THE DIGITAL EARS: A BINAURAL SPATIALIZATION PLUG-IN

by

Sydney Martel Sumner

HONORS THESIS

Submitted to Texas State University
in partial fulfillment
of the requirements for
graduation in the Honors College
May 2022

Thesis Supervisor:

Mark Erickson

FAIR USE AND AUTHOR'S PERMISSION STATEMENT

Fair Use

Duplication Permission

.

DEDICATION

I would like to dedicate this thesis to any student that thinks they may have been overzealous in their pursuits, bitten off more than they can chew, or simply feel that they are in over their head. The work I have done now felt impossible when I first began looking into it last fall. The idea took people by surprise, initial proposals were met with dubious looks, initial drafts with very tentative comments. But I have accomplished so much more than I thought I was capable of. Now it was not easy. It took endless videos, articles, books, questions that led to more questions, and a lot of help from Google, but I made it. Now it's your turn. Write the book, score the film, make the video game, do something that scares you. It won't be easy, you'll want to quit, it will stress you out. But the emotion you're left with when you feel finished with your endeavor is something that cannot be replicated by doing anything else. And who's going to do it if you don't?

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Mark Erickson, for sticking with me through this project. You pointed me in the right direction ten times over, explained things to me until they finally clicked (often very late), and learned everything I did so you could help me. I would also like to thank Joey Hook. I was extremely nervous about this project and from our first conversation about it you quelled my concerns, helped me understand what I was getting into, and were there to help me when I felt like I was hitting a wall. Finally, I would like to thank my friends and family for supporting me through this journey, especially while I was completing the brunt of it four states away from you all. I wouldn't have accomplished this without your encouragement, reassurance, and constant questioning, "Are you done yet?" I appreciate each of you very deeply. Thank you.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure **Page**

LIST OF ABBREVIATIONS

Abbreviations

AC-3 – Audio Compression 3rd generation

ADM-BWF – Audio Definition Model Broadcast Wave Format

AES – Audio Engineering Society

API – Applied Programming Interface

DAW – Digital Audio Workstation

DFT – Discrete Fourier Transform

DSP – Digital Signal Processing

DVD-A – DVD Audio

FFT – Fast Fourier Transform

HRIR – Head Related Impulse Response

HRTF – Head Related Transfer Function

ICTD – Inter-Channel Time Difference

IDE – Integrated Development Platform

iFFT – inverse Fast Fourier Transform

ILD – Interaural Level Difference

IR – Impulse Response

ITD – Interaural Time Difference

MLP – Meridian Lossless Packaging

OOP – Object Oriented Programming

LIST OF ABBREVIATIONS CONT.

Abbreviations

SACD – Super Audio CD

SOFA – Spatial Oriented Format for Acoustics

VST – Virtual Studio Technology

UMG – Universal Music Group

ABSTRACT

In the last year, popular music has taken the leap into an "immersive" format again. Streaming services like Apple Music and Tidal are offering "Spatial Audio," which is mainly branded as Dolby Atmos, a surround sound format intended to fully envelope the listener with sound. Spatial technology aims to produce a similar immersive experience on a range of surround sound set ups, and even headphones.

When replicating spatial audio over headphones, the multi-channel output is "mixed down" in real-time to binaural after analyzing a track's accompanying metadata. Binaural audio can also be achieved without any prior three-dimensional processing. In binaural synthesis, a more direct approach, a mono audio clip recorded with a typical microphone can be mathematically combined with a quantification of human hearing to place the sound anywhere around the listener. Until recently, binaural synthesis was mainly applied to audio for video games and virtual reality, but could translate to music for a target audience of headphone users.

To test this theory, a "binaural effect" is needed. This effect would function using Head Related Transfer Function (HRTF) measurements that quantify the natural equalization of audio from the human head and torso in a free field listening space. Processing an input clip with the HRTF for the position of the user's choice, the output would appear to place the source in the new location, giving a sense of spatial audio by manipulating human perception instead of physically extending the listening sphere. The ideal, yes elusive goal of this research was the creation of a "binaural effect" for use

as an audio plug-in. While not yet attained, various aspects of digital signal processing, different ways of achieving the desired output, and coding for a VST plug-in will be investigated in pursuit of implementation. Previous attempts to popularize immersive music, relevant immersive playback systems and mixing techniques, and the methods used to develop and implement the software's algorithms will be surveyed as an accompaniment to the software development.

1. SURROUND SOUND IN POPULAR MUSIC PAST AND PRESENT

What is currently seen on the market, branded as “Spatial Audio”, is music in a three-dimensional format, played back via surround sound or binaural audio. The music most listeners consume is released in a stereo format. Stereo music plays back on two speakers built into a radio, computer or through headphones or earbuds without any complications. Upon playback, the listener can imagine a line in front of them between the two speakers and generally localize instruments along this line. Today’s spatial audio now employs full 360° panning and the element of height to fully surround the listener. Before the advent of technology that could achieve this, immersive music was realized by physically extending the listening sphere using surround sound. There have been two big pushes since the early ‘70’s to create consumer technology capable of playing back multi-channel music in the home. Both endeavors proved ultimately unsuccessful, and only time will tell if the current form will gain more traction.

1.1 QUADRAPHONIC SOUND

The first attempt at surround sound music appeared in the early ‘70’s as quadraphonic systems using four speakers intended for use with LP’s and 8-tracks. These systems were based on the idea that the majority of what is heard in concert halls is the reflections off walls and ceilings, only a small portion of our perception originating from the source’s direct sound.[29] Two additional speakers placed behind the listener could offer synthetic “reflections” to simulate a sense of space. The intention was that consumers could easily upgrade their systems to support quad records by adding the extra speakers and a receiver

that would not disrupt the original compatibility with stereo formatted records. This was achieved in two ways.

The first was a matrix technology that encoded the four quad channels into the existing two-channel format of vinyl discs. A decoder then split the two channels back into four during playback, but not without issue. A phenomenon called “cross-talk” occurred frequently in the separation process resulting in information intended to play from the rear speakers leaking into the front speakers, compromising the separation that enhanced the listening experience in the first place. Despite these issues, matrix formatting was still extremely convenient for production because it did not require any costly changes in the manufacturing of records, and it could be played over the radio. Both the CBS corporation and later Sansui developed matrix-based discs called SQ (or QS) that were heavily supported by Columbia Records.[29]

In parallel, discrete based quadraphonic systems were also developed. These differed from matrix systems in that the four channels remained separate on discs and a demodulator was used to disperse them to their respective speakers. This system maintained excellent separation in channel playback at the cost of expensive production. Special styluses and cartridges had to be used on a more durable vinyl after finding the extra channels would wear away on typical vinyl. The resulting technology was the CD-4 format produced by JVC and supported by RCA. While also downward compatible like the matrix systems, CD-4 and SQ were not compatible with each other, so the release of CD-4 in ’72 was highly criticized for its division of the market, and prevention of SQ

from becoming a standard. Additionally, the release of CD-4 appeared premature. Demodulators weren't available on the market until mid-1973 and only 5 records had been released for the format while SQ had 71 records available and the support of 51 decoder manufacturers.[29]

Consumers now feared they would invest in the "wrong" system. If one were to win out over the other, they would be left with a system no one is producing discs for. This new hesitation resulted in a halt in anticipated sales growth, neither system reaching their projected revenues in '74 and '75.[29] Other contributing factors to quad's diminishing success included confusion on how the system worked and aversion to feeling like one was "in the band" as a listener.[8] Due to these issues, quadraphonic music and systems were discontinued by '76 in favor of familiar stereo, leaving behind a considerable amount of unsellable technology.[29]

1.2 DIGITAL SURROUND SOUND

After quad's failure to make immersive music the standard, we did not see a subsequent attempt for another 20 years. In 1992, Dolby released its new AC-3 file format which could replicate 6-channel audio for films, commonly used for a 5.1 format (left, center, right, left surround, right surround, and low frequency channel). Although some late VHS's carried surround sound, it became much more standard with the advent of DVD in 1997.[20]

With the success of AC-3 on DVD, and more consumers adding 5.1 surround systems to their home theatres, this opened a window for music to jump back into

surround as well. Recording industry associations from the U.S., Japan, and Europe came together in 1998 to form the International Steering Committee with the goal of making the CD more like the DVD with abilities to house high quality and multi-channel audio. Despite its popularity, CD audio was limited to two channels and would require new technology to allow for enough storage to contain four more. The committee expected developers to produce a disc with surround sound that was downwards compatible with typical CDs. Sony and Phillip's solution was direct stream transfer or DTS, a lossless compression format on a dual-layered disc named SACD (Super Audio CD).[20] These discs required an SACD player to be heard as intended, but both CDs and SACDs would play back in either player. This worked so that consumers could experience albums released or re-mixed for 5.1 in a dedicated listening space, but still get a stereo playback in their car or boomboxes.[21]

Like the "war" we saw with formatting in the age of quadraphonics, again not everyone was happy to let SACD become a standard for surround sound music. The DVD Forum, led by Toshiba, refused to pay Sony and Phillips for rights to the patent, instead developing DVD-Audio (DVD-A) in 2001.[21] It employed Meridian Lossless Packaging or MLP to achieve the same 74 minutes of high quality, multi-channel audio on a disc that required its own player that was incompatible with CDs.[20] Understandably, many consumers were not willing to invest in a system that made the rest of their collection useless and with record labels again choosing sides, neither format really caught on.[22] SACD and DVD-A's combined sales during their time on the market were

comparable to LP sales or 1/700th of CD sales in 2005.[8] The split technologies, lack of standards for channels, and shift from packaged media to digital downloads after the release of the iPod spelled the end of multi-channel music for the second time.[22]

1.3 THREE-DIMENSIONAL MUSIC

With the continued success of 5.1 (later 7.1) surround sound for film, sound corporations continued to innovate, wanting to further immerse the listener. Technologies like Auro 3D, Dolby Atmos, and DTS:X began to appear in the early 2010s. Both Dolby Atmos and Auro 3D saw theatre debuts in 2012 with the movies Brave and Red Tails, while DTS:X focused on home theatre, first seen on a disc release of Ex Machina in July 2015.[28] While all taking different approaches, they aimed to add the element of height to the soundscape, using techniques like layered speakers and even ceiling mounted speakers to accomplish this. Instead of a two-dimensional circular image, they provided a three-dimensional sphere-like bubble of sound. However, these systems did not just add extra channels for static speaker locations. Instead, they restructured the way surround sound is conceptualized, adding object-based panning on top of a more familiar channel-based system. The object concept allowed an individual track to move, or pan around, the listening space as its own entity, instead of the traditional practice of confinement to one speaker. Due to the systems' starts in film, the programming also allowed for huge numbers of objects in each finished track.

Object-based systems also touted flexibility on playback. Object's positions are recorded and saved using metadata, allowing the finished files to translate to any

surround system with any number of speakers, of course with varying success. The ideal playback set-up in the home looked different for each approach. Auro 3D, the least flexible of the systems, relied on a layered set-up with a 5.1 base, four elevated speakers, and a ceiling speaker, resulting in something we might call 9.1.1. DTS:X and Dolby Atmos were more similar, calling for a 7.1 base with two or four ceiling-mounted speakers. Any of these are fairly extreme for the average home, so Dolby and DTS worked with manufacturers to create speakers and soundbars with directional and upward firing horns to bounce sound off walls and ceilings to simulate extra speakers, helping to solidify the immersive effect without the need to have an extremely dedicated viewing space. While maybe not providing a "movie-theatre quality" experience, the special speakers made the format much more accessible.[7] In addition, using a "binauralization" technique to translate the metadata to a binaural format, object-based audio could also be heard over headphones.

At this point, object-based technology was almost entirely reserved for film applications, but with key partnerships with the Universal Music Group (UMG) and Avid, plans were forming to introduce Dolby Atmos for mixing music.[13] Dolby had been working with Avid to continue developing their "Atmos Production Suite," which began as a hardware and software combination, smoothing out the kinks to reduce it to one software product by early 2017. This allowed for complete "in-the-box" workflows, leaving a monitoring system as the largest investment for an engineer.[33] With an established workflow, UMG ensured that there was plenty of material to test as spatial

music. For its 50th anniversary, Sgt. Pepper's Lonely Hearts Club Band was remixed and released as the first album formatted for Dolby Atmos that summer.[13] In the next two years, over 2000 songs had been remixed for Atmos before another competitor joined the market.[9] Sony released their own object-based system, 360 Reality Audio, in 2019. While Sony was a few years behind, they began releasing music in the format and both formats were picked up by different streaming services like Amazon Music, Deezer, Tidal, and Apple Music in 2021.[4]

Streaming platforms now offered the music, but the question of how users would receive it persisted. With the audio only widely available for about a year, it's hard to say whether the concept has fully caught on. Artists have continued to release new music spatially, attesting that the format hasn't gone ignored. Popular artists like Billie Eilish, Justin Bieber, Lil Nas X and more have released all their new music in a spatial format, some even going so far to re-mix their entire catalogue, proving the concept has fully taken hold on the production side. Consumers on the other hand, may need an adjustment to fully appreciate the immersive quality of the music. It has been argued that most music is now consumed as a background to other activities or viewed as a part of an environment, not as its own entity. Music consumption has shifted from very active listening to accompaniment, with vast libraries of curated playlists for any activity or mood.[20] Spatial music disrupts this, demanding more attention than a typical stereo mix due to its nature of immersion. Ultimately, consumer reception will likely determine whether or not spatial audio will finally popularize music in an immersive format.

2. PLAYBACK FORMATS AND PROCESSING TECHNIQUES

2.1 STEREO PLAYBACK

In popular music, stereophonic recordings have been the standard for the last 80 years. No matter the medium, whether it be vinyl, CD, mp3, or m4a, your favorite songs are all two track masters meant for playback on two speakers. These two tracks manipulate binaural cues to place sources, like individual instruments, along a straight line spanning the distance between two speakers in front of the listener.[9] These binaural cues include interaural time differences (ITDs) and interaural level differences (ILDs), the main signals used in human sound localization. ITDs measure the time between a signal reaching one ear before hitting the other, giving clues as to where the sound source is located. ILDs are similar, describing the difference in volume of a signal between both ears. Stereo programs use inter-channel time differences (ICTDs) and/or inter-channel level differences (ICLDs) to emulate these natural phenomena to create a stereo image.[9]

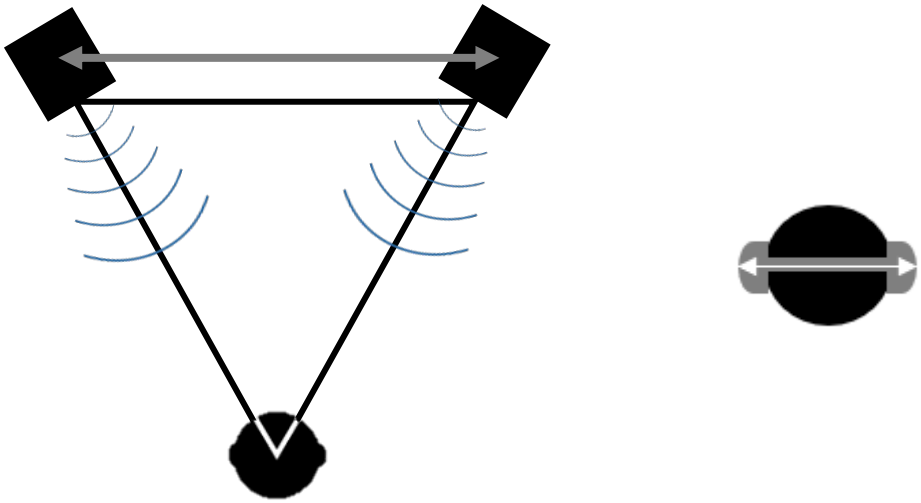


Fig. 1 Stereo "Sweet-Spot" and Phantom Plane

This is accomplished by feeding the same mono signal to both speakers at various levels and delays to produce phantom signals in the center of the speakers.
In headphones, this stereo image is much less clear because the listener is no longer in the "sweet spot", the intended listening position. [9] The horizontal plane now appears to go through or very near to the head and the transition from ICLD to ILD does not occur (Fig. 1).

These types of recordings, as quality as they may be, typically do not provide any immersion for the listener. The listener is aware that the source of the music are the speakers, and nothing is done to reduce that perception. Spatial audio aims to change that, its goal being that only the virtual source is perceived by the listener. This can be attempted in several ways.

2.2 SPATIAL AUDIO PLAYBACK

There are only two real ways to reproduce spatial audio for listening. They can be divided by the mode of delivery, either through loudspeakers or headphones. The first, surround sound, the second, binaural. Each has their strengths and weaknesses in creating an immersive sound field for the listener.

2.2.1 SURROUND SOUND

Surround sound systems are capable of processing more than two audio channels for an immersive experience. A receiver is used to decode the multi-channel audio, sending each channel to its respective speaker around the listener. There are a few standard monitoring set-ups for surround sound that do not change between professional

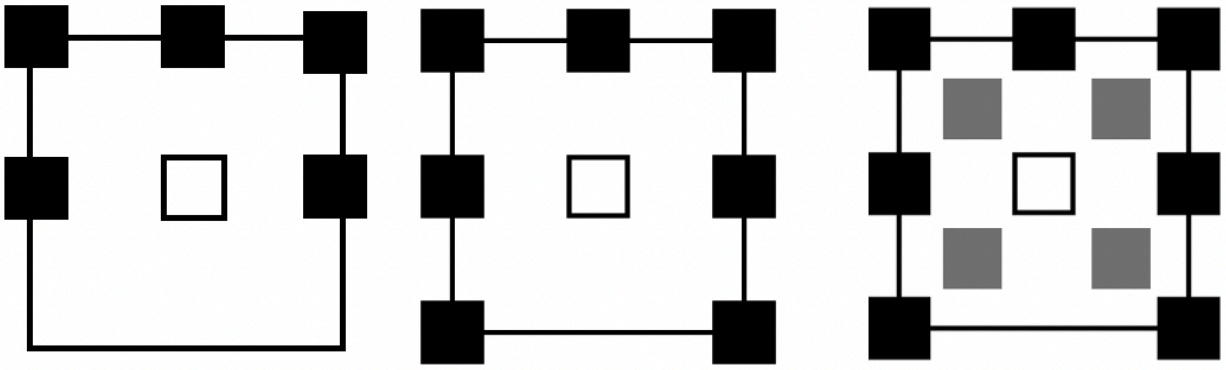


Fig. 2 Surround Setups: Three common surround setups are displayed. Black indicates an ear level speaker, gray indicates a ceiling mounted speaker, white indicates the listener. Left to right: 5.1, 7.1, and 7.1.4, otherwise known as Atmos.

and consumer listening. These setups are denotated with a numbering system, the first indicating the number of speakers, then the number of subwoofers, and finally the number of speakers above the listener. The simplest of these is 5.1, including left, right, center, two surround speakers and a subwoofer. Taking it another step, we reach 7.1 that's the same as before with the addition of two rear speakers behind the listener. Finally, the most complicated form is Dolby Atmos, ideally a 7.1.4 system that adds ceiling speakers to fully surround the listener with sources.

Multi-channel audio is played by a consumer on DVD, DVD-A (audio only), SACD (Super Audio CD), Blu Ray, or by stream. These devices use specific file formats meant for compressing large audio files. The most common of these formats were both developed by Dolby Laboratories. The first is "Dolby Digital" or AC-3 (Audio Compression, generation 3), capable of handling up to 5.1 coded audio. The second, the newer ADM BWF (audio definition model broadcast wave format), a similar format expanding to support 7.1 and Atmos coded audio.[10]

The most sophisticated surround sound system that a consumer listener will experience is most likely in a movie theater. Theaters use substantial amounts of speakers to create "sweet spots" throughout the theater so no matter where you're sitting you receive a somewhat accurate sound field. At home, unless there's a dedicated theater space, soundbars and other "spatial enabled" devices may be used instead that may be lacking the level of immersion a fleshed-out system may provide.[7]

2.2.2 BINAURAL

While the term "binaural" could loosely refer to any two-channel sound that enters a listener's left and right ears, it specifically refers to two-track audio that has been filtered using a combination of human localization cues to mimic natural hearing. The goal is to re-create an acoustic signal as close to the eardrum as possible, eliminating any filtering aside from that included in the finished track. This requires the use of isolated sound sources for each ear, most easily achieved with headphones.[33]

There are a few important limitations and issues in binaural formatting that should be mentioned. Binaural audio is intended to be heard by a stationary listener. No matter how the listener moves their head, the scene around them will stay the same. This can pose an issue due to limitations of human hearing, resulting in a few undesirable phenomena. One of the most common of these is referred to as the cone of confusion, occurring when a sound source is directly in front or behind the listener. With ITD and ILD both reduced to zero, it can become impossible to determine if the source is in front or rear without turning the head to place one ear closer to the source. One possible

solution is to include headtracking functionality in the playback device. Inertial sensors are a popular choice, using accelerometers and gyroscopes to record the listener's movement. When used to manipulate the azimuth, distance, and elevation, the filtering adjusts, granting the listener the ability to look around in the auditory scene. [33]

Another undesirable occurrence is "inside-the-head-locatedness," or the feeling that the source is emanating internally, instead of externally. This is partially caused by the measurements used to achieve some forms of binaural audio. We rarely experience sound in a fully open field that does not create reflections or reverberations, and we use these as important clues to determine the distance of a sound. Some techniques used to create binaural audio rely on measurements that must be recorded, usually obtained in anechoic chambers that silence everything but original sources. Adding back in a sense of space to binaural audio created from the measurements can help to relieve this tiresome listening experience.[33]

While other methods like multi-driver headphones and ear speakers have been used to attempt to listen back to binaural audio, headphones or earbuds with headtracking seem to be the most accurate and controllable means of binaural playback.

2.3 SPATIAL PROCESSING TECHNIQUES

2.3.1 OBJECT-BASED SYSTEMS

Object-based audio systems are the newest development in multi-channel audio. The addition of elevated or ceiling mounted speakers to a typical surround sound set up creates a three-dimensional listening space. Audio material can be placed all around this

listening space as “objects.” The audio assigned to an object can be processed as desired before placement with a 3D panner. As the object is moved around, metadata is created to record its position over time. The metadata for all of the objects in the finished track is saved in the output file to be read by a “spatial-enabled” receiver, translating the positional data to the employed speaker set-up.[5]

Creating three-dimensional audio with object-based systems has become much more accessible since the original appearance of this concept. In the earlier versions of Dolby Atmos, the technology depended on four plugins, standalone Atmos Renderer software, and a physical Rendering and Mastering Unit (RMU) to work with audio in this format. By 2017, it was thankfully reduced to one plug-in and the Renderer, and today most object-based systems solely rely on multiple instances of a plugin.[6] Past the initial differences in software, each proprietary tool is quite similar. Spatial Audio Designer, WalkMix 360 (Sony 360 Reality Audio), and DearVR Pro all use multiple instances of their plug-in to route audio for three-dimensional mixing. Setting the chosen tool up for monitoring varies in complexity, but they all offer conversion to binaural audio for headphones or multi-channel routing for speakers.[1] The user can then mix as they normally would, simply using the panner in the chosen plug-in instead of the DAW’s built in stereo panner.

Once finished, there are a few different file formats that will retain the objects’ metadata. Dolby Atmos provides users the option of an ADM-BWF (Audio Definition Model Broadcast Wave Format) .wav file or an .mp4. The ADM-BWF essentially is a

typical wav file with a large header containing the metadata for objects and other information about the track. This is the deliverable most streaming services are looking for when uploading a finished Dolby Atmos mix. An .mp4 is usually thought of as a video format, but with built in storage for multi-channel audio with the Dolby Digital plus Joint Object Coding (DD+JOC) codec, it acts as a convenient vessel for playing back Dolby Atmos on consumer devices with multi-channel receivers.[5] Sony 360 takes a different approach, formatting its files with .MPEG-H 3D, but still providing a container to hold the audio and its metadata.[1] These formats allow for real-time rendering upon playback, meaning that the track is formatted as near to the original intentions as possible for any listening system.

2.3.2 AMBISONICS

Ambisonic recording is an implementation of the sound field approach to immersive audio. This approach focuses on the replication of sound waves, steering completely away from object and channel-based audio. Sound field work aims to control the physical properties of sound for reproduction where Binaural or Object-based approaches focus on manipulating the perception of sound. While the concept of Sound Field synthesis has been around since the 1930's, appearing as the basis for stereo recording techniques, Ambisonics was Michael Gerzon's brainchild. He presented the system in 1973 as a way to equally represent vertical and horizontal aspects of a sound. Using a tetrahedral array of cardioid microphones to record incoming sources, a 4-channel signal is recorded to represent the dimensions: W, X, Y, and Z. W is an

omnidirectional dimension representing all the sound captured, X contains the front and back dimension, Y: left and right, and Z: up and down. This system is referred to as 1st order Ambisonics. Increasing the Ambisonics order, therefore the amount of microphones and channels, increases the amount of dimension separation, allowing for more spatial detail in larger spaces.[27]

With any order of ambisonics, the raw microphone output, or A-Format, must be "encoded" to B-Format to represent its dimensions correctly, typically accomplished on the recorder itself or using a plug-in. Once in B-Format, the recording can be manipulated with different plug-ins to point in different directions, reflect different pick-up patterns, or even flip the entire sound-field.[19] While there is room for manipulation after the recording has been captured, the ambisonics process is focused on recreating the recorded auditory scene. While changing the listener's perspective may add something to the final product, it's still centered on replication, not creation from scratch.

2.3.3 BINAURAL RECORDING OR SYNTHESIS

There are multiple strategies to achieve binaural audio, the focus of this research lying on one in particular. Playback of a binaural recording is the simplest of these strategies. The most popular ways to record binaurally include using microphones with diaphragms located in the ear canals of a dummy head, or in the ear canals of a human listening to the source. Once a recording is obtained, the signals are simply played back through headphones, re-creating the hearing of whom or whatever's ears were used to record with.[33]

Another form of binaural audio comes in the form of a virtual "listening room". The objective is to imitate the experience of listening to loudspeakers by creating virtual point sources for audio to emanate from. Theoretically, this can be done with as many speakers as the user would like, making anything from 5.1, to 7.1.4, or even larger setups a possibility. A binaural room impulse response (BRIR), a measurement formed from a combination of others (room acoustics, loudspeaker characteristics, listener's natural filters), is superimposed in stereo pairs on a source to create a virtual loudspeaker at some position around the listener.[33]

Lastly, and mainly, binaural playback can be achieved through synthesis. This process takes a mono or stereo signal and filters it using measurements of human localization to produce a binaural signal in playback. This form is our focus because it eliminates the need to record a source in a specific manner and fully immerses the listener by removing the limitation of a virtual stationary source. The measurement needed to synthesize binaural audio is called a head related transfer function (HRTF). It is a measure of the filter system created by a listener's torso, shoulders, head, and pinnae (visible portion of the ear).[3]

A generic or individualized HRTF is chosen for the calculations along with a monophonic signal. The signal is then mathematically combined with both the left and right ear HRTFs for a specific location in the sound field to produce a two-channel output that places the source at that location. To make the source move, the HRTF must be constantly updated to match the desired motion of the source, calling for continuous

interpolation.[33]

In practice, a binaural synthesis effect would keep everything the engineer would need in their DAW of choice. A plug-in, a common audio tool that is hosted within a DAW, would accomplish this. The plug-in would be inserted on individual tracks or busses within a session so each source could be manipulated independently, or multiple

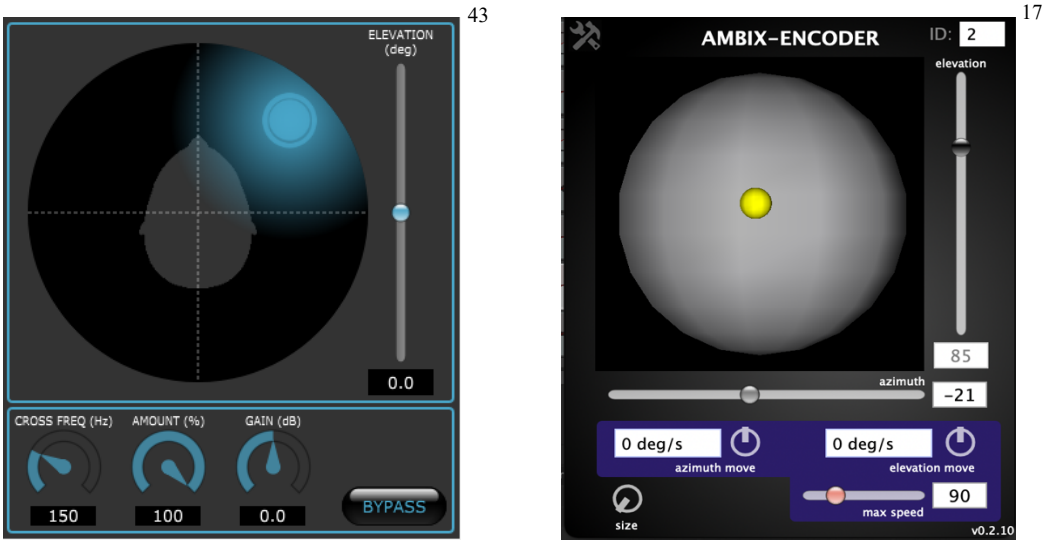


Fig. 3 Three-dimensional Panners [43] [17]

sources could be positioned in the same place. The user interface would likely look like those shown in Fig. 7, making use of a "three-dimensional" panner to allow the user to indicate the desired height, azimuth, and distance of the source from the listener's perspective. Other functions that might be useful for mixing in binaural could include a wet/dry control, changing how much "binauralization" a source receives, a size control that could spread the source over a larger area, or the ability to engage a high-pass filter that would control the amount of low frequency content passing through the binaural

effect to create a sound that behaves more like natural audio.[43]

The engineer would use this tool in conjunction with any of their typical processing tools to mix in mostly the same manner as they would a more common stereo mix, simply adding the extra panning capabilities to create something immersive. The process of saving and distributing files would not change, any binaural mix would still result in a two-channel output but will only provide the intended listening experience over headphones. Aside from being limited to headphones, the largest drawback of this mixing process would likely be the amount of strain placed on the computer running the DAW. The program would need to run as efficiently as possible to avoid overloading the system's processing power when many iterations of the plug-in are running simultaneously.

3. PROGRAM DEVELOPMENT

With an understanding of the different ways to create and hear immersive audio, the development of the plug-in can be discussed in more detail. Components of signal processing necessary to perform "binauralization", Head Related Transfer Functions, the intentions and hinderances in the program's functionality, and snippets of coding will be surveyed and discussed.

3.1 DIGITAL SIGNAL PROCESSING CONCEPTS

Digital Signal Processing (DSP) involves using mathematical algorithms to alter a signal from its original state. While it can be applied to any digital signal, like an EKG or to logical electronics, in audio, it is the basis for manipulating input signals until a desired output is achieved.

3.1.1 DIGITAL SIGNAL REPRESENTATION

The first concept to grasp in DSP is how computers read audio signals. In the analog realm, where audio is heard aloud or recorded to a physical medium like tape, sound exists as a continuous signal modeled by a function $f(t)$, where t is time and there is a value of f for every possible value of t. When audio is converted to a digital format, it is impractical, and impossible, to record each and every value of f and it must be modeled as a discrete signal, meaning data points are only available at sampled points of t, occurring at regular time intervals called the sampling interval or period, T_S , to result

in a representation notated as $f(n)$, with n as each sample and N as the total number of

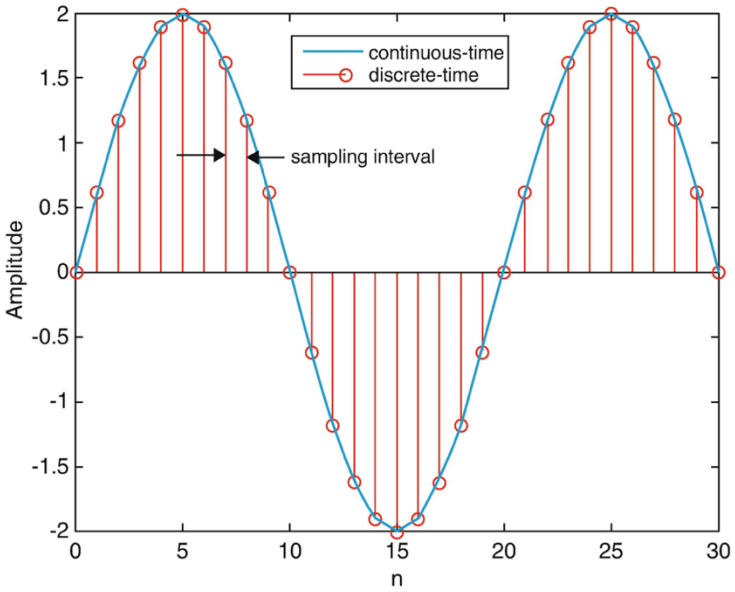


Fig. 4 Sampling Process

samples.[36] The sample rate, $\frac{1}{T_s}$, represents the number of samples taken per second. To accurately reconstruct an analog signal for playback, the sample rate must be equal to twice the value of the highest frequency represented, referred to as the Nyquist frequency. Common sampling rates include 44.1 kHz and 48 kHz to contain frequencies within the human hearing range of 20Hz to 20kHz with ample padding.[24]

These samples also occur over a continuous range of amplitude levels that must be recorded discreetly. The process of reducing the infinite resolution of signal amplitude is referred to as quantization, where each measurement must be rounded to a discrete level. Quantization introduces noise into a digital signal as the difference between the continuous amplitude values and the quantized amplitude values. The resolution of the

amplitude measurement, or bit depth, must be set high enough so that the noise is imperceptible upon analog reconstruction, and low enough that the audio signal still contains a small enough amount of data to be read, stored, and transmitted easily.[25] Common bit depths include 16 and 24 bits, the latter providing more than 16 million discrete amplitude levels.[26]

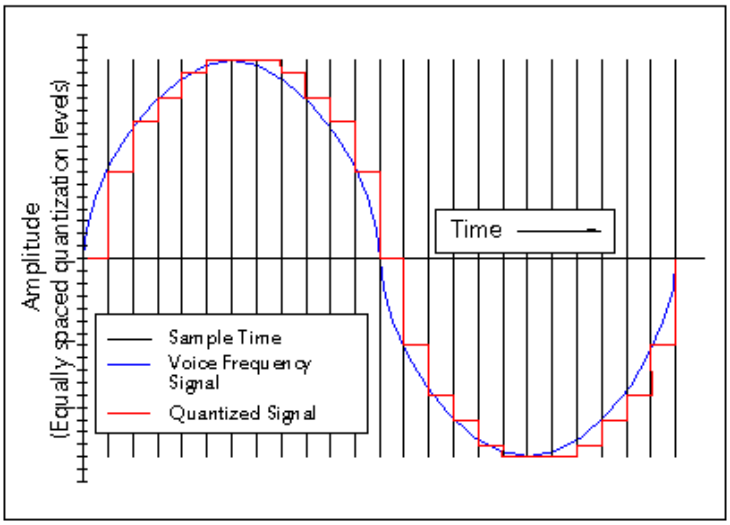
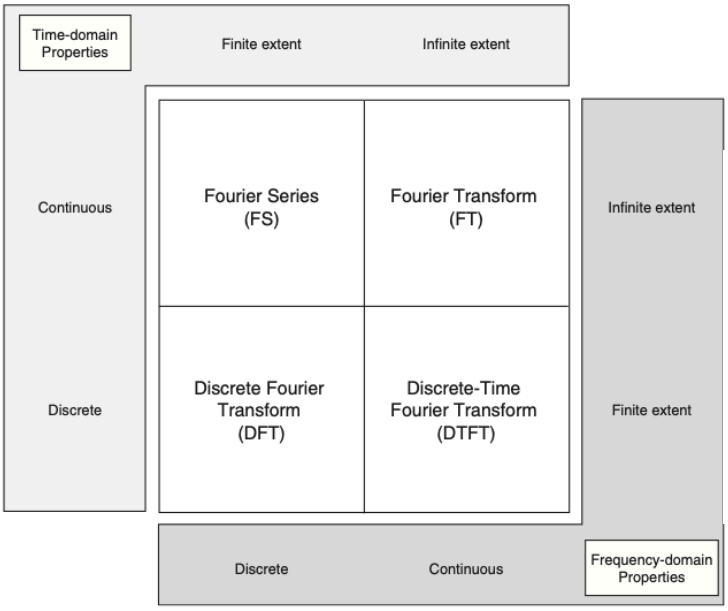


Fig. 5 Quantization Process

3.1.2 RELEVANT SIGNAL PROCESSING TECHNIQUES

With a visualization of digital signals in the time domain, the frequency domain may now be considered. This domain inserts frequency, instead of amplitude, on the y-axis, typically using color-coding to indicate the intensity of each frequency in the signal over time. Naturally occurring sounds are sums of sine waves at different amplitudes and phase offsets. We can separate a sound into its individual sine waves, or pure frequencies, using an algorithm called Fourier transformation. In theoretical math, this algorithm can

be described as an integral of a continuous function $f(n)$, producing the sum of every frequency present in the signal, but the algorithm can be specified as one of four types for more practical applications.



[18]

$$DFT = \sum_{-\infty}^{+\infty} (x[n])e^{-j\omega n}$$

Where:

j – imaginary number
ω – angular frequency
n – sample

Fig. 6 Sects of Fourier Transform and DFT Equation

The four algorithms are separated based on whether the signal to be transformed is continuous or discrete in either the frequency or time domains. For digital signals, working with the Discrete Fourier Transform (DFT) is most appropriate, due to constraints of the sampling rate and bit depth[18]. Most specifically, this project is concerned with a Fast Fourier Transform algorithm (FFT), an implementation for the practical computation of a DFT. These processes are effectively the same and referred to interchangeably in terms of DSP.[31] Essentially, computing an FFT allows us to analyze a signal's frequency spectrum or process the signal in the frequency domain, while

retaining enough information to return the signal to the time domain with an inverse FFT. Processing in the frequency domain can be useful to reduce the computing power necessary to perform the desired functions on a signal, in the context of this research, specifically convolution.

$$Convolution\ Equation: \quad y[n] = \sum_{m=-\infty}^{+\infty} x[m]h[n-m]$$

Convolution is a function where an input signal is combined with a signal representing a system to produce an output characteristic of the actual system's processing. The signal representative of the system is called an impulse response in the time domain. It is indicative of how that system responds to a single sample and can then be applied to (convolved with) any number of samples to result in the same behavior for every sample processed in linear, time-invariant systems. In other words, convolution is the superposition of scaled and time-shifted impulse responses.[15]

Superposition: $x_1[n] + x_2[n] + x_3[n] \rightarrow h[n] \rightarrow y_1[n] + y_2[n] + y_3[n]$
Scaling: $3x[n_1] + 1.2x[n_2] + .7x[3] \rightarrow h[n] \rightarrow 3y[n_1] + 1.2y[n_2] + .7y[n_3]$
Shift Invariance: $x[n-m] \rightarrow h[n] \rightarrow y[n-m]$

In practice most impulse responses are obtained through physical recording. An input signal with a value of 1 at time zero and zero for all following points is fed through a system, and the output of the system is recorded and stored as the system's impulse response. Examples of a system could be a digital filter, a piece of analog outboard gear,

or a physical location or object. A system is linear when it produces an output that corresponds to any superposition or scaling on the input side. A system is time-, or shift-, invariant when processing the input from a sample, other than its initial sample, produces the output corresponding to that sample[15]. A linear and time-invariant system produces a finite impulse response, a signal with a discrete length and frequency content perfect for practical convolution.

While using an impulse response to represent a system can simplify the filtering process, convolution in the time domain is computationally expensive. The required operations equal the square of the total number of samples in the input signal (N^2) resulting in massive quantities when considering the previously mentioned common sampling rates. Returning to the idea of FFT, the Convolution Theorem can remedy this. This theorem proves that convolution in the frequency domain is equivalent to multiplication. Using an FFT to shift both the input signal and impulse response, now a transfer function, into the frequency domain, the two signals can simply be multiplied

$$\sum_{-\infty}^{+\infty}\left(\sum_{m=-\infty}^{+\infty} x[m]h[n-m]\right)e^{-j\omega n} \qquad n' = n - m \quad , \quad n = n' + m$$

$$\sum_{n'=-\infty}^{+\infty}\left(\sum_{m=-\infty}^{+\infty} x[m] \times h[n']\right)e^{-j\omega(n'+m)} \qquad e^{-j\omega(n'+m)} = e^{-j\omega n'} + e^{-j\omega m}$$

$$\sum_{m=-\infty}^{+\infty} x[m]\, e^{-j\omega m} \times \sum_{n'=-\infty}^{+\infty} h[n']e^{-j\omega n'}$$

$$X[n] \times H[n]$$

Fig. 7 Convolution Theorem Proof, Equivalency to Multiplication

together to produce the same output as in the time domain. Then, taking the inverse Fourier transform of the resulting signal, it is transposed back into the time domain and can be read and played back as a typical audio signal. Making use of element-wise multiplication instead of convolution significantly reduces the number of operations required to $2N$. The FFT process is also computationally reasonable, the number of operations to perform the transformation sitting at $Nlog_2N$. This brings our total number of operations to $2N(log_2N + 1)$, reducing the number of operations to less than 2% of convolution in the time domain.[16]

3.2 HEAD RELATED TRANSFER FUNCTION

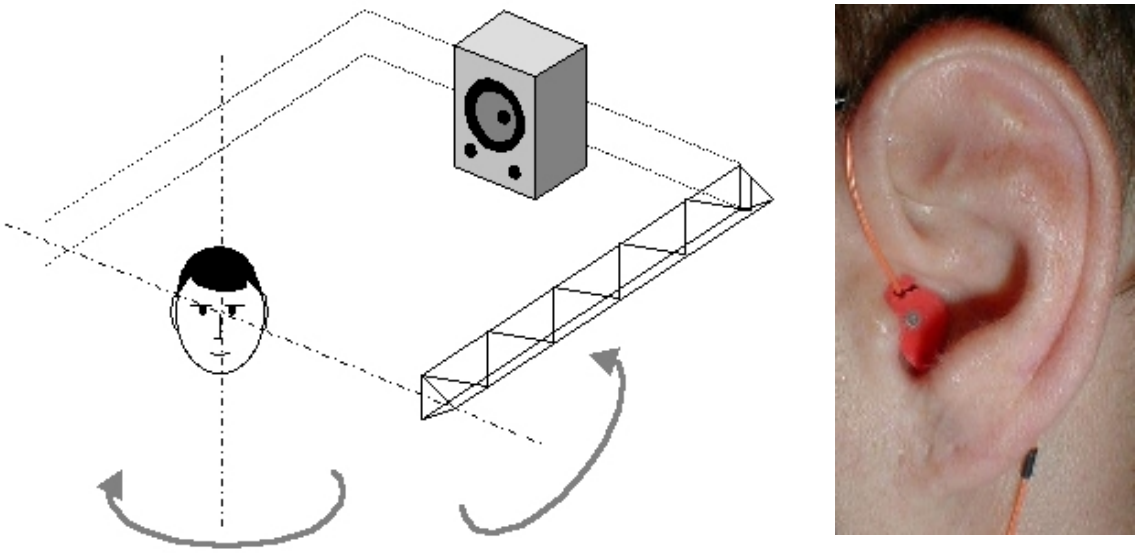


Fig. 8 Methods of HRIR Measurement, Microphone and Source Placement

A Head Related Transfer Function (HRTF) is a frequency-domain signal representative of the filter system created by a listener's torso, shoulders, head, and

pinnae (visible portion of the ear). Every human form diffracts and scatters incident sound waves slightly differently, resulting in a personal bandpass filter that slightly boosts or cuts some of the frequency content present in incoming sound. This filter is measured by recording the impulse response of a human listener, or their HRIR (HRTF time-domain equivalent).[44] The impulse response is taken by placing microphones in the blocked ear canals of a listener and playing a signal from sources all around them in three different planes: azimuth, elevation, and distance. These measurements are typically

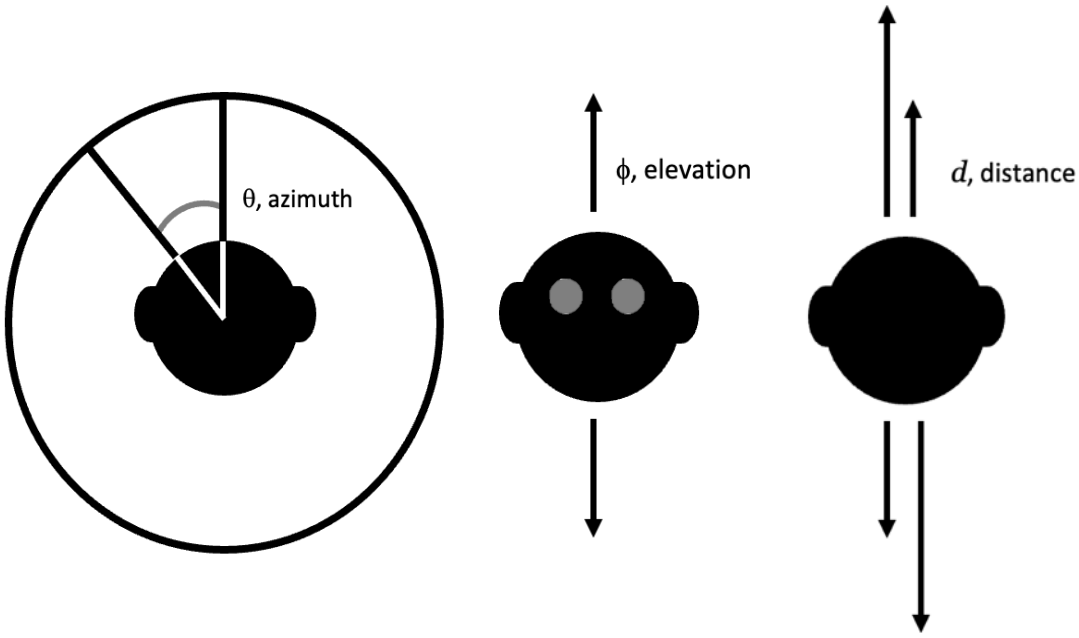


Fig. 9 Components of Three-Dimensional Plane Around the Head

taken in large batches, attempting to record as many points as possible from each subject for many subjects because HRIRs can greatly vary based on the subject's anatomy.[25]

If no personal HRTF is available, choosing an HRTF to perform calculations with would seem like a complicated process extremely dependent on the intended listener, but

surveys of large batches of HRTFs mostly disprove that worry. Averaging HRIR measurements typically results in similar ITDs (inter-aural time differences) and ILDs (inter-aural level differences), meaning an arbitrary HRTF provides reasonable azimuth perception. Elevation is the component hardest to reflect accurately without using individualized HRTFs but can still be perceived without the need for a personalized measurement.[42]

There have also been efforts to standardize the method of storing HRTFs. While there are multiple format factions, SOFA (Spatial Oriented Format for Acoustics) created by the AES (Audio Engineering Society) is one of the most popular choices, and it's obvious why. The format standardizes descriptions of data and contains measurements for one or more listeners or distances along with relevant metadata describing methods of measurement in a compressed binary file for convenient storage and transfer. There are nearly 30 publicly available databases of SOFA formatted HRTF measurements for over 700 human subjects as well as HRTFs of dummy heads and numerically calculated sets. Furthermore, many developers have released APIs (Application Programming Interface) to work with the file format in different programming languages and applications, the functions available for MATLAB proving extremely useful.[35]

Choosing from the many available HRTF databases was a task focused on simplicity. Not quite knowing what to expect from a .SOFA file, choosing a smaller and well-organized database seemed to be the obvious decision. The LISTEN Project had a sample of 18 listeners, a constant distance, and a specific number of data points for each

dimension, lending itself as a good option for starting small. Their measured HRTFs were recorded in anechoic room using a mechanically rotating chair and crane system to perform azimuth and elevation changes, resulting in 187 measurements per listener.[12]

Elevation (degrees)	Azimuth increment (degrees)	Points per elevation
-45	15	24
-30	15	24
-15	15	24
0	15	24
15	15	24
30	15	24
45	15	24
60	30	12
75	60	6
90	360	1

Fig. 10 LISTEN HRTF Database Measurements per Listener

3.3 FUNCTIONALITY

3.3.1 REAL-TIME OR WHOLE FILE PROCESSING

In previous programming endeavors, the work never left the digital audio workstation (DAW), meaning it was not necessary to consider how the audio files were processed. The DAW handled the audio independently of anything programmed into the plug-in. Working with MATLAB however, there was no DAW to rely on. To attempt VST implementation, it would be necessary to understand how audio is processed in real-time.

When working in a DAW, one track of imported audio could be played many times, started and stopped in different places, combined, separated, or trimmed. These abilities are provided via "real-time" processing. Whenever audio is played, the program is reading it piece by piece, preparing the next piece as the current piece plays and so on until playback is stopped or there is no more audio. The "next piece" is stored in a "buffer" until it is ready to be played. When a plug-in is added into the path, it functions in the same manner, receiving audio from the DAW's buffer, processing, and returning it before it is time for that piece to play out loud.

There are a few ways working in real time can be simulated or achieved using MATLAB. The simulation involved reading in a whole audio file, and subsequently splitting it into sections of a typical buffer size. The sections may then be processed in the desired manner and added back together, written into a new audio file, and then played back as a whole file. MATLAB's DSP Toolbox also contains a pre-written object $AudioFileReader$ that will read through an audio file much like a DAW would, calling its buffer an "audio frame". Its counterpart, $AudioDeviceWriter$, then reads the output for playback through the selected audio device. Lastly, the processing code can be framed as a class, and tested using MATLAB's "Audio Test Bench" that simulates a DAW, using the built-in functions described above to import and export audio without the need to include them in the code.

3.3.2 INTERPOLATION AND HRTF ACCESS

As previously stated, HRTF measurement is obtained by recording the impulse

response of a listener. While a binaural microphone or a dummy head can be used as the "listener" in this process, recording with a human subject is typically preferrable. Unfortunately, this limits the number of points that can be measured per subject as humans tire of sitting still for long periods of time. Researchers must limit the variance in azimuth and elevation to shorten the amount of time necessary to complete a single subject's HRTF database. This necessitates interpolation between recorded HRTFs to provide the perception of a continuous free field around the listener. While truly constant data would be nearly impossible to achieve, narrowing the gap between each available HRTF improves the transition between different placements and offers more selections in static source placement.[11]

There have been many propositions on how best to accomplish this. There are direct methods based on HRTFs adjacent to the desired source location, implemented on an inter-positional transfer function basis, with tetrahedral interpolation, or bilinear interpolation. In-direct methods pool the HRTFs for any source position as combinations of a basis function like a Fourier Bessel Series or a parametric model. Calculations can occur in the time or frequency domain, often resulting in more success with the latter. While numerous experimental methods have also been proposed, they quickly delve into a level of computation that escape the scope of this research.[32]

Of the more established methods, tetrahedral interpolation appears to be the most thorough of the aforementioned methods, capable of creating data for any point in a virtual three-dimensional space. Using an HRTF database with varied distance (in

addition to elevation and azimuth), Delaunay Triangulation can be applied to determine the four points of a tetrahedron that fully encloses the desired location. Calculating the scalar weight of each point, the desired point can now be calculated using the four adjacent points. This method typically results in points with less error than other direct methods, the mean square error and spectral distortion noticeably smaller.[11]

A binauralization tool would reap large benefits from an interpolation system, not only for added degree of freedom, but for real time and automated manipulation of source locations. Tetrahedral interpolation with three-dimensional HRTF data has the potential to create a dense sphere of points around the listener for an extremely immersive experience. Despite these massive advantages, implementing something like this would be extremely difficult at the current level of this research, requiring a dedicated effort to the topic.

3.3.3 BINAURAL PROCESSING METHOD

Putting it all together, the signal processing method for "binauralization" is a combination of the techniques described above. The essential process consists of transposing the input signal and HRIR for the chosen location to the frequency domain, multiplying the signals together, and returning the resulting signal to the time domain for saving and playback. In practice that process was completed in eight sections.

1) Read an input signal – First the program needs to see the user's input signal and obtain information like its sample rate, bit depth, and the total number of samples in the file.

2) Choose Source Location – The program must now find and read an HRIR for the placement location of the user's choice. In a sophisticated program, if there was not an HRIR for the chosen location, there would be a system in place for interpolating the desired HRIR from the closest points available. A less complicated program might remedy this by quantizing the user's location to the nearest available data point or turn to the method seen in the current version of the proposed program, providing a list of available data points and producing a "choose again" error message if the user ignored it the first time around. The program would then need to gather information about the HRIR in the same manner as the input audio including its sample rate, bit depth and its length in samples.
3) Compare Gathered Information and Prepare for Processing – Once both audio files have been chosen and read, the bit depth and sample rates would be considered, and the user input would be resampled to match the HRIR. To correctly process the FFT, both signals must have a length of: $(input\ length + HRIR\ length) - 1$.[31] Each signal would be padded with trailing zeros (extra samples at the end of the signal) to achieve the desired length.
4) Fast Fourier Transform – Both signals are now ready for transposition to the frequency domain. Existing functions within the programming language can be used to perform the task.

5) "Convolution" – With both signals in the frequency domain, the convolution theorem is taken advantage of, and each corresponding sample is multiplied into two resulting signals, the left and right channels of the desired output. They are then combined into a single two-channel signal and ready to return to the time domain.
6) Inverse Fast Fourier Transform – Returning to the time domain is trickier than leaving it was. Despite both original signals existing practically and in the real plane, they may return from the frequency domain as a signal full of complex numbers that will not reconstruct into an intelligible signal. For this program's purposes the complex number can be ignored in favor of the real part, so that must become a part of the inverse FFT function applied.
7) Save as New Audio File – Safely returned to the time domain, the new signal can be designated as an audio signal and saved as a .wav.
8) Output – The program would now read in the newly saved .wav to gather the information necessary to play it back for the user.

3.4 DEVELOPMENT

3.4.1 CODING PLATFORM AND LANGUAGE

For this project, visual block coding platforms or Reaper's JS language did not feel appropriate for the magnitude of processing performed. Platforms like MATLAB or JUCE appeared to have much more functionality built into them and shared the potential

to turn code into a VST, aligning much better with the research goals. To create a plug-in, JUCE expected an algorithm in C++, a language that can be difficult to learn to write and understand for testing.[30] MATLAB used a language all its own, loosely referred to as m, that a user could be working with within an afternoon. MATLAB is also mainly used for mathematical applications, meaning it had built in functions like $fft(x)$ and $conv(x, h)$, making it much simpler to test code and learn from it. MATLAB's audio plugin capabilities furthermore allow a user to generate a VST from a previously created MATLAB class, meaning the software's development could unfold all in one place. Once the platform was decided on, learning the language began. A fundamental aspect of the m language is its focus on batch processing. MATLAB is an abbreviation for "Matrix Laboratory" and the language is based on working with multiple numbers at the same time, stored in matrices and arrays. Individual points of arrays and vectors may be called or stored in their own variable using array indexing. Typing *array (row, column)* produces the value stored in that position. The indication *:* also allows for extraction of entire rows or columns to smaller vectors. When taking an audio input using the function *audioread*, the language stores it as a vertical vector for mono or a two-column matrix for stereo. The program can also store the audio file's sampling rate from this function, as the sample rate must be specified when attempting to play audio in MATLAB. Another function *audioinfo* further gives the user access to information like the number of samples, duration in seconds, and the input's bit depth.[37]

The next hurdle to jump was learning how to use the API provided by SOFA to

work with the HRIR/HRTF measurements. The API gave instructions on how to make MATLAB see the API so the user can access its functions. These functions, displayed in Appendix B, included things like loading the desired HRTF database into MATLAB as an object, displaying general information on the loaded database, displaying all source placements measured, and calculation of the apparent source placement from the listener perspective. These functions were indispensable, making it extremely easy to focus on implementation.[35]

3.4.2 IMPLEMENTING SOLUTIONS

Development of a MATLAB script that correctly outputs a "binauralized" sound came in steps. Before learning how to use the SOFA API, it felt necessary to first understand a more stripped-down version of the "binauralization" goal. A more common tool using similar processing techniques is a convolution reverb. Instead of using Head Related Impulse Responses, it focuses on applying recorded impulse responses from locations like canyons, specific rooms, warehouses, and other interesting environments to give input sounds a different reverberation quality, like they had resonated in that space originally. The signal processing is the same but removes the need to wade through large databases of unfamiliar data. With a basic Impulse Response and an expected output in mind, a first attempt to program the convolution reverb in the frequency domain was made. This go was failing and to get it functioning, the convolution had to be performed in the time-domain.

Once that version was functional, the issues surrounding the FFT process were explored. Two issues came to light and were worked on separately. The first involved matching the lengths of the input signal and the impulse response. Initially, it was believed that the two signals had to be the same length, so the program was designed to compare the two and pad the shorter signal with trailing zeros. More research revealed that the matching length requirement remained, but their length needed to equal the sum of their individual lengths, minus one sample for the process to work correctly.[31]

```
% initial length matching solution

diff = inputLength - irLength;
if diff > 0
    irFFT = fft(userLocation, inputLength);
    inputFFT = fft(inputAudio);
else
    irFFT = fft(userLocation);
    inputFFT = fft(inputAudio, irLength);
end
```

```
% working length matching solution

totalLength = (inputLength + irLength)-1;
inputFFT = fft(inputAudio, totalLength);
irFFT = fft(userLocation, totalLength);
```

Fig. 11 Zero-Padding Techniques

Once this issue was resolved, MATLAB was unable to render the result of the inverse FFT (iFFT) as an audio signal. Looking into the first few samples of the result of the iFFT showed that the samples were complex values such as $3 + .068i$. While complex numbers on the imaginary plane can be interpreted as an audio signal, as far as MATLAB was concerned, the process could not be completed without resolving the

sample values to real numbers. A few attempts were made to try to convert the complex number to the real plane using complex conjugates, but that method resulted in empty playback.[31] While hesitant that it would be a solution, the algorithm was amended to include the built-in function $real(x)$ while performing the iFFT to throw away the imaginary portion of the complex number. With this in place, MATLAB finally recognized the result of the iFFT as audio. When compared to the result of the previous iteration's time domain convolution, the two signals were extremely similar in sound and sample values, differences between them appearing at a resolution of .0001 or smaller. Considering that a success, it was time to begin working with the HRIRs.

Beginning to explore the SOFA API revealed that it was rather easy to work with after understanding a few things. MATLAB's version of object-oriented programming (OOP) is documented as "structure arrays". These "structs" can contain nested and labelled data referred to as fields and accessed by dot indexing (object.field). Fields can contain single values, arrays, matrices, and even other structs. The most important fields were *SourcePosition* and *Data*. Navigating to *SourcePosition*, a table with source points is displayed, so the user can determine the index of the source position they would like to apply. Accessing *Data* reveals the sample rate, units of delay if the IR contains them, and the location of the HRIRs themselves. The next concept to chew on was how the HRIRs are stored. Matrices in MATLAB are not limited to a two-dimensional plane, many often taking on a 3rd or even 4th dimension past rows and columns. In a three-dimensional matrix the extra dimension is referred to as pages. In hrtf.Data.IR, the row numbers refer

to a source position (1-187), the columns acting as left and right channels (1 or 2), and the pages as each sample of the 512n long IR. To call one of these IRs, a combination of array and dot indexing is used, *hrtf.Data.IR(row, : , :).* This calls all columns and pages of the desired row, resulting in a piece of data stored as a $1 \times 2 \times 512$ three-dimensional matrix. To process the two-dimensional input audio, it is necessary to remove the extra dimension. Thankfully, MATLAB has a function for this, *squeeze*. This function removes any dimension of length 1, shifting the other dimensions to the next less complex dimension, resulting now in a 2×512 matrix. Performing one more transposition, we can swap the rows and columns and finally have a signal in the same format as our input. With enough knowledge to determine what the chosen row should sound like and how to call it, it was time to insert the HRIRs into the homemade "binauralization" framework.

3.4.3 VERSIONS

To much joy, the first attempt to implement the HRIRs (Appendix C) was successful. It reproduced sounds at the expected locations and a simple modification to the script gave the user control over which IR they wanted to use. This version was solely focused on functionality, usability coming with the second version. The next step was a solution to prevent the user needing to edit the script itself.

MATLAB receives user input using the terms *prompt* and *input(prompt)*. The user input can then be stored as variables and used throughout the rest of the program. In this project, user input was obtained to give control over which HRIR database to use, what input to use, and where the source should come from. Knowledge and time constraints

barred the creation of a search or interpolation function, but there needed to be a way to find the desired IR without the need to mentally run through the list of available measurements or try to count the row from the *SourcePosition* matrix. The first instinct was to look to the SOFA API's function, *SOFAcalculateAPV*, which converts the *SourcePosition* matrix to the apparent source vector to the listener. Unfortunately, the output of the function is terribly similar to the original matrix. The only difference in the two was the notation of the azimuth, changing from a description in $360°$ to $\pm 180°$, which seemed to only worsen the issue of determining the desired IR. The next approach was to modify the SOFA API's function to provide a table array, a variable that prints labels at the top of each column. Adding an extra column containing each row's number, the resulting table provided the user with the information needed to choose an IR and input its corresponding index so the program could call it. While the table accomplished what was needed of it, calculating the unlabeled apparent source vector matrix was unnecessary and going unused. Preferring the $360°$ azimuth notation anyway, providing the same table from the *SourcePosition* matrix made the program more efficient and easier for the user to interpret.

Smush_02 (Appendix D) also employed looping to make it a more user-friendly program, giving the option of repositioning the sound multiple times, changing the desired input, or replacing the test HRTF database with another .sofa formatted HRTF. This was accomplished with while loops, a programming tool that cycles through a section of code based on a conditional statement. Simple yes or no user inputs changed

variables that the while statement would test, prompting the loop to continue or break, progressing to the next segment of code or ending the run of the program. With this addition, a user could run the program as desired with much less instruction overall.

The last version completed, Smush_03 (Appendix E) focused on implementing real-time processing. Using a real-time simulation in MATLAB, an incoming signal was fed through a while loop that split the signal into frames of 1024 samples. Still within the while loop, each frame was processed as usual with the resulting signal still held in K. The complex part of real-time processing was that the output was longer than the input by the length of the HRIR minus one sample. For the LISTEN database used in testing, this meant each frame was now 511 samples too long, causing the output Z to playback incorrectly when pieced together from K. To remedy this K was split into two pieces: L the first 1024 samples, and M, a "save buffer". M was required to hold on to the extra samples from the first frame's processing to later be summed with the next frame. The process of hold and sum then carried through the whole track, the output, Z, receiving correctly timed sums of M and Z, the last frame consisting of 511 extra processed samples and 513 zeros to account for the last tail. The while loop breaks its cycle with a test to prove it's gone through the entire input file, and the new output file is written from Z to be heard out loud.

3.4.4 FURTHER RESEARCH AND IMPLEMENTATION POSSIBILITIES

While the current implementation is functional in multiple aspects, it is not quite at a point of full VST implementation. Solving the issues with real-time implementation was a great step, but some further changes would be necessary to continue moving forward on that journey. MATLAB employs the functions *validateAudioPlugin* and *generateAudioPlugin* to make a standalone software entity from a class of code in the m language. The current program implementations are formatted as MATLAB scripts and would need to be re-written in terms of methods and properties to fit this condition. Furthermore, where SOFA had been extremely helpful when implementing HRTF access, it would prevent validation of a new *audioPlugin* class because it is a class itself with its own variable properties. Both issues would require further research into object-oriented programming (OOP) with MATLAB or workarounds that might additionally reduce functionality within a VST.

Lastly, with no true search or interpolation system developed yet, there still lies much more work to be done to develop a graphic user interface that would function well for this type of processing. Good options could include a three-dimensional panner like seen previously, or simply a few controls allowing the user to input the desired azimuth, elevation, distance from the head. This graphic user interface can be designed using MATLAB objects *audioPluginInterface*, *audioPluginParameter*, and *audioPluginGridLayout,* dictating their own period of further research as well.

Implementation of the “binauralizer” into a VST plug-in is a goal that may be

achieved with further research into the topics discussed above. While finishing this research at a point of brushing the ultimate goal leaves something to be desired, the wealth of knowledge gained on three-dimensional audio in all its formats, DSP, and implementation through coding in MATLAB has transformed a fanciful idea into an attainable objective.

APPENDIX A: MATLAB BASICS

MATLAB's language is focused on processing data in batches, different forms of matrices and arrays are common methods of storing data.

Horizontal Vector/Array:

```
>> a = [1, 2, 3, 4, 5, 6, 7, 8, 9]

a =

     1     2     3     4     5     6     7     8     9
```

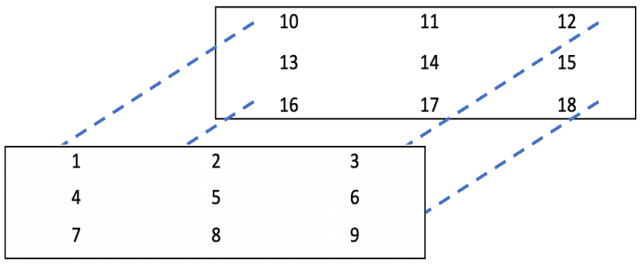
Vertical Vector/Array:

```
>> b = [1; 2; 3; 4; 5; 6; 7; 8; 9]

b =

     1
     2
     3
     4
     5
     6
     7
     8
     9
```

3x3 Matrix Array:

```
>> c = [1,2,3;4,5,6,;7,8,9]

c =

     1     2     3
     4     5     6
     7     8     9
```

Three-dimensional Matrix:




```
>> c(:,:,2) = [10, 11, 12; 13,14,15; 16,17,18]

c(:,:,1) =

     1     2     3
     4     5     6
     7     8     9


c(:,:,2) =

    10    11    12
    13    14    15
    16    17    18
```

There are a few different ways to import and export audio with MATLAB. *audioRead* and *sound* work together as built in functions, *dsp.AudioFileReader* and *audioDeviceWriter* a built-in object pair, and finally the Audio Test Bench are all methods of importing audio. *audioInfo* can also be used to get important data like the sample rate, bit depth, and length of the audio imported.

Basic Audio Functions:

```
>> audioInfo = audioinfo("/Users/sydneysumner/Downloads/AnnaBlanton_Rachel/06_Cello.wav")

audioInfo =

  struct with fields:

             Filename: '/Users/sydneysumner/Downloads/AnnaBlanton_Rachel/06_Cello.wav'
    CompressionMethod: 'Uncompressed'
          NumChannels: 1
           SampleRate: 44100
         TotalSamples: 2033354
             Duration: 46.1078
                Title: []
              Comment: []
               Artist: []
        BitsPerSample: 24
```

```
filename = 'newfile.wav';
audiowrite(filename,audioInput,sR,'BitsPerSample',24);
```

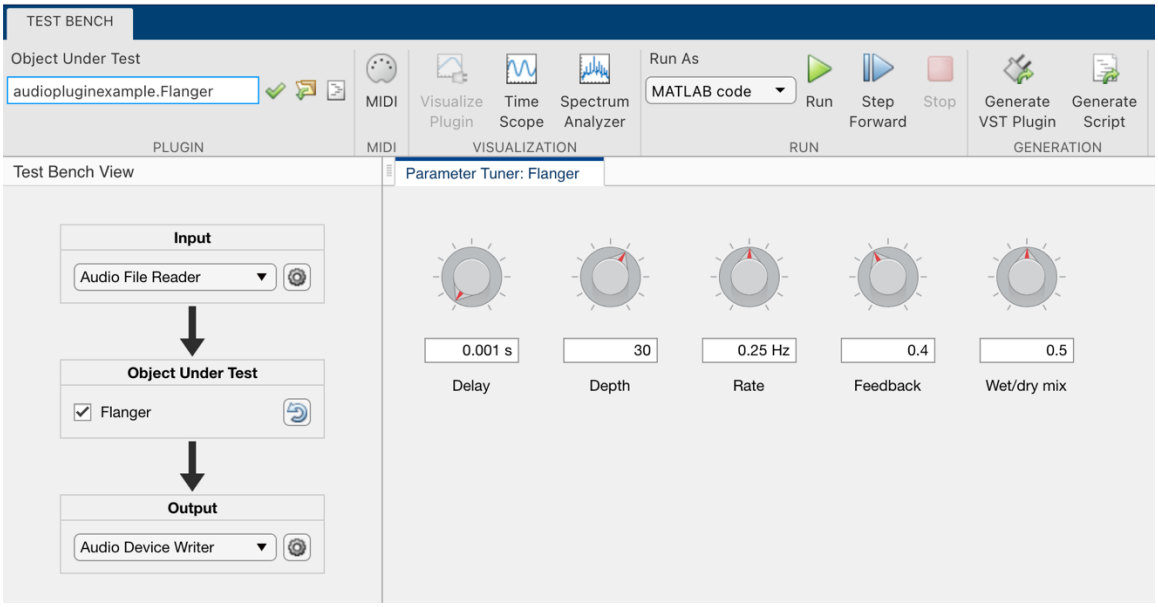
Read and Sound with Functions:

```
>> [audioInput, sR] = audioread('Gtr.wav');
>> sR

sR =

       44100

>> sound(audioInput,sR);
```

Read and Sound with Objects:

```
>> frameLength = 1024;
fileReader = dsp.AudioFileReader( ...
    'Gtr.wav', ...
    'SamplesPerFrame',frameLength);
deviceWriter = audioDeviceWriter( ...
    'SampleRate',fileReader.SampleRate);

while ~isDone(fileReader)
    signal = fileReader();
    deviceWriter(signal);
end

release(fileReader)
release(deviceWriter)
```

Audio Test Bench:



User Input:

```
prompt = "Enter the file path of .sofa HRTF: ";
filename = input(prompt);

prompt = "Enter the file path of audio to binauralize: ";
filename = input(prompt);

prompt = "Choose Source Position from table: ";
index = input(prompt);
```

APPENDIX B: SOFA API

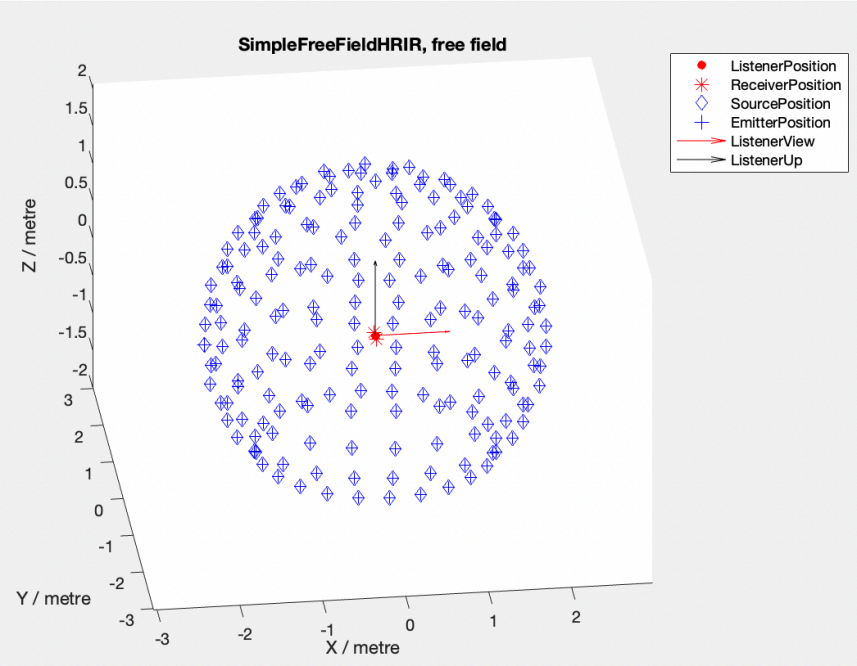
Loaded HRTF as Structure Array:

```
hrtf =

  struct with fields:

            GLOBAL_Conventions: 'SOFA'
                GLOBAL_Version: '1.0'
        GLOBAL_SOFAConventions: 'SimpleFreeFieldHRIR'
  GLOBAL_SOFAConventionsVersion: '1.0'
               GLOBAL_DataType: 'FIR'
               GLOBAL_RoomType: 'free field'
                  GLOBAL_Title: '1002'
            GLOBAL_DateCreated: '2015-04-14 16:25:30'
           GLOBAL_DateModified: '2017-04-24 14:58:13'
                GLOBAL_APIName: 'SOFA C++ API'
             GLOBAL_APIVersion: '1.1.0'
          GLOBAL_AuthorContact: 'hrtf@ircam.fr'
           GLOBAL_Organization: 'Institut de Recherche et Coordination Acoustique/Musique (IRCAM)'
                GLOBAL_License: 'Copyright :↵Copyright (c) 2014 - 2017 IRCAM. All Rights Reserved'
       GLOBAL_ApplicationName: 'libspat'
    GLOBAL_ApplicationVersion: '7.1.0 (build Fri - 21/04/2017)'
               GLOBAL_Comment: 'The head-related impulse responses (HRIR) have been measured in IRCAM's anechoic chamber'
               GLOBAL_History: 'raw measurement, windowing, diffuse field equalization, truncation'
            GLOBAL_References: 'Olivier Warusfel - LISTEN HRTF database'
                GLOBAL_Origin: 'http://www.hrtf.ircam.fr'
          GLOBAL_RoomShortName: 'IRCAM Anechoic Room'
       GLOBAL_RoomDescription: 'All measurements were performed in IRCAM's full anechoic chamber.'
          GLOBAL_RoomLocation: 'IRCAM, Paris, Anechoic Room (Studio 7)'
     GLOBAL_ListenerShortName: '1002'
    GLOBAL_ListenerDescription: 'Human subject with microphones positionned at the entrance of the blocked ear canal.'
        GLOBAL_SourceShortName: 'Tannoy 600'
     GLOBAL_SourceDescription: 'One single Tannoy 600 loudspeaker'
     GLOBAL_ReceiverShortName: 'Knowles FG3329'
    GLOBAL_ReceiverDescription: 'Knowles FG3329. A custom preamplifier was designed, with 40 dB gain.'
       GLOBAL_EmitterShortName: 'Tannoy 600'
    GLOBAL_EmitterDescription: 'Tannoy system 600 driven by a Yamaha P2040 amplifier that delivers up to 20 watts into 8 ohms loads.'
          GLOBAL_DatabaseName: 'IRCAM LISTEN'
                     GLOBAL_URL: 'IRC_1002_C_44100.sofa'
              GLOBAL_RoomVolume: '103'
                            API: [1×1 struct]
                           Data: [1×1 struct]
                     ListenerUp: [0 0 1]
               ListenerPosition: [0 0 0]
         ListenerPosition_Type: 'cartesian'
        ListenerPosition_Units: 'metre'
                  ListenerView: [1 0 0]
              ListenerView_Type: 'cartesian'
             ListenerView_Units: 'metre'
                EmitterPosition: [0 0 0]
          EmitterPosition_Type: 'cartesian'
         EmitterPosition_Units: 'metre'
              ReceiverPosition: [2×3 double]
          ReceiverPosition_Type: 'cartesian'
         ReceiverPosition_Units: 'metre'
                 SourcePosition: [187×3 double]
            SourcePosition_Type: 'spherical'
           SourcePosition_Units: 'degree, degree, metre'
                     RoomVolume: 103
                RoomVolume_Units: 'cubic metre'
```

SOFAinfo Function:

```
>> SOFAinfo(hrtf)

1002
====

Anechoic HRTF mesurement done by Institut de Recherche et Coordination Acoustique/Musique

Contact: hrtf@ircam.fr
License: Copyright :
Copyright (c) 2014 - 2017 IRCAM. All Rights Reserved
IRCAM - 1, place Stravinsky 75004 PARIS FRANCE
URL: http://www.hrtf.ircam.fr
Reference: Olivier Warusfel - LISTEN HRTF database

Measurement details:
--------------------
Number of azimuth angles:      24
Number of elevation angles:    10
Number of radii:                1
Sampling Rate: 44100 hertz
Dummy head: 1002
Loudspeaker: One single Tannoy 600 loudspeaker
Listener at (0.0,0.0,0.0) metre
187 source positions from (-0.0,-45.0,2.1) degree, degree, metre to (-0.0,90.0,2.1)
```

SOFAplotGeometry Function:



Additional SOFA Functions:

```
>> hrtf.SourcePosition

ans =

         0  -45.0000    2.0600
   15.0000  -45.0000    2.0600
   30.0000  -45.0000    2.0600
   45.0000  -45.0000    2.0600
   60.0000  -45.0000    2.0600
   75.0000  -45.0000    2.0600
```

```
>> hrtf.Data

ans =

  struct with fields:

          SamplingRate: 44100
    SamplingRate_Units: 'hertz'
                 Delay: [187×2 double]
                    IR: [187×2×512 double]
```

APPENDIX C: SMUSH_01.m

```
%% Smush_01.m

%% first attempt at adding fft,conv,and hrtf together
%changing letter of audio/IR every time it is processed, A-D
input, F-H IR, else processed.

%% input audio, need length and sR

[A,sR] = audioread("arbitrarymono.wav");
infoA = audioinfo("arbitrarymono.wav");
lengthA = infoA.TotalSamples;
SOFAstart;
hrtf = SOFAload(['arbitraryHRTF.sofa']);

%% choose hrtf measurement and format to match input audio
%extract measurement, remove extra dimension,and transpose

F = hrtf.Data.IR(180,:,:);
G = squeeze(F);
H = G';
lengthH = size(H,1);

%% match length for fft

totalLength = (lengthA + lengthH)-1;

%% fft
B = fft(A,totalLength);
I = fft(H,totalLength);

%% convolve
JLeft = B.*I(:,1);
JRight = B.*I(:,2);

J = [JLeft,JRight];

%% ifft
K = real(ifft(J));

%% write audio file
filename = 'smush_01_results.wav';
audiowrite(filename,K,44100,'BitsPerSample',24);

outPut = audioread('smush_01_results.wav');
outPutInfo = audioinfo('smush_01_results.wav')

%% sound
sound(outPut,sR);
```

APPENDIX D: SMUSH_02.m

```
%% second attempt at all processing
%changing letter of audio/IR every time it is processed, A-D
input, F-H IR, else processed.

SOFAstart;

hrtfgo = 'y';

while hrtfgo == 'y'
    prompt = "Enter the file path of .sofa HRTF: ";
    filename = input(prompt);

    hrtf = SOFAload(filename);
% test hrtf  'arbitraryHRTF.sofa'
    hrtf.Data
    SOFAdisplaySources(hrtf);

    inputgo = 'y';

    while inputgo == 'y'

     %% input audio, need length and sR

        prompt = "Enter the file path of audio to binauralize: ";
        filename = input(prompt);

     % test audio "artibtrarymono.wav"
        [A,sR] = audioread(filename);
        infoA = audioinfo(filename);
        lengthA = infoA.TotalSamples;

        processgo = 'y';

        while processgo == 'y'

            %positioning loop return

            prompt = "Choose Source Position from table: ";
            index = input(prompt);

            F = hrtf.Data.IR(index,:,:);
            G = squeeze(F);
            H = G';
            lengthH = size(H,1);

            %% match length for fft
            totalLength = (lengthA + lengthH)-1;

            %% fft
```

```
            B = fft(A,totalLength);
            I = fft(H,totalLength);

            %% convolve

            JLeft = B.*I(:,1);
            JRight = B.*I(:,2);

            J = [JLeft,JRight];

            %% ifft

            K = real(ifft(J));

            %% write audio file
            prompt = "Designate file path of binauralized
            audio:";
            filename = input(prompt);
            % path: 'arbitarysavepath.wav'
            audiowrite(filename,K,44100,'BitsPerSample',24);

            outPut = audioread(filename);
            outPutInfo = audioinfo(filename)

            %% sound

            sound(outPut,sR);

            prompt = "reposition? [y/n]:";
            processgo = input(prompt,"s");

            clearvars -except processgo hrtf A sR infoA lengthA
        end

    %% clear to start at new Source Position same input
        prompt = "new input, keep HRTF? [y/n]:";
        inputgo = input(prompt,"s");
        clearvars -except hrtf inputgo
    end

%% clear to change input or hrtf
    prompt = "new HRTF? [y/n]: ";
    hrtfgo = input(prompt, "s");
    clearvars -except hrtfgo
end
```

APPENDIX E: SMUSH_03.m

```
%attempt at real time processing

%% basics
SOFAstart;

    hrtf = SOFAload('arbitraryhrtf.sofa');

    %hrtf.Data
    %SOFAdisplaySources(hrtf);


    [A,sR] = audioread('arbitrarymono.wav');
    infoA = audioinfo('arbitrarymono.wav');

    lengthA = infoA.TotalSamples;

    prompt = "Choose Source Position from table: ";
    dex = input(prompt);

    F = hrtf.Data.IR(dex,:,:);
    G = squeeze(F);
    H = G';
    lengthH = size(H,1);

done = 0;
index = 1;
Z = [0 0];
Y = zeros(513,2);
N = [0 0];
gain = .316227766;

%% while loop

while ~done
    elem = A(index:index+1023);

    %% process elem

    TL = (1024 + lengthH)-1;

     %% fft
        B = fft(elem,TL);
        I = fft(H,TL);


    %% convolve

        JLeft = B.*I(:,1);
        JRight = B.*I(:,2);
```

```
            J = [JLeft,JRight];

        %% ifft

            K = real(ifft(J))* gain;

        %% buffering

            L = K(1:1024,:); %L is 1024 in real time
            M = K(1025:end,:); %M is saved 511
            O = L + N;  %add save buffer to next frame
            N = [M; Y]; %zero pad m for correct addition

            if size(Z,1) < 1026
                Z = [Z; L];
            else
                Z = [Z; O];
            end

            index = index + 1024;

            if index > lengthA - 1023 - 1024
                done = 1;
            end
    end

    %% write audio file

            % path:

audiowrite('smush_03_results.wav',Z,44100,'BitsPerSample',24);

            outPut = audioread('smush_03_results.wav');
            outPutInfo = audioinfo('smush_03_results.wav')

            %% sound

            sound(outPut,sR);
```

REFERENCES

1. 360 Reality Audio for creators - Create immersive music without limits. Sony. Accessed April 25, 2022. https://www.sony.net/Products/create360RA/
2. 360 Reality Audio now available on Amazon Music Unlimited with any headphones. Sony.com. Accessed April 25, 2022. https://www.sony.com/en_us/SCA/company-news/press-releases/sony-electronics/2021/360-reality-audio-now-available-on-amazon-music-unlimited-with-any-headphones.html
3. Ben-Hur Z, Alon D, Mehra R, Rafaely B. Binaural Reproduction Based on Bilateral Ambisonics and Ear-Aligned HRTFs. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, Audio, Speech, and Language Processing, IEEE/ACM Transactions on, IEEE/ACM Trans Audio Speech Lang Process*. 2021; 29:901-913. doi:10.1109/TASLP.2021.3055038
4. Deezer launches “360 Sessions” to showcase Sony 360 Reality Audio. What Hi-Fi? Published June 16, 2021. Accessed May 2, 2022. https://www.whathifi.com/us/news/deezer-launches-360-sessions-to-showcase-sony-360-reality-audio
5. Dolby Atmos Renderer Guide. Dolby.com. Published 2018. Accessed March 5, 2022. https://professional.dolby.com/siteassets/content-creation/dolby-atmos/dolby_atmos_renderer_guide.pdf

6. Dolby Laboratories. Dolby Atmos Renderer Release Notes.; 2021. https://developer.dolby.com/globalassets/tools--media/production-tools/dolby-atmos-production-suite/dolby_atmos_renderer_v3.7_release_notes.pdf
7. Easdown R. Winning partnership. *The Age (Melbourne)*. May 23, 2019. Accessed March 1, 2022. https://search-ebscohost-com.libproxy.txstate.edu/login.aspx?direct=true&db=n5h&AN=DOC75G1JABKIKM87LD3JEP&site=eds-live&scope=site
8. Fisher J. Surround History 101. In Fisher J eds. *Instant Surround Sound.* CMP Books; 2005. Accessed November 4, 2021. https://searchebscohost.com.libproxy.txstate.edu/login.aspx?direct=true&db=cat00022a&AN=txi.b5112956&site=eds-live&scope=site
9. Geluso P. Stereo. In Rogniska A, Geluso eds. *Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio.* Routledge; 2018:1-34. Accessed Sep 1, 2021.
10. Holman T. *Surround Sound: Up and Running*. 2nd ed. (Holman T, ed.). Focal Press; 2008.
11. Hugeng H, Jovan A, Dadang G. Implementation of 3D HRTF Interpolation in Synthesizing Virtual 3D Moving Sound. International Journal of Technology. 2017;8(1):186-195. doi:10.14716/ijtech.v8i1.238

12. IRCAM Team. LISTEN HRTF DATABASE. Ircam.fr. Published 2003. Accessed April 5, 2022. http://recherche.ircam.fr/equipes/salles/listen/index.html
13. Kenny T. THREE YEARS, THREE PLAYERS Dolby, UMG/Capitol Studios, PMC And the Launch of Atmos Music. *Mix*. 2020;44(5):22-26. Accessed March 1, 2022. https://search-ebscohost-com.libproxy.txstate.edu/login.aspx?direct=true&db=f6h&AN=142821828&site=eds-live&scope=site
14. Kerins M. Understanding the impact of surround sound in multimedia. *The Psychology of Music in Multimedia.* Oxford University Press; 2013. https://oxforduniversitypressscholarshipcom.libproxy.txstate.edu/view/10.1093/acprof:oso/9780199608157.001.0001/acprof-9780199608157-chapter-16. Accessed February 28, 2022.
15. Kim Y. Applied DSP No. 7: The Convolution Theorem. Published September 11, 2021. Accessed April 5, 2022. https://www.youtube.com/watch?v=jzVy6OWbAB8
16. Kim Y. Applied DSP No. 8: Filtering via Fast Fourier transform. Published February 28, 2021. Accessed April 5, 2022. https://www.youtube.com/watch?v=u8t-h31baFE&list=PLOwIR4vHXwjiP2VFZCFXdfBflCLZIQ-2a&index=1
17. Kronlachner M. Ambix Suite.; 2020. https://github.com/kronihias/ambix

18. Kulkarni S. Frequency Domain and Fourier Transforms. Presented at: 2002. https://www.princeton.edu/~cuff/ele201/kulkarni_text/frequency.pdf
19. Leonard N. The Beginner's Guide to Ambisonics. Rode.com. Accessed April 5, 2022. https://rode.com/en/about/news-info/the-beginners-guide-to-ambisonics
20. Maes J, Vercammen M, Baert L. *Digital Audio Technology: A Guide to CD, MiniDisc, SACD, DVD(A), MP3 and DAT*. 4th ed / edited by Jan Maes and Marc Vercammen. Focal; 2001. Accessed March 1, 2022. https://search-ebscohost-com.libproxy.txstate.edu/login.aspx?direct=true&db=cat00022a&AN=txi.b5099539&site=eds-live&scope=site
21. Margolis G. Format wars II - SACD vs. DVD-audio. Audiophile Review. Published March 21, 2017. Accessed March 1, 2022. https://audiophilereview.com/cd-dac-digital/format-wars-ii-sacd-vs-dvd-audio/
22. Margolis G. Why super audio CD failed. Audiophile Review. Published December 30, 2016. Accessed March 1, 2022. https://audiophilereview.com/cd-dac-digital/why-super-audio-cd-failed/
23. Marshall L. Do People Value Recorded Music? Cultural Sociology. 2019;13(2):141-158. doi:10.1177/1749975519839524
24. Murthy A. 2. Sampling Theorem - Digital Audio Fundamentals. Published 2020. https://www.youtube.com/watch?v=vrXGaFV1AmE
25. Murthy A. 5. Quantization - Digital Audio Fundamentals. Published 2020. https://www.youtube.com/watch?v=1KBLguIXL30&t=426s

26. Murthy A. 6. Bit Depth - Digital Audio Fundamentals. Published 2020. https://www.youtube.com/watch?v=X4JEMCQMwOM&t=148s

27. Nicol R. Sound Field. In: Roginska A, Geluso P, eds. Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio. Routledge; 2018. https://search-ebscohost-com.libproxy.txstate.edu/login.aspx?direct=true&db=cat00022a&AN=txi.b5161643&site=eds-live&scope=site

28. Pendlebury T. DTS:X: The immersive audio standard, explained. CNET. Published September 26, 2016. Accessed May 2, 2022. https://www.cnet.com/tech/home-entertainment/dts-x-the-dolby-atmos-alternative-explained/

29. Postrel S. Competing Networks and Proprietary Standards: The Case of Quadraphonic Sound. *The Journal of Industrial Economics*. 1990;39(2):169-185.

30. Projucer Manual. Juce.com. Accessed September 30, 2021. https://juce.com/discover/stories/projucer-manual

31. Rao K, Kim D, Hwang J. 1. Introduction, 2. Discrete Fourier Transform. In: Fast Fourier Transform - Algorithms and Applications. 2012th ed. Springer; 2012:1-40.

32. Reddy C, Hegde R. Horizontal plane HRTF interpolation using linear phase constraint for rendering spatial audio. 2016 24th European Signal Processing

Conference (EUSIPCO), Signal Processing Conference (EUSIPCO), 2016 24th European. August 2016:1668-1672. doi:10.1109/EUSIPCO.2016.7760532

33. Roginska A. Binaural Audio Through Headphones. In Rogniska A, Geluso eds. *Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio.* Routledge; 2018:1-34. Accessed Sep 1, 2021.
34. Sima S. HRTF Measurements and Filter Design for a Headphone-Based 3D-Audio System. Published online September 6, 2008.
35. SOFA (Spatially Oriented Format for Acoustics). Sofaconventions.org. Accessed April 5, 2022. https://www.sofaconventions.org/mediawiki/index.php/SOFA_(Spatially_Oriented_Format_for_Acoustics)
36. Sundararajan D. Digital Signal Processing: An Introduction. Springer; 2021. Accessed April 5, 2022. https://search-ebscohost-com.libproxy.txstate.edu/login.aspx?direct=true&db=cat00022a&AN=txi.b5650564&site=eds-live&scope=site
37. Tarr E. Hack Audio: An Introduction to Computer Programming and Digital Signal Processing in MATLAB. Routledge; 2019. Accessed April 5, 2022. https://search-ebscohost-com.libproxy.txstate.edu/login.aspx?direct=true&db=cat00022a&AN=txi.b5463091&site=eds-live&scope=site

38. Tarzan A, Alunno M, Bientinesi P. Assessment of sound spatialization algorithms for sonic rendering with headphones. *Journal of New Music Research*. 2019;48(2):107-124. doi:10.1080/09298215.2019.1572766

39. Thornton M. Avid release Pro Tools 12.8HD with complete Dolby Atmos integration. Production Expert. Published June 26, 2017. Accessed March 2, 2022. https://www.pro-tools-expert.com/home-page/2017/6/24/avid-relase-pro-tools-128hd-with-complete-dolby-atmos-integration

40. Tribbey C. Dolby Atmos Coming to Blu-Ray. *Home Media Magazine*. 2014;36(18):6. Accessed March 1, 2022. https://search-ebscohost-com.libproxy.txstate.edu/login.aspx?direct=true&db=ofm&AN=97002279&site=eds-live&scope=site

41. Tribbey C. Dolby Atmos Expert Speaks Volumes About Startling Audio Technology. *Broadcasting & Cable*. 2016;146(24):19. Accessed March 1, 2022. https://searchebscohostcom.libproxy.txstate.edu/login.aspx?direct=true&db=ofm&AN=116401480&site=eds-live&scope=site

42. Wenzel E, Begault D, Godfroy-Cooper M. Perception of Spatial Sound. In Rogniska A, Geluso eds. *Immersive Sound: The Art and Science of Binaural and Multi-Channel Audio.* Routledge; 2018:1-34. Accessed August 30, 2021.

43. Woźniak T. BinAural VST.; 2016. https://github.com/twoz/binaural-vst

44. Yu G, Wu R, Liu Y, Xie B. Near-field head-related transfer-function measurement and database of human subjects. *J Acoust Soc Am*. 2018;143(3): EL194.