Hear Through App for “Using Adaptive Filters and Combining Different Sound Sources to Make Hearing in Different Circumstances and More ‘Real’ Experience”

Mankeerat S. Sidhu, Ryan M. Corey

Department of Electrical and Computer Engineering , Grainger College of Engineering , University of Illinois at Urbana-Champaign

INTRODUCTION

Many people suffer from hearing loss and listening in a crowded environment in general raises many difficulties. Although most people with hearing loss carry hearing aids, they are often expensive and require regular hearing care. Since most people already have a smartphone equipped with a mic and connected with a headphone, it might be beneficial for general purpose development to approach low level hear-through functionality through them. Although the acoustics of the microphones in the smartphones are worse off than the acoustics of hearing aids, they can be adapted in a way to provide general functionality.

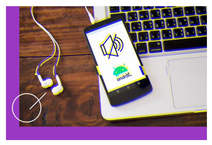
The device thus focusses on an Android app that allows the user to record the smartphone’s microphone input, amplify and play it back in real time over the user’s headphones. Despite the limitations imposed by the hardware functionalities of the device and by the android platform, the app made through AndroidStudio aims to operate in general purpose setting and later be incorporated into the multi-person, multi-speaker device that uses adaptive filters and combines different sound sources to make hearing in different circumstances a more ‘Real’ experience.

android studio

AIM

The purpose for the whole app is to do one simple task – record the microphone sound, amplify it and then play it back to the headphones. The audio should also be streamed in Realtime (< 10 ms of delay) and the app should also have some equalization technique to match the person’s hearing loss or keeping track of direction.

For hearing aids, most of the relevant sounds are in a frequency range between 500 and 6000 Hz. As the Atherogenic index (ATH index) rises but the threshold of pain doesn’t, it leads to volume compression and if the latency is more than 10 ms, the user is able to notice the delay and a delay more than 50 ms causes a conversation to be impossible. Thus, the app aims to overcome this problem with low level app development despite the poor acoustics and limited optimization for low level development in smartphones.



C++

METHOD

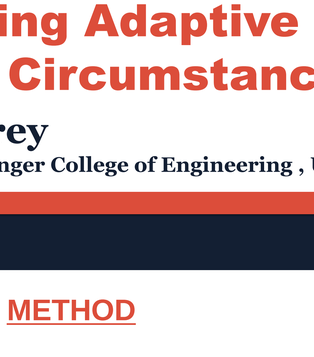
Android App Framework

Since apps on android platform are built using Java, developers do not need to worry about memory management and since Java runs on JVM, it is cross platform, removing the effort of compiling the app for every different CPU architecture.

However, JVM slows down the program due to being JIT compiled and thus making a device that needs low level latency, the mentioned disadvantages need to be bypassed.

The NDK helps with that as it allows execution of Native code (C++) and allows instructions to be executed directly on the CPU instead of the JVM.

The delay in the sound can be categorized by the latency induced by the device’s hardware and the latency induced by the app. To control that, superpowered was used (discussed in Architecture of Backend Code)



Testing the app using an Audio Interface, a head-replica with built in mics and measuring the audio signals by the external sound from the speaker and the sound coming out of the developed app.

1. Superpowered C++ Audio Library and SDK for Android, iOS, macOS, tvOS, Linux and Windows.

Superpowered C++ Audio Library and SDK is the leading C++ Audio Library featuring low-power, real-time latency and cross-platform audio players, audio decoders, Fx (effects), audio I/O, streaming, music analysis and spatialization.

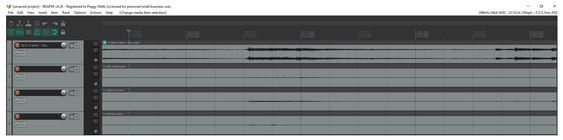
For the most up-to-date feature list, see: https://superpowered.com/audio-library-sdk

```
// In the frequency domain we are working with 1024 magnitudes and phases for every channel (left, right), if the fft size is :
while (frequencyDomain->timeDomainToFrequencyDomain(magnitudeLeft, magnitudeRight, phaseLeft, phaseRight)) {
    // You can work with frequency domain data from this point.
    // This is just a quick example: we remove the magnitude of the first 20 bins, meaning total bass cut between 0-430 Hz.
    // memset(magnitudeLeft, 0, 80);
    // memset(magnitudeRight, 0, 80);
    // We are done working with frequency domain data. Let's go back to the time domain.
    // Check if we have enough room in the fifo buffer for the output. If not, move the existing audio data back to the buffer's beginning.
    if (fifoOutputLastSample + stepSize >=
        fifoCapacity) { // This will be true for every 100th iteration only, so we save precious memory bandwidth.
        int samplesInFifo = fifoOutputLastSample - fifoOutputFirstSample;
        if (samplesInFifo > 0)
            memmove(fifoOutput, fifoOutput + fifoOutputFirstSample * 2, samplesInFifo * sizeof(float) * 2);
        fifoOutputFirstSample = 0;
        fifoOutputLastSample = samplesInFifo;
    };
    // Transforming back to the time domain.
    frequencyDomain->frequencyDomainToTimeDomain(magnitudeLeft, magnitudeRight, phaseLeft,phaseRight,fifoOutput + fifoOutputLastSample * 2);
    frequencyDomain->advance();
    fifoOutputLastSample += stepSize;
};
// If we have enough samples in the fifo output buffer, pass them to the audio output.
if (fifoOutputLastSample - fifoOutputFirstSample >= numberOfSamples) {
    SuperpoweredFloatToShortInt(fifoOutput + fifoOutputFirstSample * 2, output,numberOfSamples);
    fifoOutputFirstSample += numberOfSamples;
    return true;
} else return false;
}
```

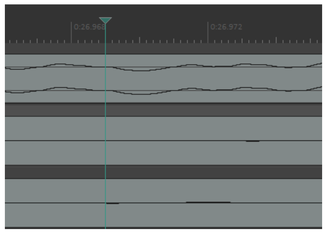
The Code in C++ and using SuperpoweredSDK to develop the Audio Processing component of the Hear-Through App.

RESULTS

Measuring the latency of the hear through app shown in ‘method’, and using Reaper, Audacity and the Antelope Audio Interface, the sound signals from the 4 different microphones (left and right mic outside the muffs(real speaker sound)) and (left and right mic inside the muffs (processed app sound)) were collected. Cross correlating the two using python and the seeing the delay between them, the delay averaged out to be 4 ms, which makes conversation perfectly natural and thus proves that the app is performing well. The difference between the two signals heard on reaper was also measure and it turned out to be around 4 ms as well confirming our results. Thus, the hear through app achieves its purpose and is ready to be implemented with multiple mics and an adaptive filter.



Using Reaper to collect multiple sound signals to be later processed in Audacity and save the sound recordings as WAV files.



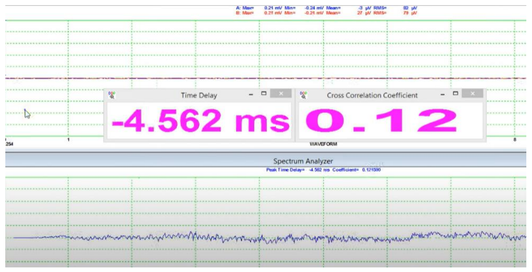
Looking at the difference in time between the first signal in the third row to the first signal in the second row giving 4 ms of delay between the same audio recording.

For continuous functions f and g, the cross-correlation is defined as:[1][2][3]

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} \overline{f(t)}\, g(t+\tau)\, dt \quad \text{(Eq.1)}$$

which is equivalent to

$$(f \star g)(\tau) \triangleq \int_{-\infty}^{\infty} \overline{f(t-\tau)}\, g(t)\, dt$$



Using cross correlation to measure time delay and seeing the latency of the app.

Architecture of Backend Code

Only the latency caused by the software can be controlled, thus all unnecessary software layers must be removed that adds to extra latency. Thus, all the sound processing code is written in C++ to avoid the JVM software layer (shown in methodology). Superpowered SDK was used to simplify the access to audio data by the device and as they claim to be the “Fastest Mobile Audio engine for Interactive audio apps, music etc.”.

Audio latency induced be the app is solely caused by the fact that all operations performed during the processing of the sound take time. Firstly, some operations require the microphone to record several samples of audio before even being able to begin processing (like e. g. the Fourier transformations that are used to apply the equalizer). But other operations (like memory access) also take time. Hence, every operation must be evaluated and if any operation is found to be not necessary, it must be removed. All remaining operations must be designed in a way that optimizes the time it takes to execute them. For example, operations should use pointers to access memory to avoid unnecessary copies of objects.

CONCLUSIONS

Using the results of the app and testing it in real life, it can be concluded that the app is functionable and provides the hear through feature with less than 10 ms of delay and amplifying the sound concurrently. It can thus be used in the broader application of combining multiple sound sources and making the listening experience through microphones more ‘real’ and making it easier to understand people in a crowded environment.

ACKNOWLEDGEMENTS

The app was developed by Mankeerat Sidhu with the aid of Ryan M. Corey (Postdoc) at the Augmented Listening Lab (P.I. – Prof. Andrew Singer) at CSL @ UIUC.

ILLINOIS

ILLINOIS ECE