AN CONTEXT-AWARE INTELLIGENT SYSTEM TO AUTOMATE THE CONVERSION OF 2D AUDIO TO 3D AUDIO USING SIGNAL PROCESSING AND MACHINE LEARNING

Bolin Gao[1] and Yu Sun[2]

[1]Fairmont Preparatory Academy, 2200 W Sequoia Ave, Anaheim, CA 92801
[2]California State Polytechnic University,
Pomona, CA, 91768, Irvine, CA 92620

ABSTRACT

As virtual reality technologies emerge, the ability to create immersive experiences visually drastically improved [1]. However, in order to accompany the visual immersion, audio must also become more immersive [2]. This is where 3D audio comes in. 3D audio allows for the simulation of sounds from specific directions, allowing a more realistic feeling [3]. At the present moment, there lacks sufficient tools for users to design immersive audio experiences that fully exploit the abilities of 3D audio.

This paper proposes and implements the following systems [4]:

1. Automatic separation of stems from the incoming audio file, or letting the user upload the stems themselves
2. A simulated environment in which the separated stems will be automatically placed in
3. A user interface in order to manipulate the simulated positions of the separated stems.
We applied our application to a few selected audio files in order to conduct a qualitative evaluation of our approach. The results show that our approach was able to successfully separate the stems and simulate a dimensional sound effect.

KEYWORDS

3D Audio, signal processing, Head Related Transfer Functions.

1. INTRODUCTION

3D audio is a powerful set of tools that allows users to virtually place sound effects in a three dimensional space by manipulating sounds coming from stereo or surround sound sources using head-related transfer functions and reverberations [5]. It is capable of tricking the brain into thinking that sounds are coming from different locations, while, for example, only using the two speakers present in headphones.

3D audio has been rising in popularity due to its ability to create immersive audio experiences, particularly for virtual reality applications, which are also gradually rising in popularity [6]. It is capable of simulating sounds from different locations without there being an actual audio source present. However, current usages are limited to sound effects within video games, and there does not seem to be enough tools specifically aimed at creating these immersive audio experiences.

Existing tools for manipulating 3D audio seem to be clunky and hard to use. For example, Dear Reality’s dearVR exists as a plugin for digital audio workstations. While it is a powerful tool and contains many features, it is difficult to navigate, and the user interface is complicated and hard to use.

This paper seeks to address the ease-of-use issues of currently existing 3D audio manipulation technologies by automating the process of creating 3D audio experiences via pre-determined layouts, then granting the users a simple and intuitive interface for continued customization. By doing this, the process of creating immersive audio experiences using 3D audio will be greatly simplified, allowing more users to have access to its capacities and lower the learning curve.

Some of the existing techniques and systems for creating immersive 3D audio include tools like Dear Reality’s DearVR, which allows users to control 3D audio in their sound production projects like music production or post production in filming. However, these tools assume the users to have professional experience with audio engineering, which is inapplicable to a large number of users. Others, such as Points2Sound, attempt to convert mono audio signals to 3D audio by using a deep learning model to interpret 3D visual information, then calculating the audio signals based on the predicted positions [7]. While their techniques are much easier to bring to users than Dear Reality’s tools, their implementations are also limited in scale because of the requirement of existing 3D visual information, which might not always be available in cases where users only have auditory information.

In this paper, we follow the same line of research as others before us that created 3D audio, but with a focus on usability. Our goal is to facilitate the conversion of mono audio to 3D binaural audio using Head Related Transfer Functions in a convenient and effective fashion. Our method is inspired by previously existing tools that implement 3D audio, like Points2Sound and DearVR and their ability to allow users to create more immersive auditory experiences. These are the main features of our system: first, the ability to perform source separation on the input mono audio file into different stems. Second, the automatic conversion of the separated mono audio signals into 3D audio based on preset positional data. Third, a comprehensive and easy-to-use tool for manipulating the positional data of the separated input audio files. We believe that these parts would constitute an efficient system that could make the immersiveness offered by 3D audio more accessible, and potentially bring it to a wider audience.

In our application scenarios, we demonstrate how the above combination of techniques would work in practice. First, we show the usefulness of our approach by presenting some practice examples, which include different audio segments, and the subsequent results after being processed by our system. Afterwards, we then obtain samples of the processed, 3D audio then compare the different presets against each other to analyze the effectiveness of our system to create the desired 3D audio effects. By comparing the different 3D audio signals generated by the system, we should be able to determine the difference between them and use them to discern the actual direction for where the sound is coming from. If we are able to discern it, then our system is effective.

The rest of the paper is organized as follows: Section 2 gives the details on the challenges that we met during the experiment and designing the sample; Section 3 focuses on the details of our solutions corresponding to the challenges that we mentioned in Section 2; Section 4 presents the relevant details about the experiment we did, following by presenting the related work in Section 5. Finally, Section 6 gives the conclusion remarks, as well as pointing out the future work of this project.

2. Challenges

In order to build the project, a few challenges have been identified as follows.

2.1. Requirement of individual audio stems

In order to create a spatialized sound effect, multiple stems of the audio would be needed. Else it would sound like all the inputted audio is coming from a singular direction. By having multiple audio stems, the program could simulate the effect of sounds coming from a multitude of directions. One option would be to ask for individual audio stem inputs in the function, however, they might not always be available, so we thought about a different approach: source separation. By using stem source separation technology, the singular audio source that users input can be separated into different stems automatically, allowing us to then apply the 3D audio sound effects to the different stems and place them into different virtual locations and create the desired immersion with headphones. For this project, we have opted to use Spleeter by Deezer, an open source source separation library written in python with state-of-the-art performance [8].

2.2. Application of 3D audio effects

With the separated audio stems, we then need a method to create the 3D sound effects on these stems. In order to solve this problem, we decided to utilize head-related transfer functions (HRTFs), which tells us how the left and right ears receive sounds from different locations differently [9]. When we apply these HRTFs to the audio stems and play them back through stereo headphones, sounds from specific locations and directions can be mimicked. By applying different HRTFs to different audio stems and merging them together, we can then create the effect of different sounds coming from different directions. We have decided to use the LISTEN HRTF database, measured at Ircam and AKG. It contains HRTFs measured from every direction surrounding the subject plus different elevations that would be useful for adding dimension to our 3D audio project. There are also a large number of subjects to choose from, which is more than enough for our purpose.

2.3. The customization of the output signal

After separating the audio stems, our system then applies 3D audio effects by using predetermined prefabs that put specific audio in specific locations. However, this does not exploit the full potential of 3D audio to create immersive sound designs. Our proposal to solve this problem is to add further functionality by allowing users to customize their input audio signal's 3D position with an interactive UI element [15]. This will allow the users to dictate where they want the audio source to sound like they come from, allowing them to customize the 3D audio experience. Also, we will allow the users to add multiple input signals. This allows for the users to have more choice while uploading their audio file. For example, users can upload individual audio stems for better and more customizable positioning than what is offered by Spleeter, which only allows separation up to 5 predetermined stems, or they can merge completely different audio files together for an immersive experience.

3. Solution

Our system allows for the creation of 3D audio from mono audio signals. First, the user will input a mono or stereo audio file. Then we separate the audio into different stems, for example, the vocal, piano, drums. We will also allow the user to input their own stems for further customization, as the automatic separated stems only allow up to 5 different stems. By letting the user upload their own stems allow more stems to be added for more flexibility. Afterwards, the system then calculates the new, specialized positions using the HRTFs and separated audio files. Now that the spatialized audio is created, we then implement a graphical user interface to allow for further customization of the 3D audio signal [13]. In order to achieve this goal, our system consists of the following components:

- A source separation algorithm for the separation of the input signal into respective stems
- A series of computations to automatically create the new specialized, 3D audio using HRTFs and the separated stems
- An interactive graphical interface to allow for further customization of the resulting 3D audio

The following diagram illustrates the components of our system.

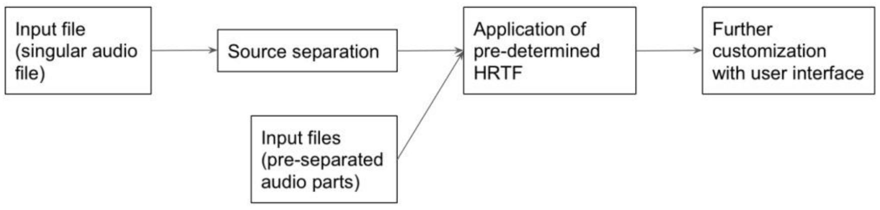
Input file (singular audio file)
Source separation
Application of pre-determined HRTF
Further customization with user interface
Input files (pre-separated audio parts)

Figure 1. Components of our system

In the next section, we will example and examine how each component works in depth.

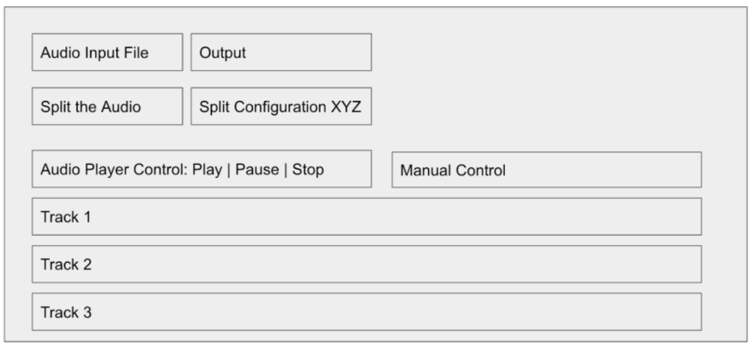

Figure 2. How components work

In this section we will explain how each component of the system works, starting with the source separation algorithm.

1. Source separation algorithm

The goal of the source separation algorithm is to splice the original, singular audio file into different parts so different HRTFs can be applied to them in order to create the effect that sounds are coming from different locations, therefore creating the 3D audio effect. In order to accomplish the source separation, we will use a pre-trained algorithm called Spleeter, made by Deezer. This algorithm is capable of 2-stem, 4-stem, and 5-stem source separation. It uses the original input file and uses pre-trained algorithms to extract parts such as the vocals, drums, and piano. With this feature, we can now apply directional information to the separated stems.

2. Application of directional information to the audio stems

Now that we have the audio stems, we need a way to create 3D audio effects by applying directional information to each of the stems to make them sound like they are coming from different directions, simulating the effects of being present in a 3D space and hearing sounds from different locations. To accomplish this effect, we will be applying HRTFs to the different audio stems. HRTFs are pre-recorded from different directions that show how the two ears react to them differently. By applying these HRTFs to our audio stems, we can calculate how the audio stems would sound like if they are from these directions. For our system, we will be using the LISTEN HRTF database, which contains HRTFs for all 360 degrees from the listener, along with different elevations above and below the listener for a larger number of options. We will automatically apply the HRTFs of specific directions to the audio stems to automatically create the 3D audio effects, then allow the user to further customize the resulting 3D audio signal with an interactive user interface.

3. Graphical user interface to allow for further signal customization

While we already successfully generated a 3D signal automatically, we have yet to access the full potential of 3D audio. In order to do that, we will allow the user to further customize their 3D audio using a graphical user interface. In the graphical user interface, the user can determine the simulated location they want each of their audio stems to come from. We will also add the ability for users to upload their own audio stems and directly apply the 3D audio to them. This allows users to directly apply positional data to their own stem, allowing for possibilities beyond the 2 stems, 4 stems, 5 stem separations offered by Spleeter.

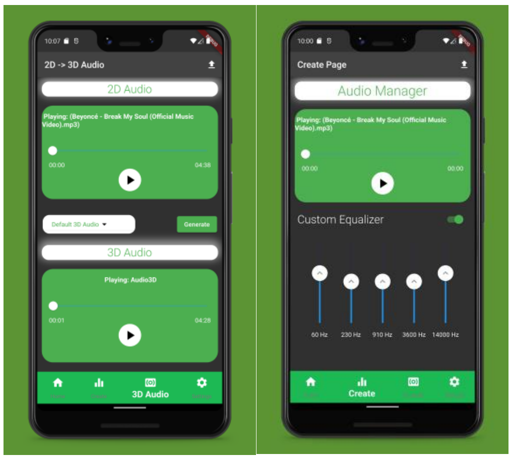

Figure 3. GUI for customizing 3D audio

4. Experiment

4.1. Experiment 1

To test the effectiveness of our system, we have decided to test it on two different real use case situations in the following procedure:

1. Grab the audio file
2. Use mat lab to plot the functions
3. Apply the HRTF to it
4. Plot it again and compare their differences
5. Add the audio files too for listening purposes

One case where we use Spleeter and one case where we just upload the parts.

Test a few different angles of the thing.

In order to test the effectiveness of our system, we decided to conduct an experiment to see how the system functions under a practical situation.

Here is the original audio file we are testing, it is composed of a drum track, a trumpet track, pianos, and a variety of other accompaniment in the background:

Embed the wav file here

Here is the Time Domain visualization of the input signal

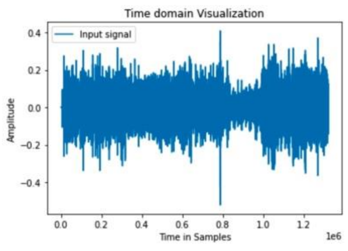

Figure 4. Time Domain visualization

And here is the Frequency Domain visualization of the input signal

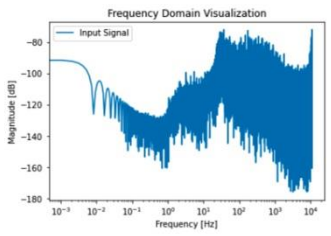

Figure 5. Frequency Domain visualization

For the first step of our system, we use the source separation algorithm in order to split the singular input file into multiple stems. We will utilize the 5 stem option in Spleeter from split the audio into 5 distinctive stems. Spleeter is capable of separating the track into the vocals, piano, drums, and bass. It will put the rest of the audio that does not fall into the previous sections into one signal called “other”.

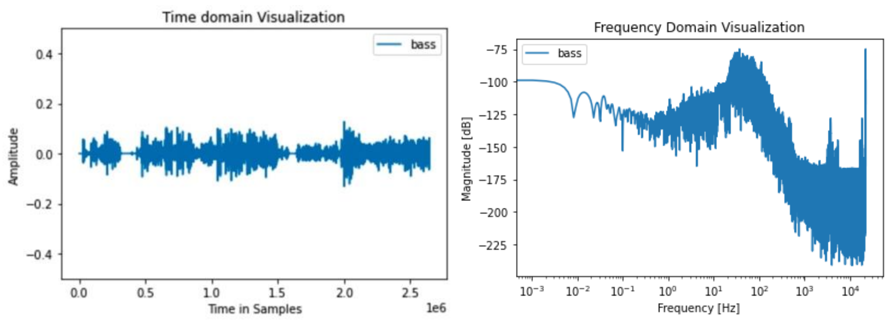

Figure 6. Time and frequency domain for bass

For the bass track, we can see that the amplitude is lower than the amplitude of the original audio signal. This means that the source separation algorithm was able to successfully separate parts of the original out of it. We can also see that this section does not contain much frequency of the higher ranges, notably from 100 to 1000 Hz. At the same time, the magnitude of the lower frequencies seem to be high, indicating that it successfully captured the bass of the original mix. There seems to be a peak at the very high frequencies, though. This is assumed to be an artifact from the source separation algorithm.

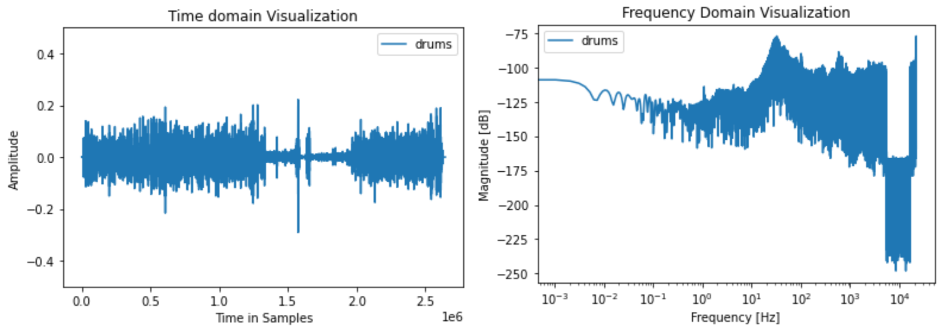

Figure 7. Time and frequency domain analysis for drums

The drum occupied a large amount of the original mix, so the amplitude seems to be higher than the amplitude on the bass most of the time. The frequency domain contains a large gap around 1000 Hz. We do not know the cause of this, and it seems like another artifact of the source separation algorithm. The drums contain a relatively large amount of magnitude at the very high frequencies after the gap, though. This is interpreted as the very high frequencies present in the original audio signal, as the high-hats are capable of generating these frequencies, unlike anything in the bass.

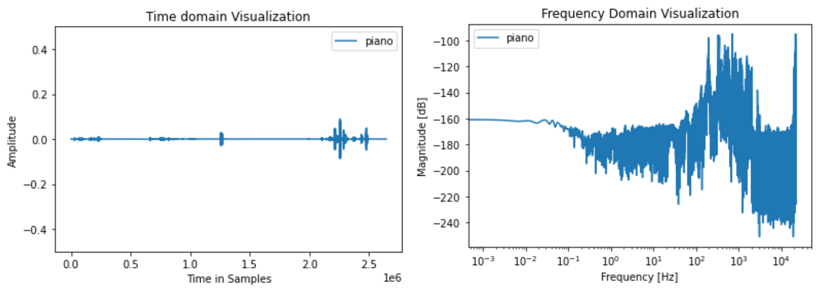

Figure 8. Time and frequency domain analysis for piano

The piano stem does not contain much of the original mix. The stem separation algorithm was not able to separate much of the piano out of the mix.

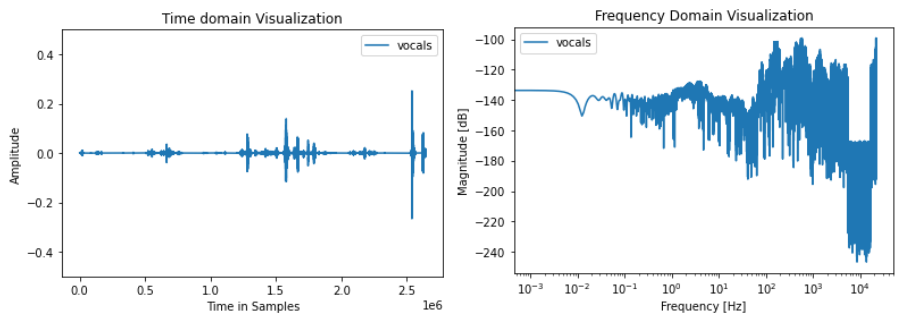

Figure 9. Time and frequency domain analysis for vocal

There should not be anything in the vocal stem, as the input audio does not have vocals. This is evident in the low amplitudes throughout the song. The signals that do exist here are artifacts of the source separation algorithm.

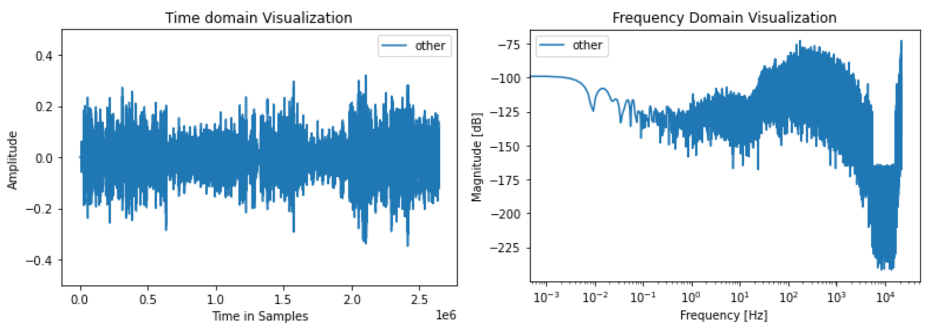

Figure 10. Time and frequency domain analysis for other instruments

Most of the instruments of the original audio falls outside of the category, so this stem contains much of the original audio. This is evident, as the amplitude of this stem is the highest. This distinctive gap at around 1000 Hz is still present.

Next, we will apply the HRTFs to the separated audio stems. In this example we will select four random HRTFs to the audio stems.

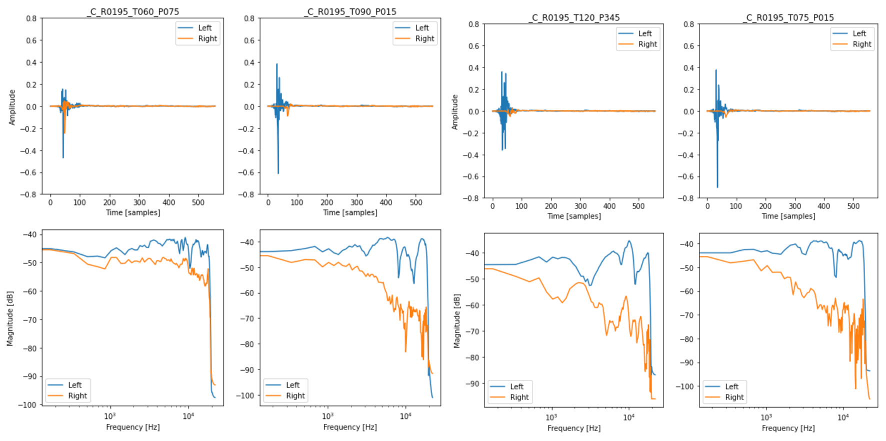

Figure 11. Time and frequency domain of the four HRTFs that are selected at random

We can see that all of these four amplitudes happen to be on the left, so we would expect more audio to be on the left side of the final mix.

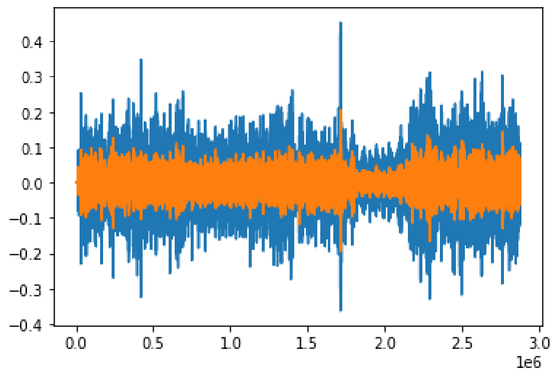

Figure 12. Amplitude of the left and right channels for the final mix

And that is confirmed by this graph showing the final amplitude.

6. Related Work

In this paper, the authors demonstrated a model that generates a binaural version of 2D audio based on 3D visual information from a scene [7]. This is comparable to our work, as we are both attempting to convert between 2D and 3D audio. However, their approach relies on visual information to determine how the generated binaural audio sounds. In our approach, we decided that rather than relying on visual cues, allowing direct customization of the output 3D signal with an interactive interface would allow for more freedom and generate more satisfactory results.

This paper presented a synthesis process in order to generate binaural audio from a mono audio source, also similar to what we are trying to accomplish [10]. Instead of utilizing head-related transfer functions, they created a novel process for generating binaural audio utilizing diffusion models. Their system works very well compared to the HRTFs that we have used. As their system does not contain a system for further customization of the resulting signal. It would be nice if the two approaches can be connected together.

This paper showed a method for generating binaural audio with a deep neural network instead of HRTFs [11]. They also included a system to extract positional information from images, and with the model, they were also able to estimate a depth map to aid the generation of the final binaural signal, which is something we have not implemented in our system. We would like to implement the depth aspect of their model in the future. With this, the interface for customizing the final binaural signal could be further improved upon.

7. CONCLUSIONS

In this paper, we have proposed a system that enables the conversion of 2D to 3D audio. The system consists of the following components: the separation of the input audio file, the application of HRTFs to the audio file, and an interface for further customization of the output 3D audio file [12]. For the separation of the input file, we have utilized Splitter, a source separation algorithm that's capable of 2, 4, and 5 stem separation. After separating the input audio into stems, we then apply different HRTFs to each of the stems. For this, we will utilize the Listen HRTF database. After applying the predetermined configurations to the audio stems, we then allow the users to further customize their output file with a graphical interface that allows users to apply different HRTFs to each stem.

To test the effectiveness of our system, we have applied it to different audio files to determine its performance. First, we analyzed whether or not the source separation algorithm was able to successfully accomplish its goal. Then we applied different HRTFs to the audio sources and analyzed its effectiveness. We then analyzed the time and frequency domains of the files at different stages. Based on the results of our experiments, we were able to see that the source separation algorithm and the application of HRTF has worked. And with this, we have concluded that our system is effective and was able to accomplish our goals.

For our current method, there exists a range of limitations that we have not yet been able to address. First, the source separation algorithm is only capable of performing source separation for a few pre-determined types of stems, such as the vocals, drums, and bass. Within these confinements, it is able to perform quite well, however, more options for separation will allow for more possibilities of customization. Another limitation of the current method is that it is unable to simulate the distance from the virtual audio source. If the distance is able to be simulated, then the virtual environment of the 3D audio would be complete. We would also like to add a way to dynamically apply HRTFs to the audio stems to make them appear to be moving through a 3D environment [14].

For the source separation algorithm, we hope to improve on it by further training the algorithm to obtain better results, and extend the capabilities of it to allow for further separation of different components of the input audio file. For the distance from the virtual audio source problem, we can manipulate the audio to approximate the distance. We would also like to have a method to make the audio files to appear as if they are moving through space.

REFERENCES

[1] Boas, Y. A. G. V. "Overview of virtual reality technologies." Interactive Multimedia Conference. Vol. 2013. 2013.

[2] Cummings, James J., and Jeremy N. Bailenson. "How immersive is enough? A meta-analysis of the effect of immersive technology on user presence." Media psychology 19.2 (2016): 272-309.

[3] Brinkman, Willem-Paul, Allart RD Hoekstra, and René van EGMOND. "The effect of 3D audio and other audio techniques on virtual reality experience." Annual Review of Cybertherapy and Telemedicine 2015 (2015): 44-48.

[4] Sundareswaran, Venkataraman, et al. "3D audio augmented reality: implementation and experiments." The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, 2003. Proceedings.. IEEE, 2003.

[5] Frauenberger, Christopher, and Markus Noistering. "3D audio interfaces for the blind." Georgia Institute of Technology, 2003.

[6] Sherman, William R., and Alan B. Craig. "Understanding virtual reality." San Francisco, CA: Morgan Kauffman (2003).

[7] Lluís, Francesc, Vasileios Chatziioannou, and Alex Hofmann. "Points2Sound: From mono to binaural audio using 3D point cloud scenes." arXiv preprint arXiv:2104.12462 (2021).

[8] Hennequin, Romain, et al. "Spleeter: a fast and efficient music source separation tool with pre-trained models." Journal of Open Source Software 5.50 (2020): 2154.

[9] Brown, C. Phillip, and Richard O. Duda. "An efficient HRTF model for 3-D sound." Proceedings of 1997 Workshop on Applications of Signal Processing to Audio and Acoustics. IEEE, 1997.

[10] Leng, Yichong, et al. "BinauralGrad: A Two-Stage Conditional Diffusion Probabilistic Model for Binaural Audio Synthesis." arXiv preprint arXiv:2205.14807 (2022).

[11] Parida, Kranti Kumar, Siddharth Srivastava, and Gaurav Sharma. "Beyond mono to binaural: Generating binaural audio from mono audio with depth and cross modal attention." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2022.

[12] Hornstein, Jonas, et al. "Sound localization for humanoid robots-building audio-motor maps based on the HRTF." 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006.

[13] Guizzo, Eric, et al. "L3DAS21 Challenge: Machine learning for 3D audio signal processing." 2021 IEEE 31st International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, 2021.

[14] Azim, Asma, and Olivier Aycard. "Detection, classification and tracking of moving objects in a 3D environment." 2012 IEEE Intelligent Vehicles Symposium. IEEE, 2012.

[15] Sermuga Pandian, Vinoth Pandian, Sarah Suleri, and Prof Dr Matthias Jarke. "UISketch: a large-scale dataset of UI element sketches." Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems. 2021.