Implementation of a Digital Hearing Aid with User-Settable Frequency Response and Sliding-Band Dynamic Range Compression as a Smartphone App

Saketh Sharma, Nitya Tiwari, and Prem C. Pandey

Department of Electrical Engineering, Indian Institute of Technology Bombay, Mumbai, India
{sakethsharma,nitya,pcpandey}@ee.iitb.ac.in

Abstract. Persons with sensorineural hearing loss suffer from degraded speech perception caused by frequency-dependent elevation of hearing thresholds, reduced dynamic range and abnormal loudness growth, and increased temporal and spectral masking. For improving speech perception by persons with moderate loss of this type, a digital hearing aid is implemented as an Android-based smartphone app. It uses sliding-band dynamic range compression for restoring normal loudness of low-level sounds without making the high-level sounds uncomfortably loud and for reducing the perceptible temporal and spectral distortions associated with currently used single and multiband compression techniques. The processing involves application of a frequency dependent gain function calculated on the basis of critical bandwidth based short-time power spectrum and is realized using FFT-based analysis-synthesis. The implementation has a touch-controlled graphical user interface enabling the app user to fine tune the frequency-dependent parameters in an interactive and real-time mode.

Keywords. Dynamic range compression · Hearing aid · Interactive settings · Sensorineural hearing loss · Smartphone app.

1 Introduction

Abnormalities in the sensory hair cells or the auditory nerve result in sensorineural hearing loss. It is a commonly occurring hearing loss, which may be inherited genetically or may be caused by excessive noise exposure, aging, infection, or use of ototoxic drugs. It is associated with frequency-dependent elevation of hearing thresholds, reduced dynamic range along with abnormal growth of loudness known as loudness recruitment, and increased temporal and spectral masking leading to degraded speech perception [1, 2, 3]. Hearing aids used for alleviating the effects of sensorineural hearing loss provide frequency-selective amplification along with dynamic range compression [4, 5, 6, 7, 8]. They may have directional microphones for improving the signal-to-noise ratio and may employ signal processing for suppression of wind and background noise, acoustic feedback cancellation, environment learning for selection of the frequency response, and generation of comfort noise for patients with tinnitus. Some aids also provide facility of customization using mobile apps and wireless connectivity.

Frequency-dependent elevation of hearing thresholds occurs due to loss of inner hair cells and leads to reduced audibility of speech. Loss of outer hair cells results in reduced

dynamic range, i.e. increase in hearing threshold levels without corresponding increase in upper comfortable levels. It also distorts the loudness relationships among speech components leading to degraded speech perception [2, 3]. As linear amplification of sounds is unsuitable for persons with sensorineural hearing loss, hearing aids provide amplification along with dynamic range compression to compensate for reduced dynamic range. The gains and compressions are usually set by an audiologist based on the patient's audiogram and in accordance with a hearing aid fitting protocol [4, 6, 7].

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 [4, 5, 8, 9, 10, 11]. The processing is carried out as single-band compression or as multiband compression. In single band compression, the gain is calculated as a function of the signal power over its entire bandwidth. 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 currently available hearing aids provide multiband 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. It avoids the problems associated with single band compression, but results in decreased spectral contrasts and modulation depths 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 currently used in the hearing aids, a sliding-band compression technique has been reported by Tiwari and Pandey [12, 13] 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.

Hearing aids are designed using ASICs due to power and size constraints. Therefore, incorporation of a new compression technique in hearing aids and its field evaluation is prohibitively expensive. Use of smartphone-based application software (app) to customize and remotely configure settings on hearing aids provide a greater flexibility to hearing aid users and developers. Many hearing aid manufacturers (GN ReSound, Phonak, Unitron, Siemens, etc) provide apps to control hearing aids using Android or iOS smartphone. This type of app helps the hearing aid user in personalizing the listening experience by adjustment of settings during use of the device and avoids repeated visits to an audiology clinic. The smartphone-based apps may also be used for development and testing of signal processing techniques for hearing aids. Hearing aid apps [15, 16, 17, 18, 19, 20] such as 'Petralex', 'uSound', 'Q^{+}', and 'BioAid' for Android/iOS, ‘Mimi’, 'EnhancedEars' for iOS, and "Hearing Aid with Replay" and “Ear Assist” for Android provide users with moderate sensorineural hearing loss a low-cost alternative for hearing aids. In addition to providing frequency-selective gain and multiband dynamic range compression, they also offer the flexibility of creating and storing sound profiles specific to the user's hearing loss characteristics.

Implementation of the sliding-band compression technique for real-time processing using a 16-bit fixed-point DSP chip with on-chip FFT hardware has been reported in

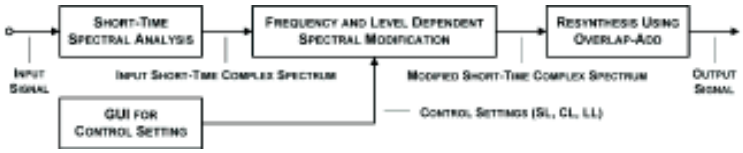

Fig. 1. Signal processing in the hearing aid app with settable frequency response and dynamic range compression.

[12] and its implementation as an iPhone-based hearing aid app has been reported in [14]. However, these implementations use predefined processing parameters. To make this low-distortion compression technique conveniently available to the hearing impaired persons and to permit its evaluation by a large number of users without incurring the expenses involved in the ASIC-based hearing aid development, a smartphone app implementing the digital hearing aid with user settable frequency response and dynamic range compression is needed. Such an app is developed for providing the user flexibility to fine tune the frequency-dependent parameters in an interactive and real-time mode by a touch-controlled graphical user interface (GUI). The implementation is carried out using an Android-based smartphone handset, as such handsets are available from a number of manufacturers with features and cost suiting a large number of users. The handset model 'Nexus 5X' with Android 7.0 is used for implementation and testing on the basis of its relatively low audio I/O (input-output) delay and high processing capacity. Most handset models may meet these criteria in the near future.

The signal processing for the app is described in the second section. The third section presents the details of its implementation as a smartphone app. Test results are presented in the fourth section, followed by conclusions in the last section.

2 Signal Processing for Settable Frequency Response and Sliding-Band Dynamic Range Compression

Block diagram of the signal processing for the app is shown in Fig. 1. The operations for compensation of increased hearing thresholds and decreased dynamic range are carried out by spectral modification. The processing uses a DFT-based analysis-synthesis comprising the steps of short-time spectral analysis, frequency and level dependent spectral modification, and signal resynthesis. The analysis involves segmentation of the input signal into overlapping frames and calculating DFT to get short-time complex spectra. A frequency-dependent gain function is calculated in accordance with the control settings and as a function of the short-time magnitude spectrum. This gain function is used for modification of the short-time input complex spectrum and its IDFT is calculated to get the overlapping output signal frames. The output signal is re-synthesized using an overlap-add technique for reducing the processing related artifacts.

In our design, it is possible to use a look-up table to provide the gain function most appropriate for improving speech perception by the listener. However, assessment of the loudness growth function as a function of frequency to derive the look-up table is not practical. Hence, the control settings are in the form of plots of desired levels for 'soft', 'comfortable', and 'loud' sounds (referred to as SL, CL, LL, respectively) and

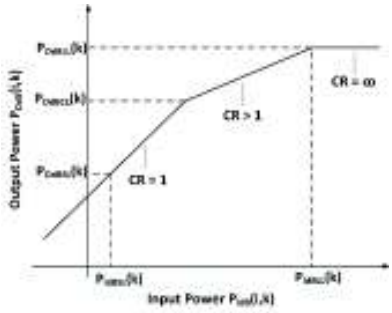

Fig. 2. Relation between input power (dB) and output power (dB) for *i*th frame and band centered at *k*th spectral sample.

input using touch-controlled GUI in an interactive manner. These values are used for calculating the frequency and level dependent gain function. For each frequency 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 Fig. 2. 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 Fig. 2, corresponds to linear gain providing the amplification needed for soft level sounds and the value of the gain is given as

$$G_{LdB}(k) = P_{OdBSL}(k) - P_{IdBSL}(k) \tag{1}$$

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:

$$P_{OdB}(i,k) = P_{IdB}(i,k) + G_{LdB}(k), \quad P_{IdB}(i,k) < P_{OdBCL}(k) - G_{LdB}(k) \tag{2}$$

The central segment of the relationship, marked as 'CR > 1', corresponds to compression with compression ratio $\mathrm{CR}(k)$ given as

$$\mathrm{CR}(k) = \{P_{IdBLL}(k) - P_{OdBCL}(k) + G_{LdB}(k)\} / \{P_{OdBLL}(k) - P_{OdBCL}(k)\} \tag{3}$$

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

$$P_{OdB}(i,k) = P_{OdBCL}(k) + \{P_{IdB}(i,k) - P_{OdBCL}(k) + G_{LdB}(k)\} / \mathrm{CR}(k), \quad P_{OdBCL}(k) - G_{LdB}(k) \le P_{IdB}(i,k) \le P_{IdBLL}(k) \tag{4}$$

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:

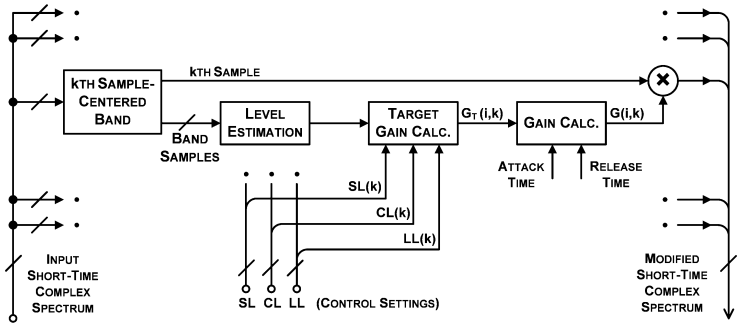

Fig. 3. Spectral modification for compensation of increased hearing thresholds and decreased dynamic range using sliding-band dynamic range compression.

$$P_{OdB}(i,k) = P_{OdBLL}(k), \quad P_{IdB}(i,k) > P_{IdBLL}(k) \tag{5}$$

In accordance with the input-output relationship as represented by the three segments and given in Eqs. 2, 4, and 5, the target gain for the spectral sample *k* in *i*th frame is given as:

$$G_{TdB}(i,k) = \begin{cases} G_{LdB}(k), & P_{IdB}(i,k) < P_{OdBCL}(k) - G_{LdB}(k) \\ \frac{G_{LdB}(k) - \{P_{IdB}(i,k) - P_{OdBCL}(k)\}\{\mathrm{CR}(k)-1\}}{\mathrm{CR}(k)}, & P_{OdBCL}(k) - G_{LdB}(k) \le P_{IdB}(i,k) \le P_{IdBLL}(k) \\ P_{OdBLL}(k) - P_{IdB}(i,k), & P_{IdB}(i,k) > P_{IdBLL}(k) \end{cases} \tag{6}$$

Block diagram of the spectral modification is shown in Fig. 3. For each frequency 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

$$\mathrm{BW}(k) = 25 + 75(1 + 1.4(f(k))^2)^{0.69} \tag{7}$$

where $f(k)$ is the frequency of the *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 Eq. 6, and is given as

$$G(i,k) = \begin{cases} \max(G(i-1,k)/\gamma_a, G_T(i,k)), & G_T(i,k) < G(i-1,k) \\ \min\big(G(i-1,k)\cdot\gamma_r, G_T(i,k)\big), & G_T(i,k) \ge G(i-1,k) \end{cases} \tag{8}$$

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 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 =

$s_a S/f_s$ and release time as $T_r = s_r S/f_s$, 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.

The processing for dynamic range compression modifies the magnitude spectrum and the signal is resynthesized using the original phase spectrum. It results in perceptible distortions due to phase discontinuities in the modified short-time complex spectrum. To mask these distortions, the analysis-synthesis method based on the least squared error estimation (LSEE) as proposed by Griffin and Lim [21] is used. It involves segmenting the input signal using L-sample frames with 75% overlap, i.e. frame shift $S = L/4$, and multiplying the segmented frames with modified-Hamming window. Complex spectrum of L-sample frame is obtained after zero-padding it to length N and calculating N-point FFT. After spectral modification as described earlier, N-point IFFT is calculated to get the modified output frame which is multiplied with the modified-Hamming window. The successive output frames are added in accordance with the overlap of the input frames to provide the resynthesized output signal.

3 App Implementation for Real-Time Processing

A signal bandwidth of 4 kHz is generally considered adequate for hearing aid application, and hence a sampling frequency of 10 kHz may be used. As the detectability threshold for audio-visual delay is reported to be 125 ms [22], the signal delay introduced by the app should be less than this value to avoid audio-visual de-synchrony during face-to-face communication. After examining the processing capability and audio I/O delay of several currently available Android-based handsets [23], implementation of the app for real-time processing has been carried out and tested using 'Nexus 5X' with Android 7.0 Nougat OS. It provides a default audio sampling frequency of 48 kHz which can be decreased using the re-sampler block but with an increase in the I/O delay due to the antialiasing and smoothening filters. Considering the computational load associated with sampling frequency of 48 kHz and increased delay with sampling frequency of 12 kHz, the processing is carried out with a down-sampling ratio of two resulting in sampling frequency of 24 kHz. The FFT-based analysis-synthesis uses 20-ms frames with 75% frame overlap and 1024-point FFT. Thus the implementation uses f_s = 24 kHz, L=480, S=120, and N=1024.

The app "Hearing Assistant 1.0" has been developed with a GUI for control settings, enabling graphical setting of the controls for threshold and range compensations. The screen is used in its landscape mode to make an appropriate use of the aspect-ratio, ignoring the 'auto-rotate' setting. The screenshot of the home screen of the app is shown in Fig. 4. The play/stop button is for on/off control. The two bars at the right side graphically display the input and output levels for assessing the effect of control settings. The graphical control for gain and range compensation is accessed through the 'settings' button of 'compression' module. A provision has been provided to include other processing modules, such as noise suppression [24, 25] and CVR enhancement [26] in the later versions of the app. All modules have on/off and 'settings' buttons. The on/off

Fig. 4. Screenshot of home screen of hearing aid app "Hearing Assistant 1.0"

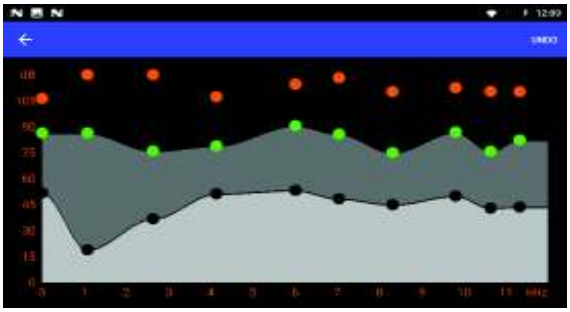

Fig. 5. Screenshot of settings screen to set parameters of frequency dependent gain and dynamic range compression

button can be used for toggling the function and the settings button can be used for setting the processing parameters graphically.

Fig. 5 shows the screenshot of the 'settings' screen for compression. It provides touch control of points, called 'thumbs', on the level vs frequency plots for setting the output levels for soft, comfortable, and loud sounds, for up to 10 frequencies. A thumb can be moved by dragging, freely in the horizontal direction and as constrained by the upper and lower curves in the vertical direction. Movement of a thumb on one curve along x-axis moves the corresponding thumbs on the other two curves, while the movement along y-axis does not affect them. When a touch event occurs, the thumbs on the three curves closer with respect to the x-position of the touch are shifted to the new x-position, and the closest one with respect to the y-position is shifted to the new y-position. This method permits changing a thumb position without dragging it from its earlier position. With each change in the thumb position, the curves for all the intermediate frequency values are obtained by a smooth curve fitting through the thumbs, using Fritsch-Carlson method of monotonic cubic spline fitting [27] to ensure that the curve

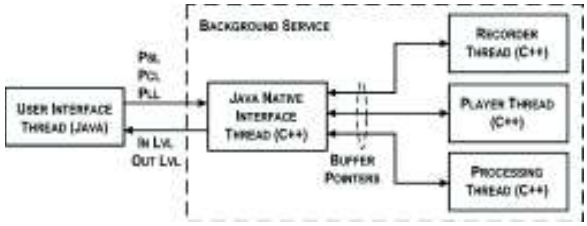

Fig. 6. Block diagram of application process in the hearing aid app.

between two adjacent control points is monotonic. The gains calculated are transferred to the spectral modification block at the next analysis frame, permitting a real-time and interactive touch-control of the processing parameters. The values of input and output levels in each frame are transferred to GUI for updating the level bars.

The program was written using a combination of C++ and Java, with Android Studio 2.1.0 as the development environment. The block diagram of application process of the hearing aid app is shown in Fig. 6. As the audio I/O delay using Java APIs (application programming interfaces) of Android is beyond the acceptable value of 125 ms, the I/O framework is implemented in C++ using Open SLES library [28] which is a hardware-accelerated audio API for input/output handling. An Open SLES audio engine with recording and playback capabilities is realized. Since the application is required to run even when other applications are switched on, the audio engine is maintained using a high priority background service. The service is started when the application comes to foreground and remains running until the playback is stopped or the application process is killed. The communication between GUI and the audio engine is carried out through background service using JNI (Java native interface) as shown in Fig. 6.

A recorder and a player object are realized with required sampling rate and frame length specifications, each with internal queues to manage the buffer blocks currently being recorded/played. Separate threads are spawned for recorder and player to ensure uninterrupted I/O. The recorder and player threads are scheduled at least once within the frame shift duration of 5 ms. Blocking calls to the operating system (OS) such as memory allocation, mutex locks, etc. can cause underruns in real-time applications. To minimize the blocking calls, the memory required to handle I/O and processing are allocated while creating the audio engine. The processing and the signal acquisition and playback are carried out using pointers and no memory allocation is required while the app is running. The new samples from ADC are filled in the *S*-word free buffer of recorder thread. The recorder thread uses 2*S*-word buffer queue and it passes the pointer of *S*-word filled buffer to JNI interface when scheduled. The processing thread when scheduled gets pointer of *S*-word buffer containing the new samples from the JNI interface and returns the pointer of *S*-word buffer containing processed samples. The player buffer when scheduled gets pointer of *S*-word buffer containing processed samples from JNI interface and these samples are output through DAC.

In the processing thread, the input samples, real and imaginary parts of spectral samples, and the output samples are stored as 32-bit floating point arrays. The input samples are acquired in *S*-word blocks from JNI interface. For each *S*-word just-filled

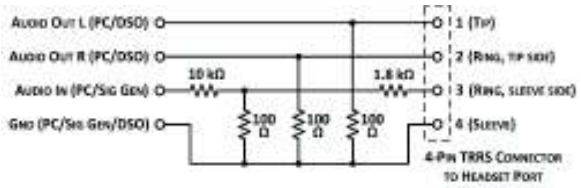

Fig. 7. Audio interface to the 4-pin TRRS headset port of the mobile handset

block, a 3*S*-word buffer stores the previous input samples. Input window with *L* samples (4*S* words) is formed using the samples of the just-filled block and the samples from previous three blocks stored in 3*S*-word buffer. These *L* samples are multiplied by *L*-point modified-Hamming window and *N*-point FFT is calculated. Open source Kiss FFT library [29] has been used for floating point FFT computation. Input power in an auditory critical band centered at each of the first *N*/2 complex spectral components is calculated as given in Eq. 7. Modified spectrum is obtained by multiplying first *N*/2 complex spectral samples with the corresponding gains. The other samples of modified spectrum are obtained by taking complex conjugate of first *N*/2 complex spectral samples. *N*-point IFFT of the modified complex spectrum is calculated and the resulting sequence is multiplied with the modified-Hamming window to get the modified output frame. The output signal is re-synthesized using overlap-add. In the current implementation, the processed signal is output on both the audio channels. Separate graphical settings of the processing parameters for the two channels may be incorporated for use of the app as a binaural hearing aid. The processing has algorithmic delay of one frame, i.e. 20 ms. For real-time output, the processing of each frame has to get completed before the arrival of the next frame, i.e. the frame shift interval of 5 ms, resulting in the signal processing delay, comprising the algorithmic and computational delays, of 20 – 25 ms.

4 Test Results

The app was tested on the handset model 'Nexus 5X' with Android 7.0 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 an audio input-output interface with a 4-pin TRRS (tip-ring-ring-sleeve) connector to the headset port of the handset, as shown in Fig. 7. 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 kΩ for it to be recognized as external microphone. The two output channels have 100 Ω load resistances. A PC sound card was used for applying the audio input and acquiring the processed output. Additional testing was carried out by providing signals from a function generator and observing the waveform using a digital storage oscilloscope.

The settable frequency response without compression was tested using a sinusoidal input of swept-frequency and constant amplitude. An example of the processing along with the frequency-dependent gains set by the GUI is shown in Fig. 8, with the soft level kept high in bands at 1 kHz and 7.5 kHz, and the comfortable level and the loud

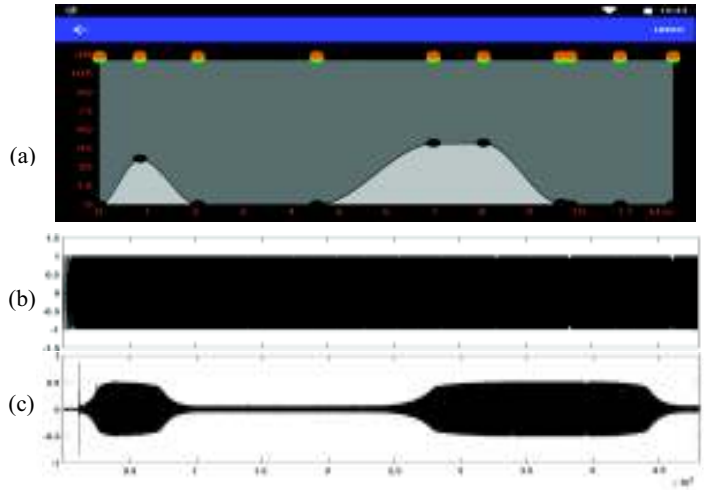

Fig. 8. Example of processing for frequency-dependent linear gain: (a) parameters, (b) input of tone of constant amplitude with frequency linearly swept from 100 Hz to 10 kHz over 10 s, and (c) processed output

level kept same and at the maximum value. The input frequency is linearly swept from 100 Hz to 10 kHz over 10 s. The processed output shows changes in amplitude of the sine wave in accordance with the soft-level curve. The processing for dynamic range compression was tested by setting different values for comfortable and loud levels and amplitude-modulated tones of different frequency and amplitude envelopes as applied inputs. An example of the compression is shown in Fig. 9. The input is an amplitude-modulated tone of 1 kHz. 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 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 Fig. 10, with an English sentence repeatedly concatenated with different scaling factors to observe the effect of variation in the input level on the output waveform. 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. There were no perceptible differences between the processed outputs from the app and the offline implementation. Examples of the processing by the app are available at [30].

The audio latency, i.e. the signal delay comprising the processing delay and I/O delay, was measured by applying a 1 kHz tone burst of 200 ms from a function generator as the input and observing the delay from onset of the input tone burst to the

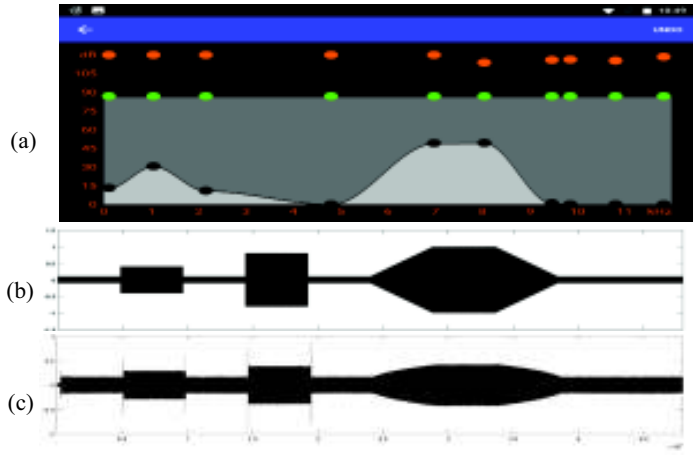

Fig. 9. Example of processing for compression: (a) parameters, (b) input of amplitude-modulated sinusoidal tone of 1 kHz, and (c) processed output

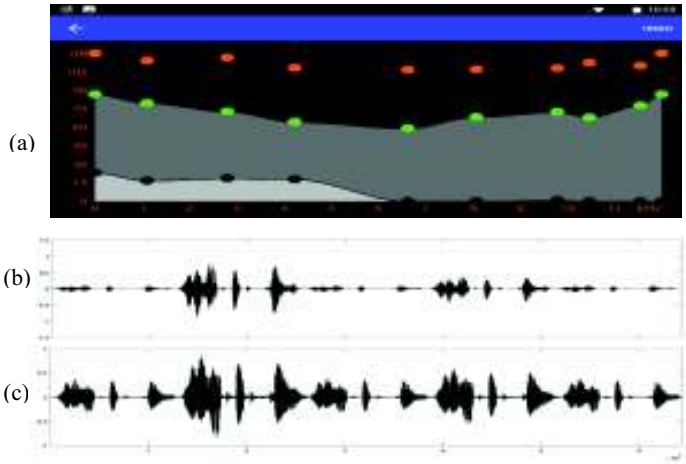

Fig. 10. Example of processing of speech signal: (a) parameters, (b) input speech signal with amplitude modulation (English sentence "*you will mark ut please*" repeatedly concatenated with different scaling factors), and (c) processed output

corresponding onset in the output, using a digital storage oscilloscope. It was found to be approximately 55 ms. The processing delay (sum of the algorithmic and computational delays) is 1.25 times the frame length, i.e. approximately 25 ms. The additional

delay is due to audio I/O delay of the handset hardware, buffering operations in the OS, and delays in the anti-aliasing and smoothening filters. The audio latency of our implementation is much lower than the detectability threshold of 125 ms and hence it should be acceptable for face-to-face communication.

5 Conclusion

A smartphone app has been developed for implementing a digital hearing aid incorporating user settable frequency response and sliding-band dynamic range compression for improving speech perception by persons with moderate sensorineural hearing loss. It is aimed at enabling the hearing of soft-to-loud speech without introducing perceptible distortions. The processing parameters can be set by the user in an interactive and real-time mode using a graphical touch interface. The implementation has been validated by graphical assessment of the processed outputs and informal listening tests. The audio latency of the app is acceptable for audio-visual perception permitting its use as a hearing aid during face-to-face conversation. Use of the app by a large number of hearing impaired listeners is needed for its real life evaluation and further enhancement. Interactive settings and processing for binaural hearing aid may be incorporated in the app. It may be extended as a hearing aid for persons with severe loss by using headsets with appropriate output levels. With further increase in the processing capacity and decrease in audio I/O delay of newer handset models, there is scope for incorporating processing for speech enhancement in the app for extending its usability in noisy environments.

Acknowledgment

The research is supported by “National Program on Perception Engineering Phase-II” sponsored by the Department of Electronics & Information Technology, Government of India.

References

1. Levitt, H., Pickett, J.M., Houde, R.A. (eds.): Sensory Aids for the Hearing Impaired. IEEE Press, New York (1980)
2. Moore, B.C.J.: An Introduction to the Psychology of Hearing. Academic, London, UK, pp 66–107 (1997).
3. Gelfand, S.A.: Hearing: An Introduction to Psychological and Physiological Acoustics. 3rd ed., Marcel Dekker, New York, pp. 314–318 (1998)
4. Dillon, H.: Hearing Aids. Thieme Medical, New York (2001)
5. Sandlin, R.E.: Textbook of Hearing Aid Amplification. Singular, San Diego, Cal., pp. 210–220 (2000)
6. Byrne, D., Tonnison, W.: Selecting the gain of hearing aids for persons with sensorineural hearing impairments. Scand. Audiol. 5, 51–59 (1976)
7. McCandless, G.A., Lyregaard, P.E.: Prescription of gain/output (POGO) for hearing aids. Hear Instrum. 34, 1, 16–21 (1983)
8. Braida, L.D., Durlach, N.I., Lippmann, R.P., Hicks, B.L., Rabinowitz, W.M., Reed, C.M.: Hearing aids – A review of past research on linear amplification, amplitude compression, and frequency lowering.

Hardy, J.C. (ed.) ASHA Monographs, Rockville, MA (1979) www.asha.org/uploadedFiles/publications/archive/Monographs19.pdf
9. Lippmann, R.P., Braida, L.D., Durlach, N.I.: Study of multichannel amplitude compression and linear amplification for persons with sensorineural hearing loss. J. Acoust. Soc. Am. 69, 524–534 (1981)
10. Asano, F., Suzuki, Y., Sone, T., Kakehata, S., Satake, M., Ohyama, K., Kobayashi, T., Takasaka, T.: A digital hearing aid that compensates for sensorineural impaired listeners. In: Proc. IEEE Int. Conf. Acoust. Speech Signal Process. 1991 (ICASSP 1991), Toronto, Canada, pp. 3625–3628
11. Stone, M.A., Moore, B.C.J., Alcántara, J.I., Glasberg, B.R.: Comparison of different forms of compression using wearable digital hearing aids. J. Acoust. Soc. Am. 106, 3603–3619 (1999)
12. Tiwari, N., Pandey, P.C.: 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.
13. Pandey, P.C., Tiwari, N.: Dynamic range compression with low distortion for use in hearing aids and audio systems, International PCT Application number PCT/IN2015/000049 (2015)
14. Tiwari, N., Pandey, P.C., Sharma A.: A smartphone app-based digital hearing aid with sliding-band dynamic range compression. In: Proc. 22nd Nat. Conf. Commun. (NCC 2016), Guwahati, India (2016) doi: 10.1109/NCC.2016.7561146
15. Mimi Hearing Technologies: Mimi hearing test v3.0. itunes.apple.com/in/app/mimi-hearing-test/id932496645
16. IT4You: Petralex hearing aid v1.4.3. play.google.com/store/apps/details?id=com.it4you.petralex, http://itunes.apple.com/us/app/petralex-hearing-id
17. Clark, N.: Bio aid v1.0.2. itunes.apple.com/gb/app/bioaid
18. Lamberg Solutions: Hearing aid with replay v2.0.0. play.google.com/store/apps/details?id=com.ls.sound amplifier
19. Newbrick, S.A.: uSound (hearing assistant) v.1.6. play.google.com/store/apps/details?id=com.newbrick.usound
20. Quadio Devices.: Q+ hearing aid. play.google.com/store/apps/details?id=com.quadiodevices.qplus
21. Griffin, D.W., Lim, J.S.: Signal estimation from modified short-time Fourier transform. IEEE Trans. Acoust. Speech Signal Process. 32, 236 – 243 (1984)
22. International Telecommunication Union: Relative timing of sound and vision for broadcasting, ITU Rec. ITU-R BT.1359 (1998)
23. Superpowered Inc: Android audio latency and iOS audio latency test app superpowered.com/latency
24. Loizou, P.C.: Speech Enhancement: Theory and Practice, CRC, New York (2007)
25. Tiwari, N., Pandey, P.C.: 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
26. Jayan, A.R., Pandey, P.C.: Automated modification of consonant-vowel ratio of stops for improving speech intelligibility, Int. J. Speech Technol., 18, 113-130 (2015)
27. Fritsch, F.N., Carlson, R.E.: Monotone piecewise cubic interpolation, SIAM J. Numer. Anal., 17, 2, 238–246 (1980)
28. Khronos group: OpenSLES. www.khronos.org/opensles/
29. Borgerging, M.: Kiss FFT. sourceforge.net/projects/kissfft
30. Sharma, S.: Dynamic range compression results from smartphone app implementation (2016) www.ee.iitb.ac.in/~spilab/ material/saketh/ihci2016