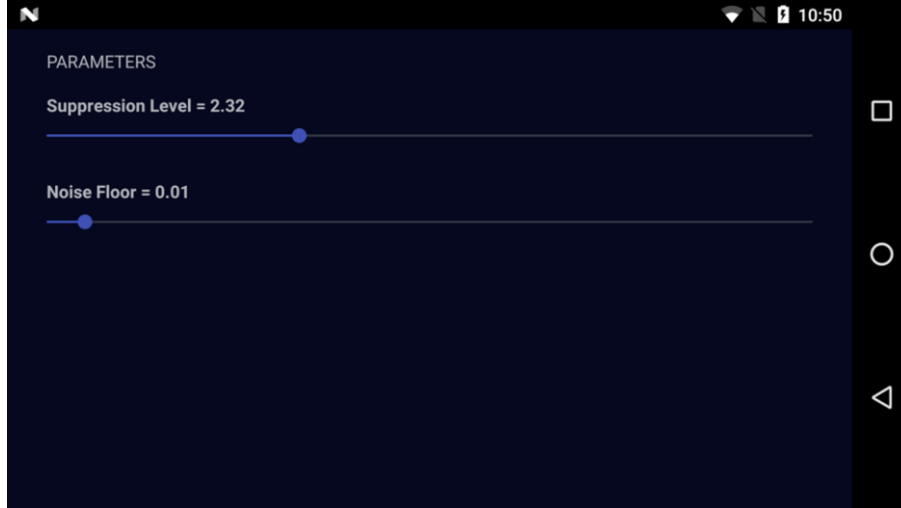
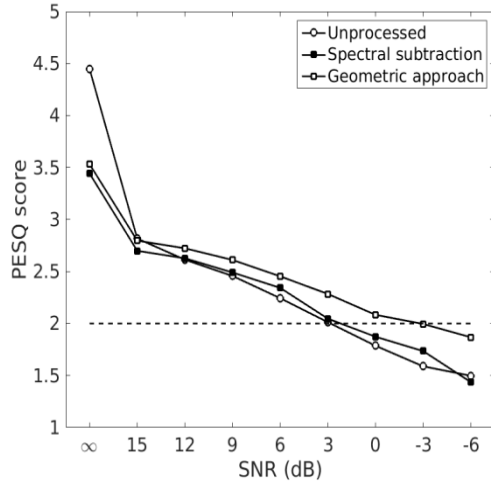


Figure 5.2 Screenshot of the settings screen for noise suppression.

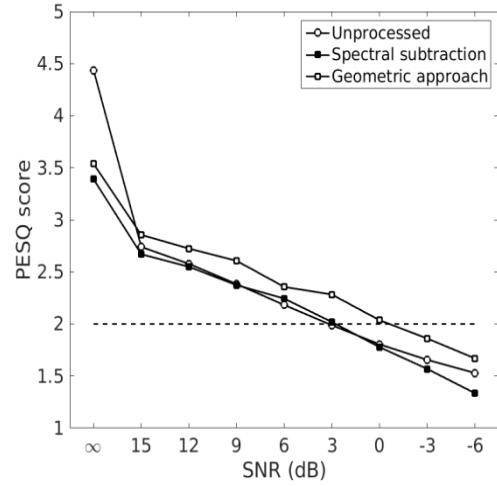
5.5 Test Results

Informal listening and objective evaluation using perceptual evaluation of speech quality (PESQ) measure [59] was used for evaluation of the noise suppression module. The processing module was tested on the handset model ‘Nexus 5X’ with Android 7.1 OS. The objective evaluation of the processing was carried out using 4-pin TRRS (tip-ring-ring-sleeve) connector as described in Section 4.3.

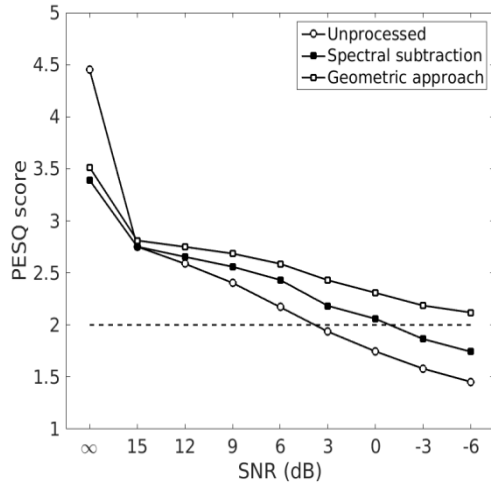
Spectral subtraction and geometric approach methods were tested by adding noise to VHSES speech material. Airport, babble, car, street, and train station noises from AURORA database [60] and white noise are added at SNR of 15, 12, 9, 6, 3, 0, and -3 dB to form noisy speech. The noisy speech was output from PC soundcard and input to the handset using TRRS connector. The processed output was captured using ‘Focusrite Scarlett 2i2’ soundcard to avoid excessive noise observed while capturing audio using PC soundcard. The degradation in PESQ score because of loopback through the device is less than 0.01. The parameters $\alpha = 2$, $\beta = 0.01$, and $\gamma = 1$ were selected as parameters for spectral subtraction as reported in [25]. $\lambda = \frac{1}{256}$ was used for both spectral subtraction and geometric approach. The recorded processed signal was downsampled to 8 kHz for calculation of the PESQ scores. The PESQ scores were obtained using the noise-free VHSES material as reference signal and the recorded processed output as enhanced signal. Figure 5.3 shows the plots of PESQ scores for different types of noises. Improvement of PESQ scores is observed at SNR levels of 12 dB to -3 dB for all type of noises with both the methods. The geometrical approach performs better than spectral subtraction in most of the cases. Table 5.1 shows the SNR improvement of both methods for a PESQ score of 2, which is considered as lowest score for acceptable speech



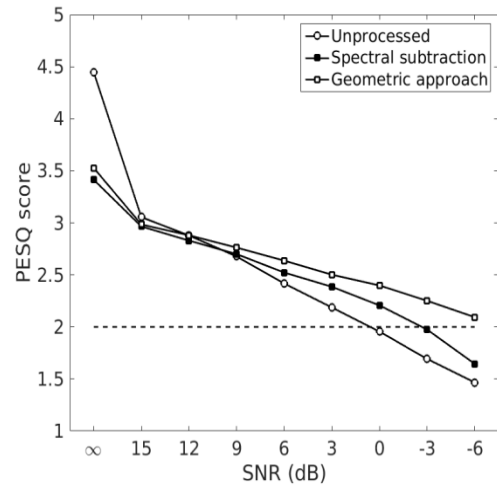
(a) Noise: airport



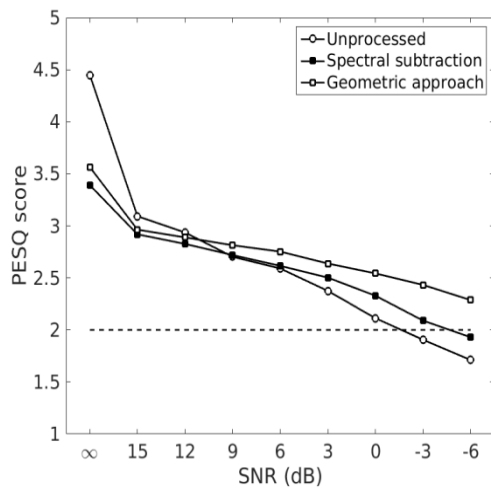
(b) Noise: babble



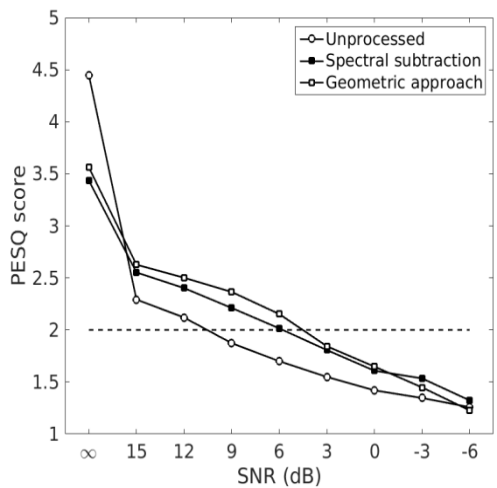
(c) Noise: car



(d) Noise: street

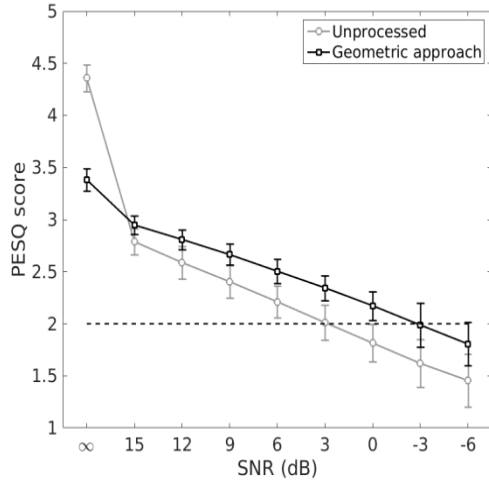


(e) Noise: train station

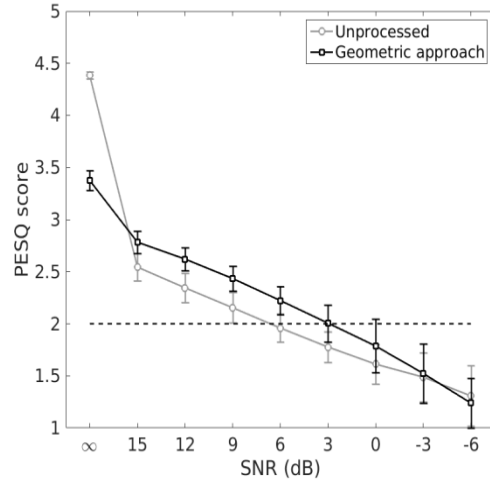


(f) Noise: white

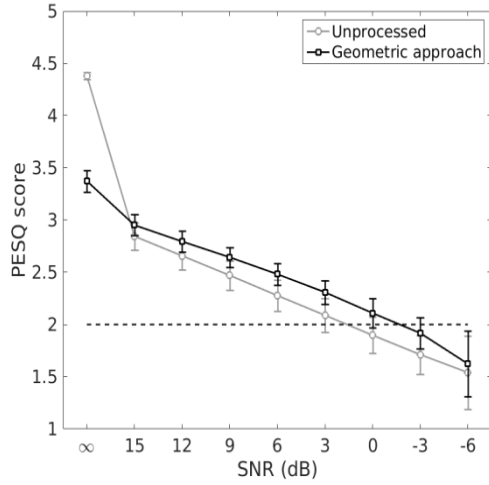
Figure 5.3 PESQ scores for the enhanced noisy speech, speech: VHSES, enhancement methods: spectral subtraction and geometric approach, processing parameters: $\alpha = 2$, $\beta = 0.01$, $\gamma = 1$, and $\lambda = 1/256$



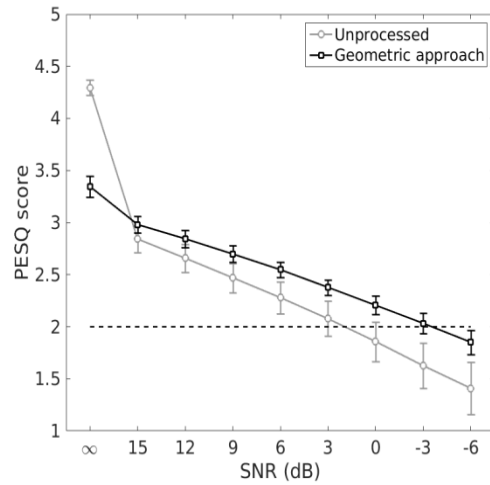
(a) Noise: airport



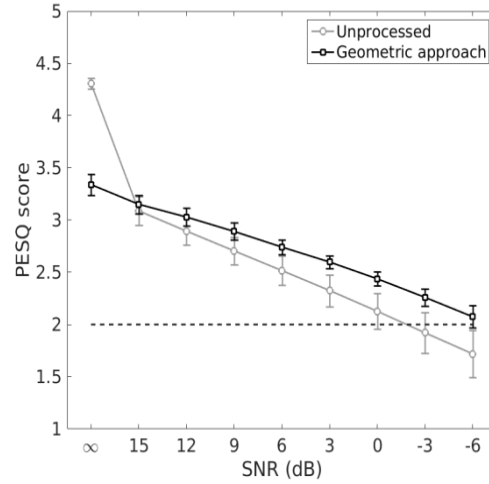
(b) Noise: babble



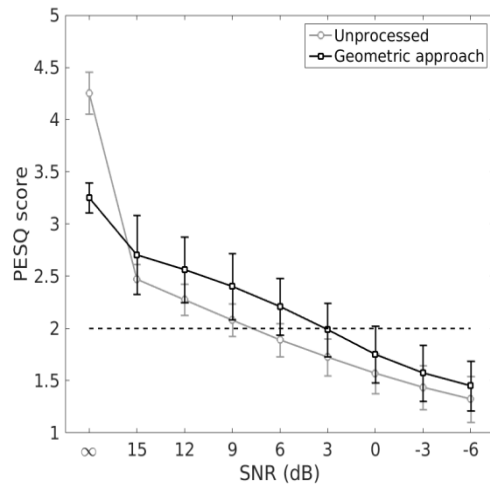
(c) Noise: car



(d) Noise: street



(e) Noise: train station



(f) Noise: white

Figure 5.4 PESQ scores for the enhanced noisy speech, speech: NOIZEUS, enhancement method: geometric approach, processing parameter: $\lambda = 1/256$

Table 5.1 SNR improvement (dB) for PESQ score of 2 using spectral subtraction (SS) and geometric approach (GA) for different types of noises from AURORA database; speech: VHSES.

Noise	Improvement	
	SS	GA
Airport	1	6
Babble	0	4
Car	5	>10
Street	4	>10
Train	3	>6
White	4.5	6

Table 5.2 Improvement in PESQ scores by noise suppression using dynamic quantile tracking and geometric approach for different types of noises; speech: 30 sentences from NOIZEUS.

Airport noise					Babble noise				
Input	Unprocessed		Improvement		Input	Unprocessed		Improvement	
SNR	Mean	S. D.	Mean	S. D.	SNR	Mean	S. D.	Mean	S. D.
6 dB	2.21	0.15	0.29	0.16	6 dB	1.96	0.13	0.26	0.14
3 dB	2.01	0.17	0.33	0.16	3 dB	1.78	0.15	0.23	0.20
0 dB	1.81	0.18	0.35	0.18	0 dB	1.61	0.19	0.17	0.24

Car noise					Street noise				
Input	Unprocessed		Improvement		Input	Unprocessed		Improvement	
SNR	Mean	S. D.	Mean	S. D.	SNR	Mean	S. D.	Mean	S. D.
6 dB	2.28	0.15	0.20	0.13	6 dB	2.28	0.15	0.27	0.15
3 dB	2.09	0.16	0.22	0.15	3 dB	2.08	0.17	0.30	0.16
0 dB	1.90	0.17	0.21	0.18	0 dB	1.86	0.19	0.35	0.17

Train station noise					White noise				
Input	Unprocessed		Improvement		Input	Unprocessed		Improvement	
SNR	Mean	S. D.	Mean	S. D.	SNR	Mean	S. D.	Mean	S. D.
6 dB	2.52	0.14	0.22	0.14	6 dB	1.89	0.15	0.32	0.26
3 dB	2.33	0.15	0.27	0.14	3 dB	1.73	0.18	0.26	0.24
0 dB	1.92	0.19	0.34	0.17	0 dB	1.57	0.19	0.18	0.26

quality. The processing resulted in SNR advantage of 0 – 5 dB for spectral subtraction and 6 – 10 dB for geometrical approach.

The processing is also tested using the 30 sentences from NOIZEUS database [61] added with noises from AURORA database. As the geometric approach performs better than

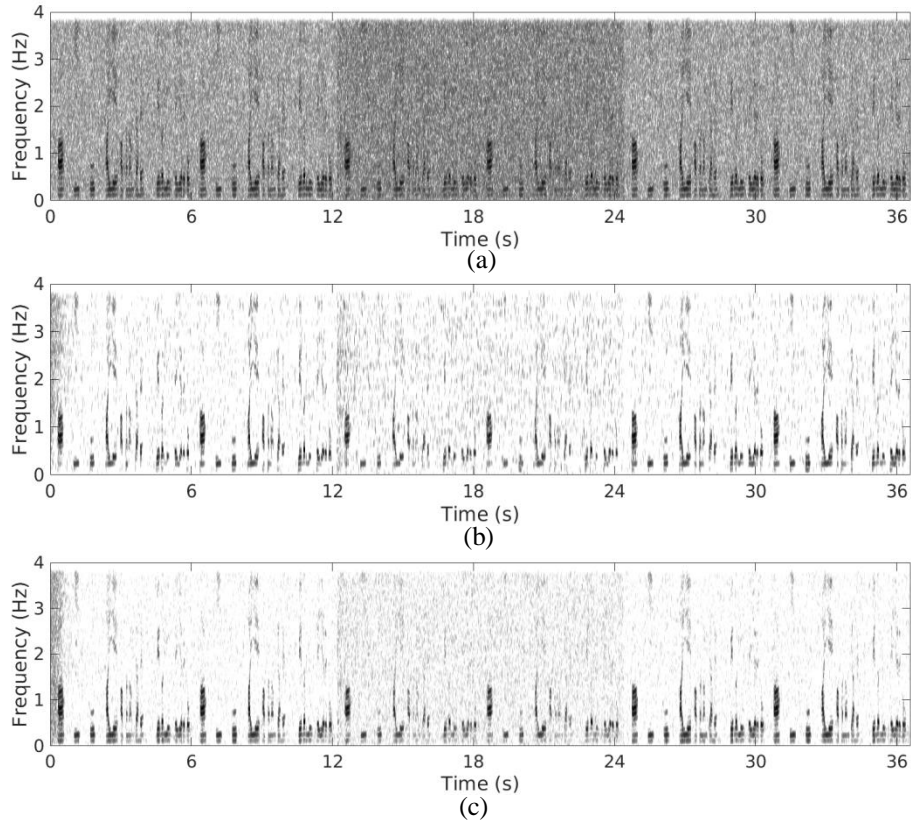


Figure 5.5 Non stationary noise suppression using dynamic quantile tracking based speech enhancement. (a) unprocessed noisy signal with modulated white noise, (b) processed using spectral subtraction, and (c) processed using geometric approach

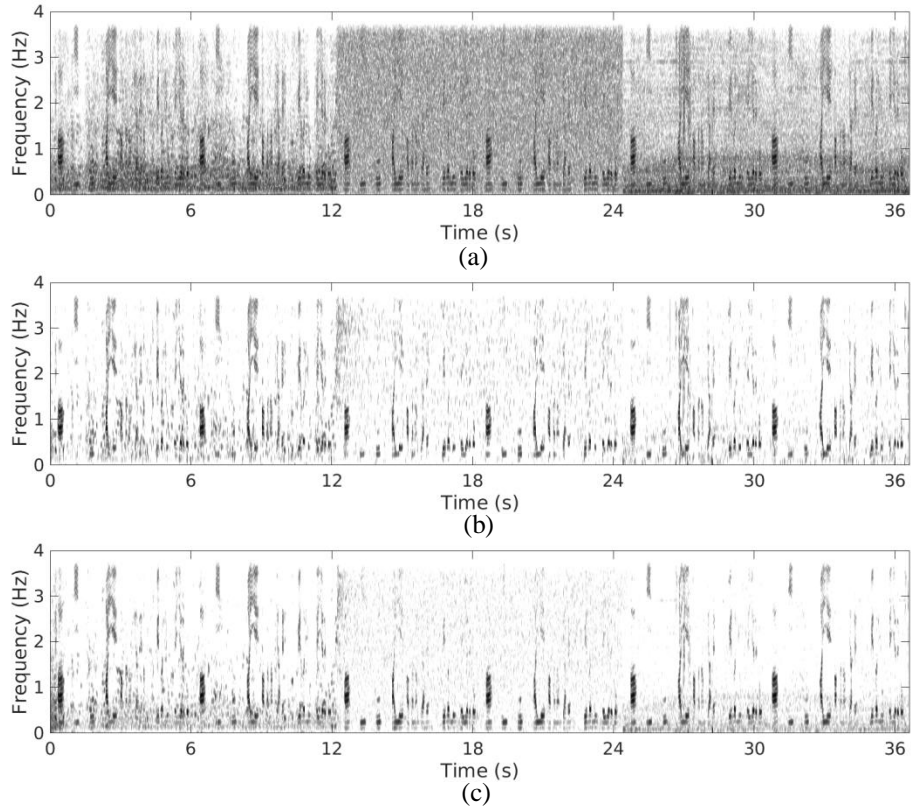


Figure 5.6 Non stationary noise suppression using dynamic quantile tracking based speech enhancement. (a) unprocessed noisy signal with babble, white and car noise, (b) processed using spectral subtraction, and (c) processed using geometric approach

spectral subtraction, the processing is carried out only using geometric approach. Figure 5.4 shows the PESQ scores along with the standard-deviation based error plot. It shows an SNR improvement of 3 – 6 dB at PESQ score of 2. Table 5.2 shows the PESQ improvement for different noises.

The processing is tested for various non-stationary noise conditions. Figure 5.5 shows spectra for processing of speech added with amplitude modulated white noise. VHSES speech segment is repeatedly concatenated and added to modulated white noise resulting to SNR varying from -6 dB to 0 dB. The spectral subtraction method results in residual musical noise which can be seen as dark speckles on spectrogram. Figure 5.6 shows an example for varying noise environment. VHSES speech segment is repeatedly concatenated and added to babble, white and car noise at 0 dB SNR. The spectra show adaptation of noise estimation with different noise environment.

The average time taken for computation per frame is measured as the difference of the average CPU time taken per block with and without noise suppression module. The average CPU time taken per block for noise suppression with spectral subtraction and geometric approach is measured to be 0.75 and 1.1 ms respectively. This corresponds to average CPU time overhead of 0.3 and 0.65 ms for spectral subtraction and geometric approach, respectively.

5.6 Summary

Noise suppression using dynamic quantile tracking based noise estimation is implemented as a processing module in the hearing aid app. The processing has been validated by informal listening and objective evaluation using PESQ. An SNR advantage of 3 – 6 dB is achieved by speech enhancement using noise suppression module. The processing time per block is measured to be 0.3 ms and 0.65 ms for spectral subtraction and geometric approach, respectively.

Chapter 6

SUMMARY AND CONCLUSION

A digital hearing aid has been implemented as a smartphone app for improving speech perception by persons with sensorineural hearing loss. Background noise suppression is implemented to reduce the effect of external spectral masking and speech audibility. Dynamic quantile tracking based noise estimation along with generalized spectral subtraction and geometric approach to spectral subtraction are implemented for noise suppression. Sliding-band dynamic range compression technique is incorporated to alleviate the problems of frequency dependent elevation of hearing thresholds and loudness recruitment. It provides user-configurable frequency selective amplification and dynamic range compression. The implementation enables the user to configure the processing parameters in an interactive and real-time mode using a graphical touch interface.

The app has been developed and tested using Nexus 5X with Android 7.1 OS. The audio latency of the application is measured to be 45 ms, making it suitable for face-to-face communication. The app is tested by informal hearing and objective evaluation. The sliding-band compression does not introduce perceptible distortions associated with the single-band and multi-band compression techniques. Background noise suppression is evaluated by using PESQ scores. Enhancement of speech corrupted with different types of additive stationary and non-stationary noise showed improvement in speech quality to be equivalent to an SNR advantage of 3 – 6 dB.

The average CPU time per frame taken by the processor thread for audio loopback (inclusive of FFT and IFFT operations) without any processing modules is measured to be 0.45 ms. The average CPU time for compression is measured to be 0.85 ms. The average CPU time for processing is measured to be 0.3 and 0.65 ms for spectral subtraction and geometric approach respectively.

The average CPU time for processing of a 5 ms frame with both compression and noise suppression modules does not exceed 2 ms. Hence, there is scope for incorporating other processing techniques for speech enhancement such as CVR enhancement [33], dichotic presentation [36], multiband frequency compression [31] etc., for extending the usability of the app.

Use of the app by a large number of hearing impaired listeners is needed for its real life evaluation and its further enhancement. VoIP (Voice over internet protocol) feature needs to be implemented in the app to enable usage of hearing aid during voice calls.

APPENDIX A

User Manual for ‘Hearing Assistant’





"Hearing Assistant" enables the use of the mobile handset as a hearing aid without needing any specialized hardware other than its headset. To enable comfortable hearing, the app consists of the following processing modules:

- **Noise suppression:** It decreases the interference of external noise in the input speech and acts as a pre-processing module for compression.
- **Compression:** It maps the input sound level into the dynamic range set interactively.

The operation is controlled through homescreen. Each module has its ‘settings’ screen.

Homescreen

The playback controls of app and processing modules are available in the homescreen. The following table provides functions of each button in homescreen

	Starts the hearing aid playback.
	Stops the hearing aid playback.
	Toggle the processing module on/off.
	Opens the settings page of the processing module.

Noise Suppression Settings






The noise suppression settings consist of parameters for speech enhancement using spectral subtraction. The user-settable parameters are defined as following:

Suppression Level	Extent of spectral subtraction. Higher value leads to increased noise reduction at a cost of attenuation of speech.
Noise Floor	The noise floor to reduce annoyance due to musical noise generated during suppression of background noise.

The noise suppression settings is disabled in ‘Hearing Assistant 1.2.2’. The speech enhancement is carried out using geometric approach based spectral subtraction, which does not need any user-set parameter.

Compression Settings

The compression module settings consist of curves to set frequency-dependent hearing levels, along with buttons to store/load settings and undo/redo button. The curves represent the output sound levels with respect to frequency. They can be changed by dragging the thumb points (circles on the curve) to the required position. It is advised to get the initial settings done by an audiologist. The following table describes the functions of the UI elements.

	Sets the level at which the soft sounds need to be played. Increase in value leads to increased gain.
	Sets the comfortable level at which the sounds are comfortably heard.
	Sets the maximum level at which loud sounds need to be played.
	Saves the current settings into the selected memory store.
	Loads the previously stored curves into the current settings.

The 'In' and 'Out' indicator levels show the total input power and output power in dB. The current sliding band smoothed input spectrum is overlaid on the curves. The app can be used in background by starting the app and switching to other apps.

REFERENCES

- [1] H. Levitt, J. M. Pickett, and R. A. Houde, Eds., *Sensory Aids for the Hearing Impaired*, New York: IEEE Press, 1980, pp. 3–10.
- [2] J. M. Pickett, *The Acoustics of Speech Communication: Fundamentals, Speech Perception Theory, and Technology*, Boston, Massachusetts: Allyn Bacon, 1999, pp. 289–323.
- [3] B. C. J. Moore, *An Introduction to the Psychology of Hearing*, London, UK: Academic, 1997, pp. 66–107.
- [4] S. A. Gelfand, *Hearing: An Introduction to Psychological and Physiological Acoustics*, New York: Marcel Dekker, 1998, pp. 314–318.
- [5] D. O' Shaughnaessy, *Speech Communications: Human and Machine*, Hyderabad, India: Univ. Press, 2001, pp. 127–128.
- [6] H. Dillon, *Hearing Aids*. New York: Thieme Medical, 2001.
- [7] N. Tiwari and P. C. Pandey, "A sliding-band dynamic range compression for use in hearing aids," in *Proc. 20th Nat. Conf. Commun. (NCC 2014)*, Kanpur, India, 2014, doi: 10.1109/NCC.2014.6811300.
- [8] S. F. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 27, no. 2, pp. 113–120, 1979.
- [9] N. Tiwari and P. C. Pandey, "Speech enhancement using noise estimation based on dynamic quantile tracking for hearing impaired listeners," in *Proc. 21st Nat. Conf. Commun. (NCC 2015)*, Mumbai, India, 2015, doi: 10.1109/NCC.2015.7084849.
- [10] L. D. Braida, N. I. Durlach, and R. P. Lippmann, *Hearing aids - a review of past research on linear amplification, amplitude compression, and frequency lowering*, J. C. Hardy, Ed. Rockville, MA: ASHA Monographs, 1979. [Online]. <http://www.asha.org/uploadedFiles/publications/archive/Monographs19.pdf>
- [11] R. P. Lippmann, L. D. Braida, and N. I. Durlach, "Study of multichannel amplitude compression and linear amplification for persons with sensorineural hearing loss," *J.*

Acoust. Soc. Am., vol. 69, no. 2, pp. 524–534, 1981.

- [12] N. Tiwari, P. C. Pandey, and A. Sharma, "A smartphone app-based digital hearing aid with sliding-band dynamic range compression," in *22nd Proc. Nat. Conf. Commun. (NCC 2016)*, Guwahati, India, 2016, doi: 10.1109/NCC.2016.7561146.
- [13] P. C. Loizou, *Speech enhancement: Theory and Practice*. Boca Raton, Florida: CRC Press, 2007.
- [14] K. Paliwal, K. Wojcicki, and B. Schwerin, "Single-channel speech enhancement using spectral subtraction in the short-time modulation domain," *Speech Commun.*, vol. 52, no. 5, pp. 450–475, 2010.
- [15] Y. Lu and P. C. Loizou, "A geometric approach to spectral subtraction," *Speech Commun.*, vol. 50, pp. 453–466, 2008.
- [16] S. Tanyer and H. Ozer, "Voice activity detection in non-stationary noise," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 4, pp. 478–482, July 2000.
- [17] R. Martin, "Spectral subtraction based on minimum statistics," in *Proc. 7th Eur. Signal Process. Conf. (EUSIPCO-94)*, Edinburgh, 1994, pp. 1182–1185.
- [18] G. Doblinger, "Computationally efficient speech enhancement by spectral minima tracking in subbands," in *Proc. 4th Eur. Conf. Speech Commun., Technol. (EUROSPEECH)*, Madrid, Spain, 1995, pp. 1513–1516.
- [19] L. Lin, W. H. Holmes, and E. Ambikairajah, "Adaptive noise estimation algorithm for speech enhancement," *Electronic Letters*, vol. 39, no. 9, pp. 754–755, May 2003.
- [20] H. Hirsch and C. Ehrlicher, "Noise estimation techniques for robust speech estimation," in *Proc. Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, Detroit, 1995, pp. 153–156.
- [21] I. Cohen, "Noise spectrum estimation in adverse environments: improved minima controlled recursive averaging," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 5, pp. 466–475, 2003.
- [22] V. Stahl, A. Fisher, and R. Bopus, "Quantile based noise estimation for spectral subtraction and Wiener filtering," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal*

Proc. (ICASSP), Istanbul, Turkey, 2000, pp. 1875–1878.

- [23] N. W. Evans and J. S. Mason, "Time-frequency quantile based noise estimation," in *Proc. 11th Eur. Signal Process. Conf (EUSPICO 2002)*, Toulouse, France, 2002, pp. 539-542.
- [24] H. Bai and E. A. Wan, "Two-pass quantile based noise spectrum estimation," Center of spoken language understanding, OGI School of Science and Engineering at OHSU, 2003. [Online]. https://www.researchgate.net/profile/Eric_Wan2/publication/2556842_Two-Pass_Quantile_Based_Noise_Spectrum_Estimation/links/54282e4e0cf238c6ea7cd273.pdf
- [25] S. K. Waddi, P. C. Pandey, and N. Tiwari, "Speech enhancement using spectral subtraction and cascaded-median based noise estimation for hearing impaired listeners," in *Proc. 19th Nat. Conf. Commun. (NCC 2013)*, New Delhi, India, 2013, doi : 10.1109/NCC.2013.6487989.
- [26] P. C. Pandey, S. S. Pratapwar, and P. K. Lehana, "Enhancement of electrolaryngeal speech by reducing leakage noise using spectral subtraction with quantile based dynamic estimation of noise," in *Proc. 18th Int. Congress on Acoustics (ICA 04)*, Kyoto, Japan, 2004, pp. 3029–3032.
- [27] C. Ris and S. Dupont, "Assessing local noise level estimation methods: application to robust noise ASR," *Speech Commun.*, vol. 34, no. 1–2, pp. 141–158, 2001.
- [28] R. J. McAulay and M. L. Malpass, "Speech enhancement using a soft-decision noise suppression filter," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 28, no. 2, pp. 137–145, 1980.
- [29] P. C. Pandey and N. Tiwari, "Method and system for suppressing noise in speech signals in hearing aids and speech communication devices," US Patent application publication no. US20170032803 A1, 2017.
- [30] K. Yasu, M. Hishitani, T. Arai, and Y. Murahara, "Critical-band based frequency compression for digital hearing aids," *Acoustical Science and Technology*, vol. 25, no. 1, pp. 61–63, 2004.
- [31] P. N. Kulkarni, P. C. Pandey, and D. S. Jangamashetti, "Multi-band frequency compression for reducing the effects of spectral masking," *Int. J. Speech Tech.*, vol. 10,

pp. 219–227, 2009.

- [32] N. Tiwari, P. C. Pandey, and P. N. Kulkarni, "Real-time implementation of multi-band frequency compression for listeners with moderate sensorineural impairment," in *Proc. INTERSPEECH*, Portland, Oregon, 2012.
- [33] A. R. Jayan and P. C. Pandey, "Automated modification of consonant-vowel ratio of stops for improving speech intelligibility," *Int. J. Speech Technol.*, vol. 18, pp. 113–130, 2015.
- [34] P. E. Lyregaard, "Frequency selectivity and speech intelligibility in noise," *Scand. Audiol. Suppl.*, vol. 15, pp. 113–122, 1982.
- [35] T. Lunner, S. Arlinger, and J. Hellgren, "8-channel digital hearing aid for hearing aid use: preliminary results in monaural, diotic, and dichotic modes," *Scand. Audiol. Suppl.*, vol. 38, pp. 75–81, 1993.
- [36] A. N. Cheeran, "Speech processing with dichotic presentation for binaural hearing aids for moderate bilateral sensorineural loss," Ph.D. Thesis, IIT Bombay, Mumbai, 2005.
- [37] P. N. Kulkarni, "Speech processing for reducing the effects of spectral masking in sensorineural hearing loss," Ph. D. Thesis, IITB, Mumbai, 2010.
- [38] IDC. Smartphone OS Market Share, 2016 Q2. [Online]. Available: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>.
- [39] Superpowered Inc. iOS and Android Audio Latency Test App. [Online]. <http://superpowered.com/latency>
- [40] F. Larrosa et al., "Development and evaluation of an audiology app for iPhone/iPad mobile devices," *Acta Oto-Laryngologica*, vol. 135, no. 11, pp. 1119–1127, 2015.
- [41] S Abu-Ghanem et al., "Smartphone-based audiometric test for screening hearing loss in the elderly," *Eur Arch Otorhinolaryngol*, 2015.
- [42] e-audiologia. Hearing Test. [Online]. [https:// play.google.com/store/apps/details?id=mobile.eaudiologia](https://play.google.com/store/apps/details?id=mobile.eaudiologia)
- [43] Quadrio Devices. Q+ hearing aid. [Online]. [https:// play.google.com/store/apps/details?](https://play.google.com/store/apps/details?)

id=com.quadiodevices.qplus

[44] Newbrick S. A. uSound (Hearing Assistant). [Online]. [https:// play.google.com/store/apps/details? id=com.newbrick.usound](https://play.google.com/store/apps/details?id=com.newbrick.usound)

[45] IT4YOU. Petralex hearing aid. [Online]. [https:// play.google.com/store/apps/details? id=com.it4you.petralex](https://play.google.com/store/apps/details?id=com.it4you.petralex)

[46] GN Resound. ReSound Control. [Online]. [https:// play.google.com/store/apps/details? id=com.gnresound.remotecontrol](https://play.google.com/store/apps/details?id=com.gnresound.remotecontrol)

[47] Phonak. Phonak remote control app. [Online]. [https:// play.google.com/store/apps/details? id=com.phonak.mobileapps.rcapp](https://play.google.com/store/apps/details?id=com.phonak.mobileapps.rcapp)

[48] Unitron. Unitron uControl. [Online]. [https:// play.google.com/store/apps/details? id=com.unitron.mobileapps.development.rcapp](https://play.google.com/store/apps/details?id=com.unitron.mobileapps.development.rcapp)

[49] Sivantos Pte. Ltd. touchControl. [Online]. [https:// play.google.com/store/apps/details? id=com.hearing.healthcare.siemens.touchcontrol](https://play.google.com/store/apps/details?id=com.hearing.healthcare.siemens.touchcontrol)

[50] Lemberg Solutions. Hearing Aid with Replay (Lite). [Online]. [https:// play.google.com/store/apps/details? id=com.ls.soundamplifier](https://play.google.com/store/apps/details?id=com.ls.soundamplifier)

[51] D Byrne and W. Tonnison, "Selecting the gain of hearing aids for persons with sensorineural hearing impairments," *Scand. Audiol.*, vol. 5, pp. 51–59, 1976.

[52] G. A. McCandless and P. E. Lyregaard, "Prescription of gain/output (POGO) for hearing aids," *Hear Instrum.*, vol. 34, no. 1, pp. 16–21, 1983.

[53] International Telecommunication Union, "Relative Timing of Sound and Vision for Broadcasting," 1998.

[54] Google Inc. Nexus 5X. [Online]. https://www.google.co.in/intl/en_in/nexus/5x/

[55] Khronos Group. Open SLES. [Online]. <http://www.khronos.org/opensles>

[56] D. W Griffin and J. S Lim, "Signal estimation from modified short-time fourier transform," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 32, no. 2, pp. 236–243, 1984.

- [57] M. Borgerding. (2013) Kiss FFT. [Online]. <https://sourceforge.net/projects/kissfft/>
- [58] F. N. Fritsch and R. E. Carlson, "Monotone piecewise cubic interpolation," *SIAM J. Numer. Anal.*, vol. 17, no. 2, pp. 238–246, 1980.
- [59] International Telecommunication Union , "Perceptual evaluation of speech quality (PESQ): an objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," ITU-T Rec., P.862, 2001.
- [60] H. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *INTERSPEECH*, Beijing, China, 2000, pp. 181-188.
- [61] Y. Hu and P. Loizou, "Subjective comparison of speech enhancement algorithms," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, Toulouse, France, 2006, pp. 153-156.

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude and respect to my guide Prof. P. C. Pandey, for his invaluable guidance and support during this project. I am thankful to him for encouraging me in taking up new challenges and motivating me in every phase of the project. He has always inspired me by his curiosity, wisdom, commitment, and strive for perfection.

I am deeply indebted to Nitya Tiwari, for her constant guidance and support throughout the implementation of this project. I am thankful to her for helping me understand various concepts of signal processing and for patiently answering all my doubts. I am thankful to Nataraj and Hirak for sharing interesting discussions with me and providing support whenever needed. I would like to thank Vidyadhar Kamble for helping me in all the lab related issues. I would like to thank my labmates Prachir, Vikas, Susmi, Shibam, and Yogesh for their help in various aspects of this project and for creating a joyful work environment. I would also like to thank my friends Hitesh, Rajesh, Prakhar, Ratin, and Hemant for their support and encouragement.

Finally, I am thankful to my parents and my sister for their unconditional love, encouragement, and support.

Saketh Sharma

June 2017

Left blank

AUTHOR'S RESUME

Saketh Sharma: The author received B. E. degree in telecommunication engineering from R. V. College of Engineering, Bengaluru (Karnataka), affiliated to Visvesvaraya Technological University, Belagavi, in 2013. Presently, he is pursuing the M. Tech. degree in electrical engineering at the Indian Institute of Technology Bombay. His research interests include speech processing, digital signal processing, and embedded system design.

Thesis related publication

S. Sharma, N. Tiwari, and P. C. Pandey, "Implementation of a digital hearing aid with user-settable frequency response and sliding-band dynamic range compression as a smartphone app," in *Proc. Int. Conf. Intelligent Human Computer Interaction 2016 (IHCI 2016)*, Pilani, India, 2016, doi : 10.1007/978-3-319-52503-7_14.