iSound: A Smartphone Based Intelligent Sound Fusion System for the Hearing Impaired

Kathryn Grebel[1], Duy Dang[1], Liran Ma[1], Donnell Payne[1], and Brent Cooper[2]

[1] Department of Computer Science, Texas Christian University, Fort Worth, TX, USA
[2] Department of Psychology, Texas Christian University, Fort Worth, TX, USA

Abstract. High quality hearing aids can come with a premium price tag. As a result, it can be challenging for a hearing impaired individual to obtain a hearing aid. Without such a device, the individual may struggle to communicate with others. To address this challenge, we aim to develop a smartphone-based hearing aid application (termed as iSound) that can serve as an alternative option for hearing impaired individuals. The iSound app is designed to have the capability to adapt to various scenarios with different requirements. Specifically, iSound can utilize both gain (volume) increases and frequency compression techniques as required. We implement iSound on an off-the-shelf Android-based smartphone. The current implementation of iSound is demonstrated to be able to compress signals to a desired range of frequencies as specified, with some distortions and delays. We believe iSound can be a viable alternative for hearing impaired individuals.

1 Introduction

Successful communication depends on all participating parties having the ability to hear as well as speak. This is not always the case because approximately 8.6% of the US population is classified as hearing impaired and in need of a hearing assistance device [1]. In addition, our auditory systems begin to deteriorate as we age, making it difficult to hear sounds at higher frequencies. Hearing aids are traditionally adopted to reduce the impact of hearing impairments.

Recently, there have been multiple techniques proposed to increase hearing aid functionality. Examples of these techniques include frequency compression, amplification, and the ability to adapt to different environments. Research shows that hearing aids utilizing frequency compression techniques generally increase the perception of speech better than the devices that do not [2]. The implementation of these techniques can produce a higher quality hearing aid.

However, these higher quality hearing aids can be cost prohibitive. As a result, many hearing impaired individuals might still struggle to obtain an effective hearing aid. The lack of affordable options motivates us to develop an alternative solution of comparable quality. Smartphones have become ubiquitous and a vital part of people's daily lives. Today's smartphones have the capabilities

to process and transform sound signals. We aim to develop an app (termed as iSound) that utilizes these capabilities to function as a hearing aid. The benefit of this approach is that a smartphone app can be easily acquired and is relatively inexpensive.

This paper discusses the design and implementation of iSound on an off-the-shelf Android-based smartphone. With the individual users needs in mind, we develop a smartphone app that determines the bandwidth of frequencies where his/her hearing capabilities are maximized. During conversations, audio signals are received and modulated to within the aforementioned bandwidth and amplified when necessary. This could afford hearing impaired individuals the ability to fully partake in communication.

The remainder of the paper is organized as follows. We start by discussing the related work in Section 2. Next, we review our design in Section 3. We then explain our current system implementation in Section 4. Following that, we report our evaluation results in Section 5 and finally, we consider future improvements to the design and conclude the paper in Section 6.

2 Related Work

The related information necessary to develop and implement the hearing aid app consists of three basic parts: frequency compression techniques, digital signals, and pitch shifting. Frequency compression refers to reducing all of the frequencies in a bandwidth (i.e. range of frequencies) into a compressed, target bandwidth. A digital signal is a representation of a continuous sound wave consisting of discrete sample points. Pitch shifting refers to manipulating the signal to adjust its pitch for the purpose of frequency compression.

2.1 Frequency Compression Techniques

Modulating the frequencies requires that they are reduced so that all are within the desired range. This is accomplished by using either frequency transposition, linear frequency compression, or nonlinear frequency compression. Frequency transposition reduces every frequency value outside of the user's range by the same quantity in Hertz (Hz). When a frequency is reduced using transposition it can overlap a value already present in the target range. This overlap can cause distortions.

Linear frequency compression reduces every frequency value by a set ratio. We call this ratio the frequency scaling ratio, r, and calculate it as follows:

$$r = \frac{c}{4000Hz}. \tag{1}$$

The variable c is the cutoff frequency for the user's bandwidth of hearing capability (i.e., the maximum frequency that the user can hear with relative clarity), and $4000Hz$ is the maximum frequency of the Human Voice Frequency Range

(HVFR). Thus, for every input frequency value, f_0, the output frequency value, f_1, is given by:

$$f_1 = rf_0. \tag{2}$$

Nonlinear frequency compression is similar to linear frequency compression in that they both utilize a frequency scaling ratio. However, nonlinear compression is predicated on the notion that hearing loss tends to be more prevalent in higher frequency bandwidths and less likely to effect lower frequency bandwidths. Certain bandwidths do not require the same amount of compression as do others, thus the same frequency scaling ratio should not be applied to every bandwidth. The HVFR is broken into the different sub-bandwidths based on the frequency regions found in an audiogram, and a scaling ratio is calculated for each. The ratio for the highest sub-band is calculated in the same manner as the ratio used for linear frequency compression. The remaining ratios are derived using a combination of the initial ratio value and the level of hearing loss for each sub-band.

2.2 Digital Signals

A digital signal is a sequence of discrete values/sample points representative of a sound wave. The number of samples used to represent one second of a sound wave is determined by the sampling rate. For example, using a sampling rate of $44,100Hz$ means that $44,100$ sample points are used to represent one second of sound. When the sampling rate is $22,050Hz$ the same number of samples represent two seconds of sound. Thus, given a sampling rate, a signals duration is measured by the number of its representative sample points.

2.3 Pitch Shifting

François discusses a pitch shifting method in [3]. Pitch shifting is the process of adjusting the frequency of a signal without affecting its duration. The method detailed in the article is mainly focused on pitch shifting for electric guitar signals which behave differently than vocal signals. The method utilizes Fourier transforms and inverse Fourier transforms which can be difficult to code and computationally inefficient. Nevertheless, the algorithm introduces the idea for effectively pitch shifting an audio signal.

3 Design

Our primary objective is to compress the frequency bandwidth of the input audio signals to within a target bandwidth. In addition, we want to maintain an acceptable latency period between input and output audio signals of less than $50ms$ and incur as few distortions as possible created by the frequency compression process. Because iSound is primarily for communication purposes, our main concern is with speech signals. We develop our design for the Human Voice Frequency Range (HVFR) $85Hz - 4000Hz$, i.e., the maximum frequency that we are concerned with is $4000Hz$.

3.1 Frequency Compression Parameter

We want to incorporate the user's specific hearing capabilities in order to adapt the app to fit his/her individual requirements. We utilize techniques similar to those that audiologists use to fit a patient's hearing aid. A person's hearing is evaluated with an audiogram (i.e., a logarithmic plot of frequency (Hz) vs hearing threshold (dB)). A sample audiogram is depicted in Fig. 1. The hearing threshold is the minimum sound level, measured in dB, required to hear a given frequency. Depending on the hearing threshold, a person's hearing ability for that frequency region is classified as normal hearing, mild loss, moderate loss, moderately-severe loss, severe loss, or profound loss. The app uses the information in the audiogram to establish a target bandwidth of frequencies corresponding to the user's hearing capabilities. This is accomplished by choosing the frequency regions that are classified as normal hearing or minimal loss.

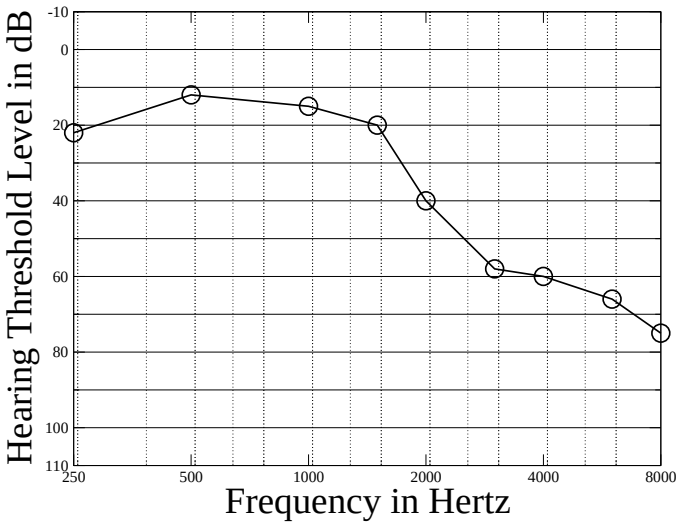


Fig. 1. Sample Audiogram

3.2 Frequency Compression Scheme

We utilize a frequency compression technique to condense all of the frequencies in the HVFR to within the target bandwidth previously discussed. In particular, linear frequency compression is employed in our design because of the relative ease in implementation and the desire to maintain acceptable latency values. A signal's frequency is inversely related to the length of its period, T, i.e., $f = \frac{1}{T}$. Thus, a simple way to reduce the frequency of the input signal is to increase the length of its period by an amount proportional to the frequency scaling ratio

discussed in Section 2.1. Unfortunately, this results in an output signal that has a longer duration than the input signal, which can cause the audio to be visually out of sync.

Thus, the linear frequency compression is carried out using a process known as pitch shifting [3]. Pitch shifting is based on the concept of increasing the length of a signal's period as mentioned above. However, steps are taken to preserve the duration of the signal. Since we are working with digital signals, preserving the duration means preserving the number of representative sample points as discussed in Section 2.2.

We know that increasing the length of a signal's period reduces its frequency, but also causes an increase in its duration. In order to preserve the signal's duration, an intermediary signal is created so that it has a duration equal to the product of the frequency scaling ratio and the duration of the input signal. The length of the period of the intermediary signal can then be increased by an amount proportional to 1 minus the scaling ratio (i.e., $1 - r$). This produces an output signal equal in duration to the input signal that also has a reduced frequency.

The pitch shifting process involves two principal steps: time scaling and up-sampling. A major benefit of utilizing pitch shifting in the design is that it is easily customizable. That is, there is more than one way for implementation. This provides us with the ability to choose an implementation that best meets our stated goals.

Time Scaling The intermediary signal is created by utilizing a time scaling technique (i.e., the method of adjusting the duration, or number of sample points, of a signal without affecting its frequency). In general, the number of sample points in (i.e., the length of) the intermediary signal, l_1, is given by multiplying the length of the input signal, l_0, by the scaling ratio r:

$$l_1 = rl_0. \tag{3}$$

The intermediary signal can be created with one of the following three time scaling techniques:

- The first technique is to directly collect the first l_1 sample points of the input signal as the intermediary signal. The rest of the input signal is discarded. This technique requires the least computation among the three techniques;
- The second technique is Pitch Synchronous Overlap and Add (PSOLA) where a signal of a single frequency is divided into a sequence of overlapping, periodic segments. Each segment is one period in length and is partially shared with its neighboring segments. Because the segments overlap each other by the exact same amount, adding or removing a segment does not affect the frequency of the overall signal. The duration of a signal is decreased by removing one or more of these overlapping segments. Since PSOLA is a time-domain-based approach, the computations are relatively simple and efficient [4];

– iii) The third technique is phase vocoder. It is a frequency-domain-based technique that utilizes an overlap-add approach like PSOLA. Instead of using pitch synchronization, this technique modifies the signal's phases in its short-time Fourier Transform to achieve synchronization between the overlapping segments. The phase vocoder can compute more complex signals than PSOLA, but it is not as computationally efficient as PSOLA [5].

Up-sampling The signal is then up-sampled (i.e., the process of inserting additional sample points to increase the duration of the signal and lower its frequency) to produce the output signal. Up-sampling can be accomplished using one of the following three techniques:

– Point insertion by averaging is the process of inserting a new point every m points in the intermediary signal, where m is calculated as follows

$$m = \frac{r}{1-r}. \tag{4}$$

The value of the inserted point is calculated by averaging the two points adjacent to it.

– Cubic spline interpolation derives a piecewise-defined function, consisting of 3^{rd} degree polynomials, where each piecewise function corresponds to a pair of points, e.g.,

$$f(x) = \begin{cases} g_1(x) & 0 \le x \le 1 \\ g_2(x) & 1 \le x \le 2 \\ \vdots & \\ g_n(x) & n-1 \le x \le n \end{cases}. \tag{5}$$

– Lagrange interpolation derives an n^{th} degree polynomial using the formula

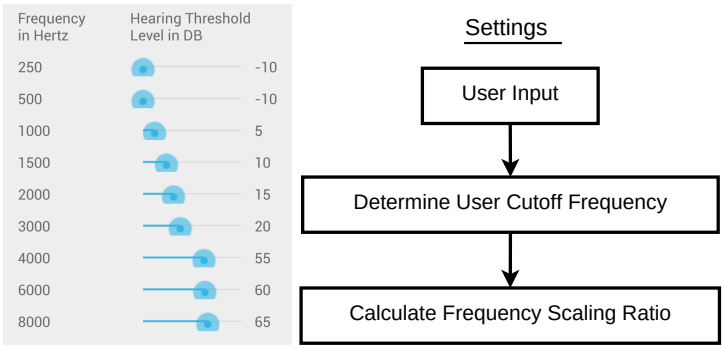
$$P_n(x) = \sum_{j=0}^{n} y_j \prod_{k \neq j} \frac{(x - x_k)}{(x_j - x_k)}. \tag{6}$$

In both cubic spline and Lagrange interpolation, the output signal is created by using the derived formulas to calculate the desired number of evenly space representative sample points.

4 Implementation

The current implementation is based on the Android 4.4.2 operating system (Kit Kat). To be specific, our implementation of iSound is on the Samsung Galaxy S5 that has a $2.5GHz$ Quad-Core Processor and $2GB$ of RAMs. The sampling rate is set at $44.1kHz$, which is equivalent to CD quality audio. The bluetooth headset utilized for the app is the Logitech Wireless Headset H800. The app's implementation can be broken down into two main parts: front end and back end. The front end (i.e., user interface) is where the user enters his/her specific hearing requirements. The back end is where the actual signal processing occurs.

4.1 Front End



(a) User input. (b) User settings.

Fig. 2. Front end.

When the app is launched for the first time, the user is prompted to enter his/her specific hearing prescription given by the audiologist. Specifically, the app's interface has eight separate input fields as shown in Fig. 2(a), one for every frequency level found on an audiogram. Each level has a corresponding SeekBar for the user to specify their hearing threshold in decibels (dB). The possible settings for the SeekBar range from -10 to 110, mirroring an audiogram as shown in Fig. 1. The app then uses that prescription information to calculate the user's settings needed for future signal processing in the back end as shown in Fig. 2(b). Specifically, the user's cutoff frequency (i.e., the maximum frequency of the user's hearing capability) is determined by evaluating each frequency level's class of hearing loss. The lowest frequency level classified as minimal hearing loss is designated as the user's cutoff frequency. For example, in Fig. 2(a) the user's cutoff frequency is $3,000Hz$. The frequency scaling ratio is calculated using Eq. 1. Specifically, the scaling ration for Fig. 2(a) is 0.75.

4.2 Back End

When the user runs the app, the back end starts processing the incoming audio signals as shown in Fig. 3. Sound is picked up by the phone's microphone and converted to a digital signal consisting of discrete sample points. These digital sample points are sent to the app. The app then creates a buffered input signal by creating an array containing a finite number of sample points. The length of the array is determined by the minimum number of sample points that the Android OS requires for a chosen sampling rate. For example, this implementation of the app buffers input signals as arrays containing $1,024$ sample points because it is the required minimum for the sampling rate $44.1kHz$.

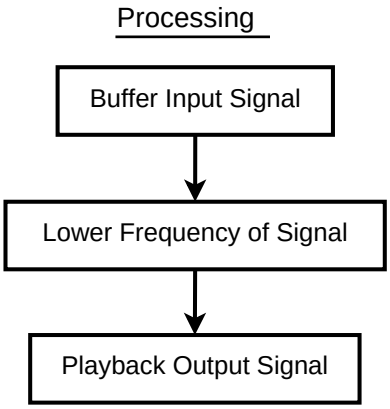


Fig. 3. Back end.

Next, the frequency of the signal is lowered by creating an intermediary signal and up-sampling it. Specifically, the intermediary signal is obtained by creating an array consisting of the first l_1 points from the input signal, where l_1 is calculated using Eq. 3. This method of time scaling is utilized because it is the simplest to implement and most computationally efficient of the three methods discussed. Up-sampling the intermediary signal creates the output signal. The up-sampling is accomplished using either the point insertion by averaging technique, cubic spline interpolation, or Lagrange interpolation as detailed in Section 3.2.

Finally, the Android AudioTrack class buffers the output signal in preparation for playback. The streaming mode of the class pushes each element of the buffered output signal one-by-one to a queue in the phone's AudioSink hardware. The hardware then plays the processed sound for the user.

5 Evaluation

We assess the application's performance with the following criteria: i) input frequencies are shifted to within the designated range; ii) latency periods between input and output audio signals are less than $50ms$; iii) distortions created by the compression process are minimized or eliminated entirely. The simulation environment is setup such that the user's maximum cutoff frequency is $c = 3000Hz$ and the frequency scaling ratio is $r = \frac{3000Hz}{4000Hz} = 0.75$.

We test the application's performance in regards to the first criterion using a Samsung Galaxy S5, Echo Capture (an Android app that plays and graphs sound) [6], an online tone generator [7], and an online tuner [8]. The tone generator produces various, specified frequencies within the HVFR. The phone's microphone is held near the computer's speaker for the purpose of capturing the generated frequency. After processing the input signal, the app stores the output signal in a public file on the phone. Echo Capture then plays the contents of the

public file as the input for the online tuner, which gives the frequency of the output signal. We then compute the actual frequency scaling ratio by dividing the output frequency by the input frequency (note: the ideal scaling ratio is given above as r), i.e.,

$$r_A = \frac{f_1}{f_0}. \tag{7}$$

This process is repeated until we obtain 30 pairs of ideal and actual frequency scaling ratios. We then use the mean absolute error method to evaluate the app's performance in producing the expected output frequency. The test data collected is displayed in Table 5. The calculated mean absolute error is 0.05. We also found that the actual ratios were very consistent; in 28 of the 30 cases the actual ratio is 0.80, i.e., 0.05 greater than the ideal ratio of 0.75. Further evaluation is necessary to determine the reason for the discrepancies in the output frequencies as we are currently unsure of the cause.

Table 1. Frequency Shifting Test Data

Input Frequency (Hz)	Ideal Ratio (r_I)	Detected Frequency (Hz)	Actual Ratio (r_A)	Absolute Difference ($\lvert r_A - r_I \rvert$)
1000	0.75	805.47	0.81	0.06
1500	0.75	1205.94	0.80	0.05
2000	0.75	1597.52	0.80	0.05
2500	0.75	1994.69	0.80	0.05
3000	0.75	2402.92	0.80	0.05
3500	0.75	2799.64	0.80	0.05
4000	0.75	3196.05	0.80	0.05
4500	0.75	3604.25	0.80	0.05
5000	0.75	4000.68	0.80	0.05
5500	0.75	4397.31	0.80	0.05
6000	0.75	4805.35	0.80	0.05
6500	0.75	5202.18	0.80	0.05
7000	0.75	5598.88	0.80	0.05
7500	0.75	5995.12	0.80	0.05
8000	0.75	6403.51	0.80	0.05
			SUM	0.76
			MEAN	0.05

Furthermore, people with no known hearing impairments tested the app during a face-to-face conversation. All agree that *i*) there are distortions present in the output signal; *ii*) the latency period between the input and output signal is noticeable and almost certainly longer than $50ms$. The majority of the distortions in the output signal are most likely a result of the time scaling method utilized. Because a portion of the tail end of each input signal is discarded, consecutive output signals do not match up perfectly. That is, the consecutive

signals are no longer smoothly connected. The result is the periodic clicking sound heard in the output. The too large latency values are a result of audio processing issues that are inherent to the Android operating system.

We also observed that cubic spline interpolation can take up to 20 minutes and Lagrange interpolation can take at least 10 minutes to accomplish the up-sampling of an intermediary signal that is less than $23ms$ in length. This currently makes the point insertion by averaging technique the better choice for up-sampling the intermediary signal.

6 Conclusion

In this paper, we discuss the design and implementation of a smartphone-based hearing aid app. Specifically, our app applies user input to tailor its parameters by calculating an individualized target bandwidth and utilizing frequency compression techniques to increase speech intelligibility. Input audio is shifted to within the target bandwidth using linear frequency compression. The scheme is implemented as an app on an off-the-shelf Android-based smartphone. Our preliminary results show that the app does lower the frequencies of input signals, but there are distortions in the output audio and the latency exceeds the desired limit, which might require moving development to another platform entirely. We intend to improve this scheme by exploring the means for further reducing latency periods and distortions. Additionally, we plan to devise a method to introduce scenario adaptability to the app.

References

[1] Holt, J.A., Hotto, S., Cole, K.: Demographic Aspects of Hearing Impairment: Questions and Answers. Center for Assessment & Demographic Studies, Gallaudet University (1994)
[2] Simpson, A., Hersbach, A.A., McDermott, H.J.: Improvements in speech perception with an experimental nonlinear frequency compression hearing device. International Journal of Audiology **44**(5) (2005) 281–292
[3] François, G.: Guitar pitch shifter. Matlab Code and Mathematic Equations http://www.guitarpitchshifter.com/matlab.html (2012)
[4] Charpentier, F., Stella, M.: Diphone synthesis using an overlap-add technique for speech waveforms concatenation. In: Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86. Volume 11., IEEE (1986) 2015–2018
[5] Laroche, J., Dolson, M.: Improved phase vocoder time-scale modification of audio. Speech and Audio Processing, IEEE Transactions on **7**(3) (1999) 323–332
[6] Capurso: Echo Capture
[7] Tone Generator: Online Tone Generator. `http://www.onlinetonegenerator.com`
[8] Seventh String Software: The Seventh String Tuner. `http://www.seventhstring.com/tuner/tuner.html`