The 3D sound rendering system

Qiguang Chu, #300042722, Liyuan Cao, #300042723

Abstract— **The concept of 3D sound rendering system is a sound modification program that can deal with mono or stereo audios and reproduce make them spatial surrounded. Horizontal rendering and vertical rendering are explicitly introduced in this article. We apply head-related transfer functions as filters to separately construct spatial and frequency domain audio systems. Besides that, designed HRIR function are implemented to achieve the same sound effect.**

Index Terms— **Sound Rendering System, Time Domain, Frequency Domain, Convolution, Designed Filter**

I. Introduction

The 3D sound is always surrounded in people's real life. Sounds arrive at our ears from every direction and from varying distances. These and other factors contribute to the three-dimensional aural image humans hear. [1] The goal of this project is to convert traditional mono and stereo audios to spatial surrounded sound.

The 3D spatial sound rendering system we made conceptually comprises two parts. For the first part, it tackles audios on the spatial domain and frequency domain individually and results in a surround sound effect audio with horizontal or vertical spin. For the second part, we implement the experimental measured spherical head model to do a spatial domain design for the half-spherical stereo audio, making it as much similar as the computation. In this experiment, Ringtone of Mario is the input audio with length of 20 seconds, and the output is a component wave file with the same length.

II. HRIR Filter

Our HRIR filter used for this experiment comes from the experimentally measured public domain sets of HRTFs from the web. We choose CIPIC HRTF Database as out HRIR filter and HRTF function (https://www.ece.ucdavis.edu/cipic/spatial-sound/ hrtf-data/). This HRTF filter results from a computed and concluded formulation provided by University of California. But as we all know, HRTFs vary significantly from person to person. It seriously depends on human modal and if the human modal does not fit, it will be seriously perceptual distortions. [2] We choose one of the CIPIC HRIR subjects as our database after testing. The HRTF filter in the frequency domain is noted as H[k], and the time domain HRIR is noted as h[n].

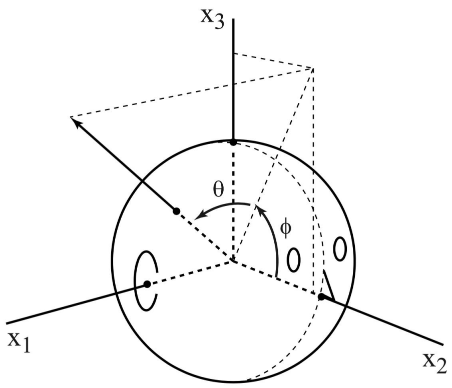


Fig. 1. HRIR Filter Model

The CIPIC HRTF Database is a public-domain database contains 45 different subjects based on high-spatial-resolution HRTF measurements. [2] The CIPIC HRTF database is a Matlab format file with three-dimension size of $2 \times 25 \times 50 \times 200$, respectively stands for 2 channels, 25 azimuth angles, 50 elevation angles and 200 samples for each. The HRIR spatial head-related model is shown as Fig.1. The head is presented as a sphere model and three axles express three orthogonal vectors x_1,x_2,x_3, correspondent to right, front and overhead. The azimuth angle θ and the elevation angle ϕ are measured in the head-centered coordinator. The azimuth varies from $-90°$ to $90°$, starting from the left ear to the front of the head and then to the right ear. Twenty five angles are extracted from $180°$ to construct a direction space. Relatively, the elevation angle is from $-90°$ to $270°$, starting from directly below to the front of the head, to overhead, to back of the head and to the below. Fifty angles are equally distributed to this whole circle. For every slice of elevation angle, the track of sound is a half circle moving from the left ear to the right ear. What we want to do is constructing designed sound tracks based on the existing track to achieve two routines. The one is moving from the center front counterclockwise to left, behind, right and front. The other way starts from the left ear, overhead and to the right.

In order to simplify the following operations, both left and right two channels are alternately interspersed into a new list, which is shown in Fig. 2.

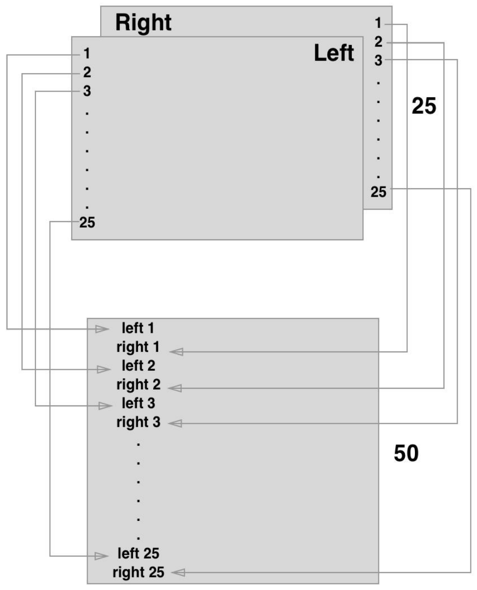


Fig. 2. HRIR Filter

A. The horizontal surrounding sound database

For the horizontal rounding part, we choose the ninth (frontal HRIR filter, hrir[:,8,:] in Python) and forty-first(back HRIR filter, hrir[:,40,:]) slices to filtrate the audio, which represents each half circle of a whole round. The requirement is to make sound moving from the front of the head to the left of the head, to the back of the head, to the right of the head and to the front of the head. So, some modifications have to be applied to these two parts in order to achieve the specified surround effect of the entire head. The frontal HRIR filter is split into two equal sizes pieces and appends to the head and the tail of reversed back HRIR filter separately. The process of implementation is shown in Fig. 3.

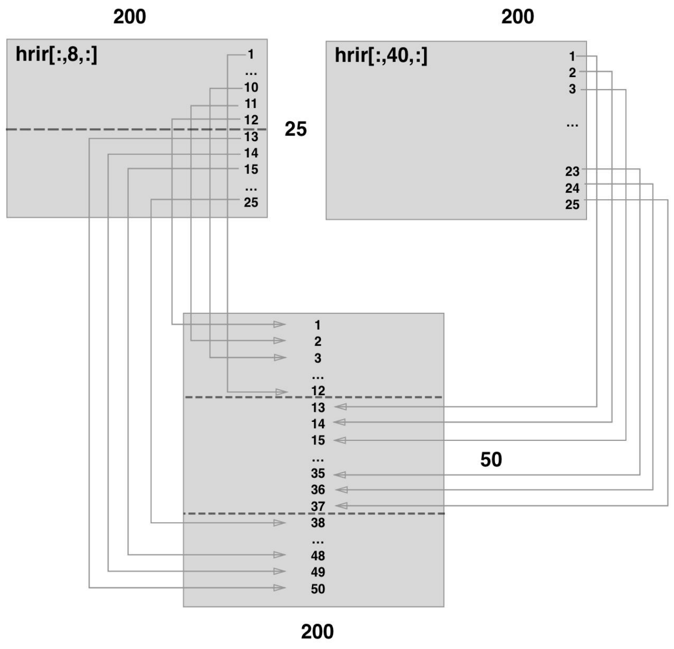


Fig. 3. Horizontal database

B. The vertical surrounding sound database

The twenty-fifth slice(hrir[:,24,:]) is for the elevation rotating. We just need to use this part as HRIR filter to achieve sound moving from the left ear to the top of the head, then to the right of the head. As usual, datasets of left ear and right ear are alternately interspersed into a new list.

C. Design HRIR filter

The formulation we retrieved from the essay (ref. [2]), the designed HRTF filter is obtained from two formulations [2]:

$$H_R(\omega,\theta)=\frac{1+j2\alpha\omega\tau}{1+j\omega\tau}e^{-j\omega T_R} \tag{2-1}$$

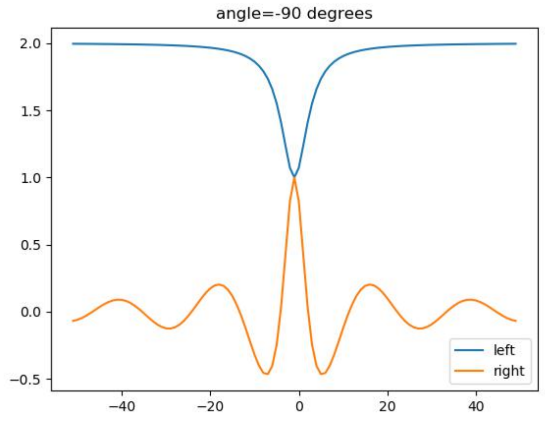
$$H_L(\omega,\theta)=\frac{1+j2(1-\alpha)\omega\tau}{1+j\omega\tau}e^{-j\omega T_L} \tag{2-2}$$

where $\alpha=\frac{1}{2}(1+sin\theta)$, $\tau=\frac{1}{2}(a/c)$, $T_R=(1-\alpha)\tau$, $T_L=\alpha\tau$. The a in these functions is the radius of the head and the c is the speed of sound.

As shown on (2-1) and (2-2), the designed HRIR filter is obtained by adjusting the input ω and θ. We take one sample for each 5° of $theta$ and totally 101 samples for the ω. When one listens to synthetic binaural sounds produced by this filter, the apparent location moves smoothly from the left ear to the right ear as θ is varied from $-90°$ to $90°$. However, this model does not provide any elevation dependence, and the apparent location is not externalized but appears to be inside the head. [2] N is the total number of samples, which equals 101.

$$-90°\le\theta\le 90°$$
$$-\pi Fs\le\Omega\le\pi Fs$$
$$-N/2\le K\le (N-1)/2$$

We take photos at $-90°$, $0°$, $90°$ as examples, which represent the sound on the center-left, center front and center-right. As we take 101 samples from the filter, the plots are like this:



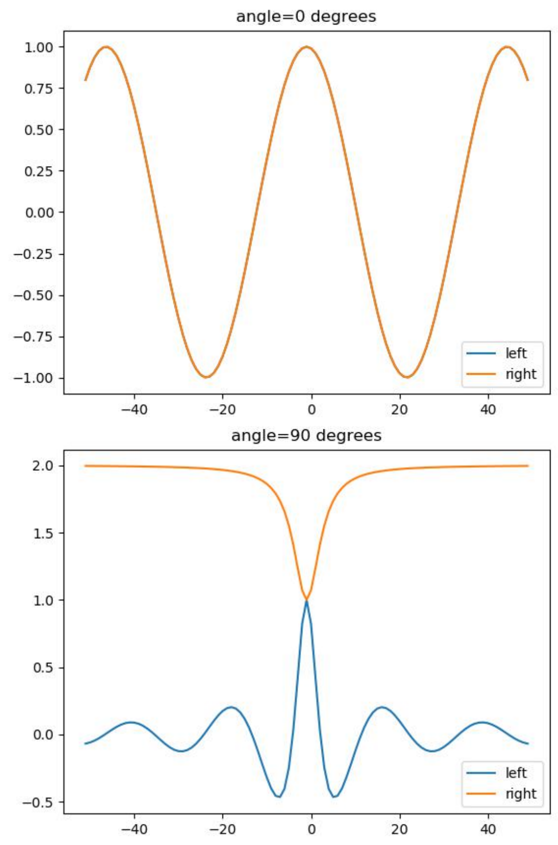
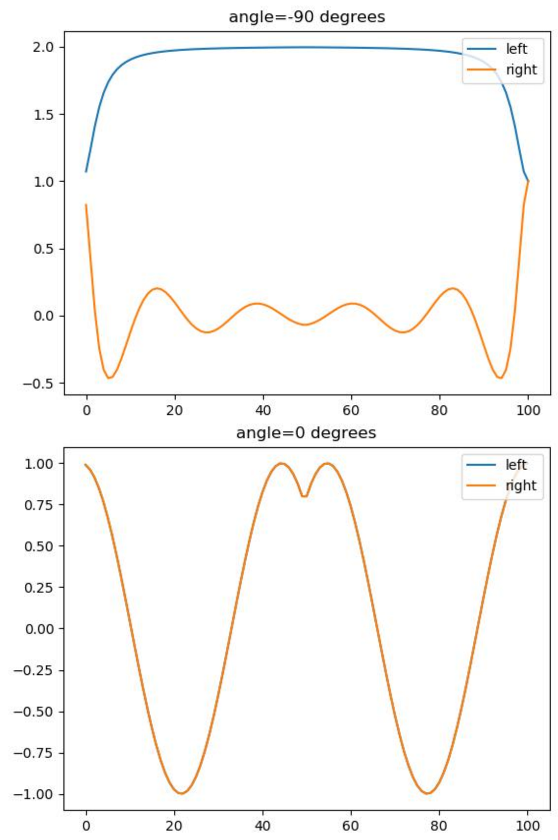


Fig. 4. The samples of the plot of H[K] before the first shift



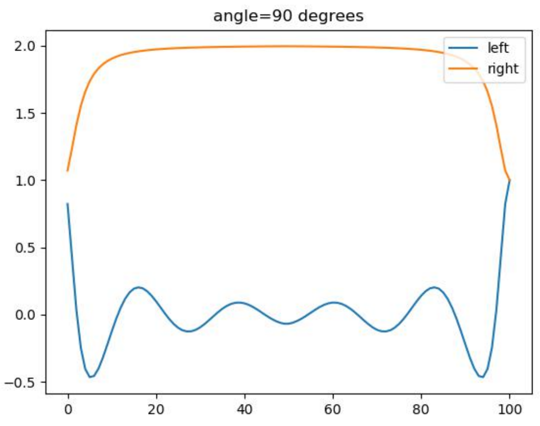


Fig. 5. The samples of the plot of H[K] after the first shift

After that, we make a shift and do Inverse Fast Fourier Transform and plot like this:

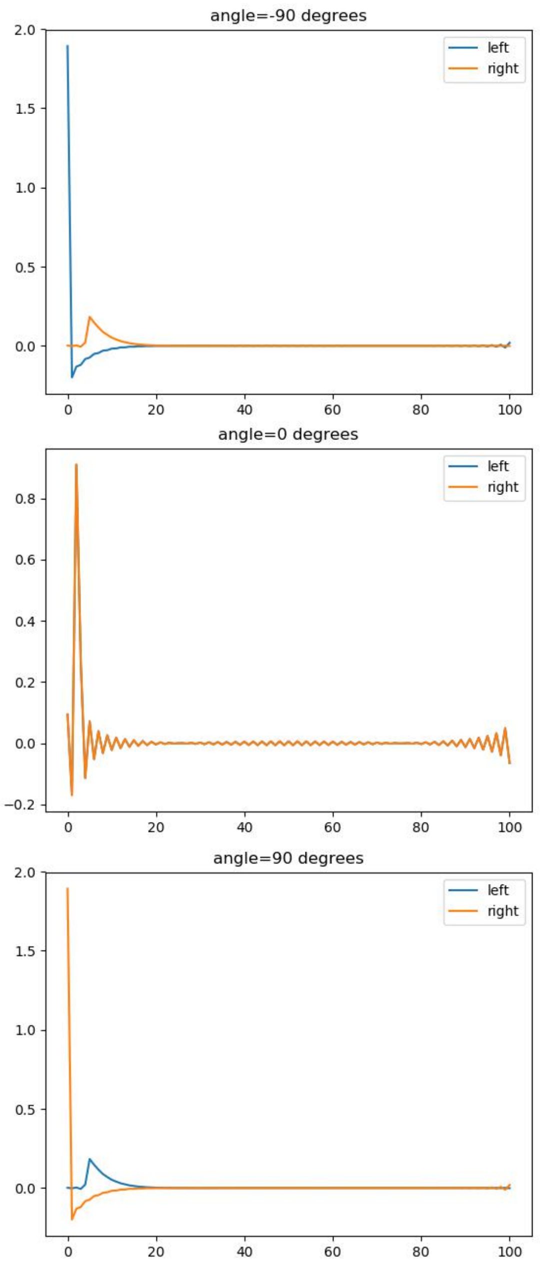


Fig. 6. The samples of the plot of h[n] before the second shift

In order to get a full period of the filter, we do a shift again to get a complete period on the right side:

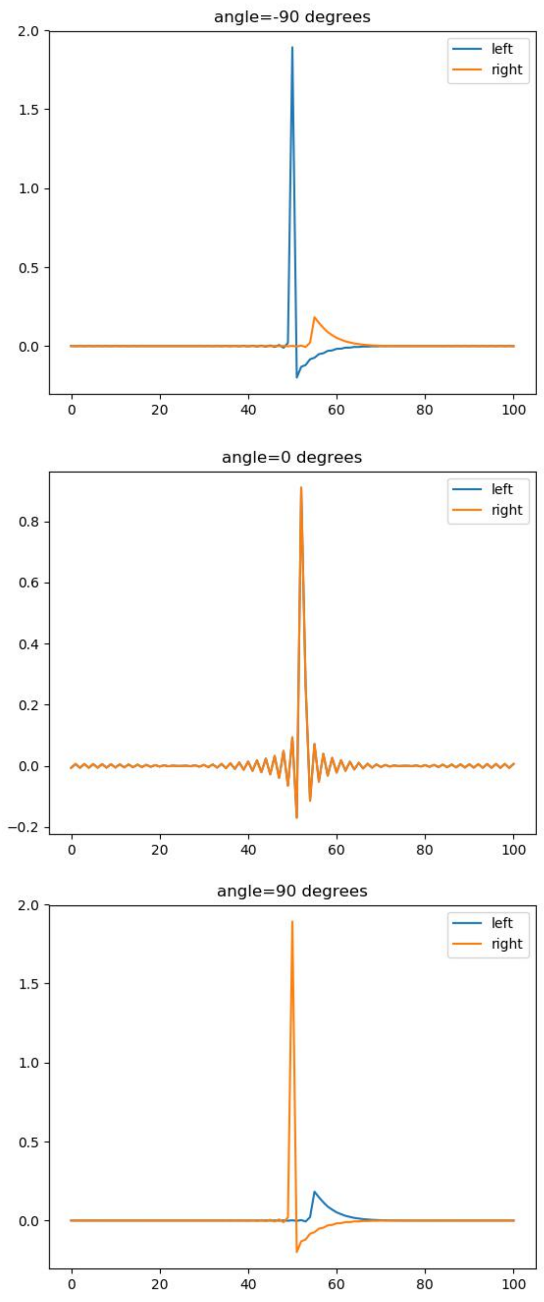


Fig. 7. The samples of the plot of h[n] after the second shift

After all these operations, the rest calculation is the same as the time domain.

III. Signal

The operation of digital signal data consists of three operations: signal loading, signal processing and signal writing. The odd rows of the processed HRIR filter represent the filters of the left channel, the even rows represent the filters of the right channel, and each row reflects an angle. In order to achieve horizontal surround or vertical surround sound affection, the first thing to do is to load the original audio signal and its properties correctly. Next, we perform the time domain convolution or frequency domain product of the original signal and the processed HRIR filter according to the assigned requirements, so as to obtain the processed signal. Finally, the processed signal is written as a result of the filtering into an audio file as a result.

A. Signal Loading

Loading is the first operation. We select Python's first party package WAVE to load the audio. Properties like the number of audio channels (nchannels), sample width (sampwidth), the sampling ratio (framerate), number of audio frames (nframes), compression type (comptype) and compression name (compname) are set properly. In this project, the sample rate of the audio is 44100Hz and 2 channels with 8 bits. The signal read by WAVE is a list of hexadecimal digits, which is not easy to deal with directly. Format conversion is a good option. In order to get a processable decimal signal, we use 'sfile.readframes()' to read the signal and ignore the array type to extract only the signal. Finally, we get an int32 type origin signal whose length is 917785. The maximum value is 11261 and the minimum value is -10996. This signal is noted as $x[n]$.

B. Signal Processing

1) Time domain

For time-domain operations, as for the signal, it has to be modified to suit the format of the HRIR list. The whole signal is cut into an arbitrary number of equal length segments depending on the vertical length of the HRIR array. For each segment, it corresponds to a specific angle and has to be convolved with the corresponding filter. We note segments with their number in the sequence of the signal as $x_1[n]$, $x_2[n]$, $x_3[n]$,...,$x_m[n]$, m is the number of segments.

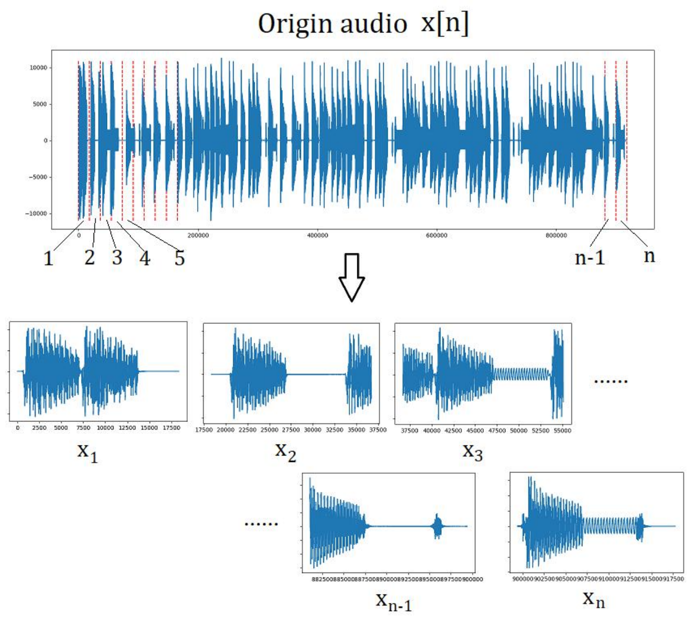


Fig. 8. Signal segmentation

Each part of audio shall convolve with relative part HRIR filter for both left and right channels individually. Each signal segment $x_i[n]$ convolve with correspondent $h_{2i-1}[n]$ and we have one short measure expressing sound comes from one arbitrary direction. The structure of the flow chart of time domain convolution process is shown as Fig.9.

$$g_{2i-1}[n] = \sum_{k=1}^{K} x_i[k] \times h_{2i-1}[n-k]$$
$$g_{2i}[n] = \sum_{k=1}^{K} x_i[k] \times h_{2i}[n-k]$$
$$i = 1,2,\ldots,m \qquad (3\text{-}1)$$

Filtered signals are kind of parts of stereo audio. After that, segments of the filtered signal forms a matrix, which has two times vertical length of the original audio.

In order to correctly output frames, segments of two-channel frames are alternately interspersed into a new array. Only the real parts of complex numbers are available for the wave file and we take them out for writing.

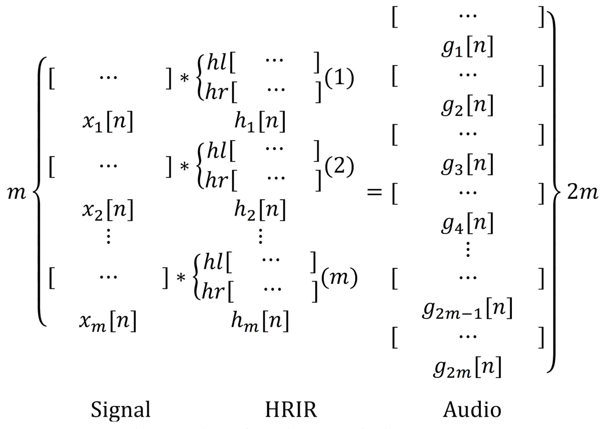


Fig. 9. Time domain convolution process

2) *Frequency domain*

In the frequency domain, the process is relatively complicated and the overlap-and-add method is applied there. The convolution in the time domain equals the multiplication in the frequency domain. The length of the two segments to be multiplied must be the same, which is not the same as convolution. The signal and HRTF filter have to be stretched by zero paddings to make it matched for each segmentation as preparation.

$$H_i[n] = FFT(h_i[n], len(h_i[n])), i = 1,2,\ldots,m \qquad (3\text{-}2)$$

For each segmentation, sliced signal and HRTF filter do Fast Fourier Transform individually with the same size L (length of a sliced signal). After that, an internal multiplication is implemented between these two frequency-domain datasets. Then, we do the inverse Fast Fourier Transform for the multiplication to get the output of one side.

$$g_{2i-1}[n] = IFFT(FFT(x_i[n]) \times FFT(hrir_{2i-1}[n]))$$
$$g_{2i}[n] = IFFT(FFT(x_i[n]) \times FFT(hrir_{2i}[n]))$$
$$i = 1,2,\ldots,m \qquad (3\text{-}3)$$

The segment after IFFT has a length of L+M-1. In order to connect these segments together while don't lose details of signals, we apply the overlap-add method to get the list of the compound signal. The tail of the former segment would be added to the head of the latter segment, with a size of M-1. k segments form the signal with kL+M.

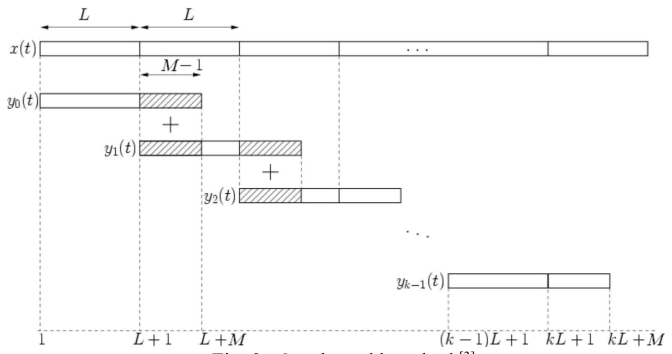


Fig. 9. Overlap-add method [3]

3) *Design part*

The procedure of designing part is as the same as the time domain convolution, except the self-designed HRTF function. We apply the designed model here to generate filter.

As we get from formulation (2-1) and formulation (2-2), the outputs are frequency domain values and we need to do inverse Fourier transform to make it available.

$$g_{Li}[n] = \sum_{k=1}^{K} x_i[k] \times IFFT(H_{Li}[n-k])$$
$$g_{Ri}[n] = \sum_{k=1}^{K} x_i[k] \times IFFT(H_{Ri}[n-k])$$
$$i = 1,2,\ldots, m \qquad (3\text{-}6)$$

The horizontal length of HRTF depends on the sampling density and the vertical length of that results from the sound affection we want to realize.

C. *Signal writing*

By convolving with the time domain or frequency domain of HRIR, the processed left and right channel signals are respectively obtained. The processed data type of the left and right channel signals is float64, so the left and right channel signals need to be converted to the int32 type before writing. In order to get the sound effect of the two-channel surround, the left and right channel signals are cross-written. Besides, the 'nchannels', which is one of the parameters of the output audio, is set to 2, which means this audio has two channels.

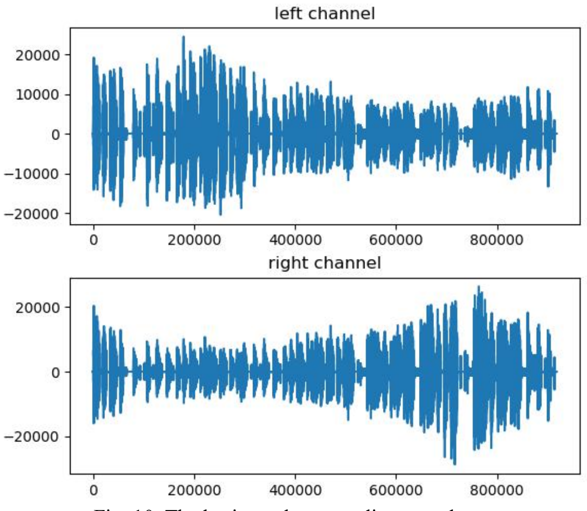


Fig. 10. The horizontal surrounding sound output

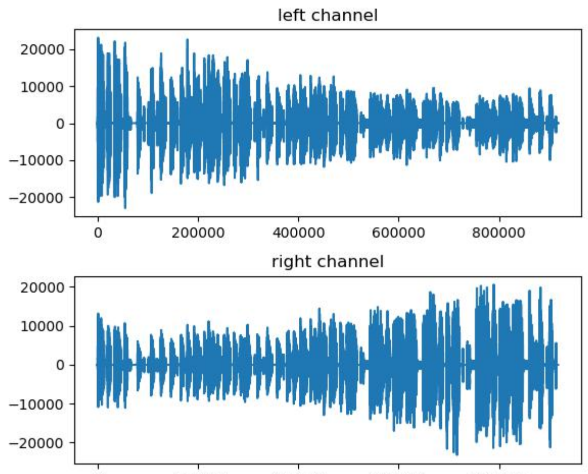


Fig. 11. The vertical surrounding sound output

IV. Problems and solutions

In this experiment, we faced some problems in programming and fixed them, harvesting experience.

When it comes to reading the audio, we applied two measures, first-party module WAVE and third-party module scipy.io. We found that when we read the file with scipy, the signal is decomposed as type int16 which we can use directly. Its shortcoming is that time costing. On the contrary, we have to transform the format when we use the wave command built inside numpy but it reads faster. As the work is not allowed to adopt the scipy package, we use the wave function here.

The procedure of the overlap-add method pretty counts. For example, the sound may be concussion around if tails of the former segment are not properly added to the head of the latter segments. This accident happened at first and we spent some time to find out why.

The convolution part is also important. As the calculation, convolution results are connected piece by piece and end to end. Two ends of each segment may have an edge effect, which means you will hear a series of slap sounds around. The solution is to set up the convolution function to avoid edge effect. Mode ‘valid’ returns output of length max (M, N) - min (M, N) + 1. The convolution product is only given for points where the signals overlap completely. Values outside the signal boundary have no effect. segments of signals are smoothly connected together and the audio sounds beautiful.

V. Conclusion

The 3D sound rendering system achieves converting flat audios into 3D spatial music. Both HRIR (time domain) and HRTF (Frequency domain) are applied to modify the audio. Experimental measured HRTF functions are used here to complete the same goal. Windowing and overlap-add methods are good for reducing edge effect. The work has great application on music software. Random music style can be achieved here by filters.

References

[1] Burgess; David A (1992). Techniques for Low Cost Spatial Audio. Proceedings of the 5th Annual ACM Symposium on User Interface Software and Technology. pp. 53–59.

[2] V. R. Algazi, R. O. Duda, D. M. Thompson and C. Avendano, “The CIPIC HRTF Database,” Proc. 2001 IEEE Workshop on Applications of Signal Processing to Audio and Electroacoustics, pp. 99-102, Mohonk Mountain House, New Paltz, NY, Oct. 21-24, 2001.

[3] R. O. Duda, "Modeling head related transfer functions", Proc. 27th Asilomar Conf. on Signal Systems and Computers, pp. 457-461, 1993.

[4] Rabiner, Lawrence R.; Gold, Bernard. Theory and application of digital signal processing. Englewood Cliffs, N.J.: Prentice-Hall. 1975: pp 65–67. ISBN 0-13-914101-4