# 2021 Audio Programming Assignment 3 Report

Qisong Ma, s2043828

## 1. General Idea

This plugin is for use of upmixing stereo audio stream into a 5.1 surround sound scheme. The main theoretical background that supports this plugin comes from papers from S. Kraft, and U. Zölzer1. It is a general multi-functional tool for any audio engineers who need an expedient way of converting stereo content into surround format. Though it is not feasible to build a plugin that's competing amongst professional industries works, the main aim of my project is to have a try at building such a kit using new development merging on the edge of technologies.

## 2. Algorithm Structure

The upmixer is mainly utilizing frequency domain analyze techniques, most of the signal processing takes place in the frequency domain. So, for input signal, it will need a handler for overlapping between audio blocks and applying window function before FFT. It's such a vital point that the omission of such a proper handle will result in serious noise for frequency-domain signal processing. I am referencing Daniel Rudrich's[2] class for this role, from which my core processing class UpmixProcessor class inherited.

After FFT, the algorithm takes a low complexity analyzing model to extract the main or direct content from the noise or ambient content. This step is deduced based on the simplification of the making up of the dominant sources of a single frequency bin within a short time interval.

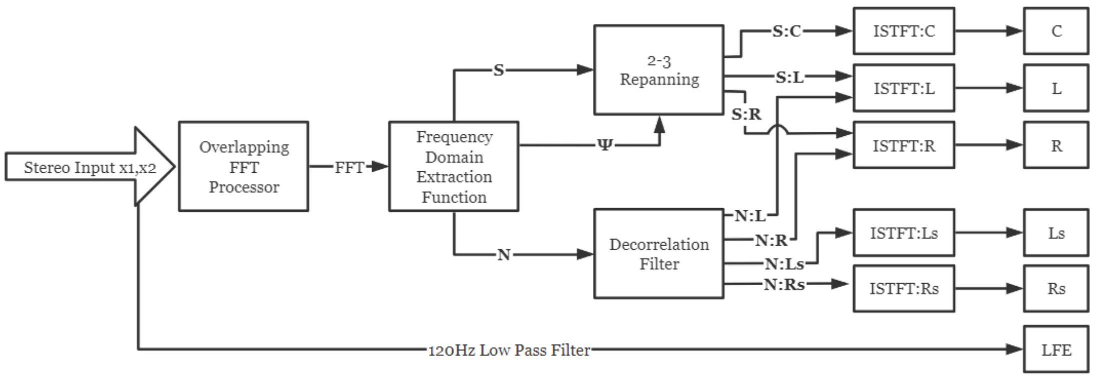


Figure 1 Block diagram of the upmix algorithm

[1] S. Kraft, and U. Zölzer, "Low-Complexity Stereo Signal Decomposition and Source Separation for Application in Stereo to 3D Upmixing," Paper 9586, (2016 May.).

[2] https://github.com/DanielRudrich/OverlappingFFTProcessor

When the main and ambient components are split, the former is sent to the 2 to 3 repanning algorithms, which will deduce the azimuth angle of the main content in each of the frequency bin's, then repanning them to fully make use of the added center speaker, based on vector based amplitude panning (VBAP) method. The latter will flow through 2 decorrelation filters, each split the signal into two unrelated parts, thus generates the four ambient signal each for one speakers in the corner.

## 3. Parameters

It is equipped with some basic while interesting parameters control for the users to grab the essence of the power of this tool. These include the separate adjustment of the gain of main, ambient content, this can even serve as an extraction of the reverb. There is also control of loudness on LFE channel. The decorrelation parameter controls the strength of decorrelation filters. When solo on ambient content, it is very noticeable how increase of decorrelation leads to the split and stretch of sound images surrounding the listener. While low decorrelation value might provide more robust downmix compatibility, high decorrelation is certainly providing a more natural and more immersive environmental sonic picture. The 'front' parameter controls the amount of higher frequency contents in ambient signal to lean over to the front channels. The ambient is mostly mid or low range signal, so it is not a very apparent controller.

I used PluginEditor to setup the user interface, and did a careful but simple design about the layout of the plugin, the positions of each sliders and their intervals are automatically specified according to total sliders numbers and window size. The sliders communicates with plugin parameters using AudioProcessorValueTreeState,

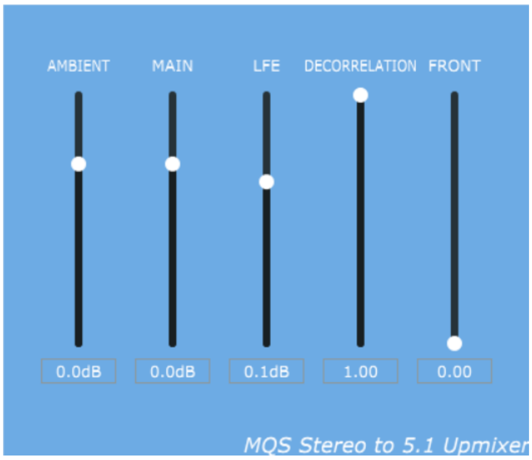


Figure 2 The UI of the Upmixer, contain 5 sliders

## 4. Programming Highlights

One difficulty in plugin programming in C++ is the complexity of debugging and testing during coding. It is especially true for building multi-channel plugins, since the testing require more than stereo plugins. I used downmixer for testing it on a headphone, simulating the setup of surround studio.

For coding aspects, it needs a bit of trial and error to fully fix all the problems occurred in this project. In fact, manipulating audio in frequency domain is like doing surgery using microscope, it might provides a clearer image, but the behavior is not so well defined.

The initialization of decorrelation filters is of an interesting part. It needs randomization along with slippery curves (otherwise immense irregular filtering in frequency domain will cause serious noise). So I set up series of sinusoidal signals thus each of it cycles with different wavelength and magnitude, while connecting all together.

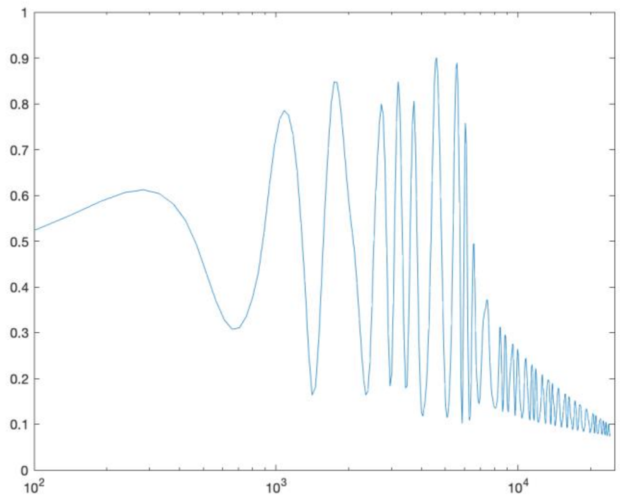


Figure 3 Decorrelation filter for rear channels when setting with high front value

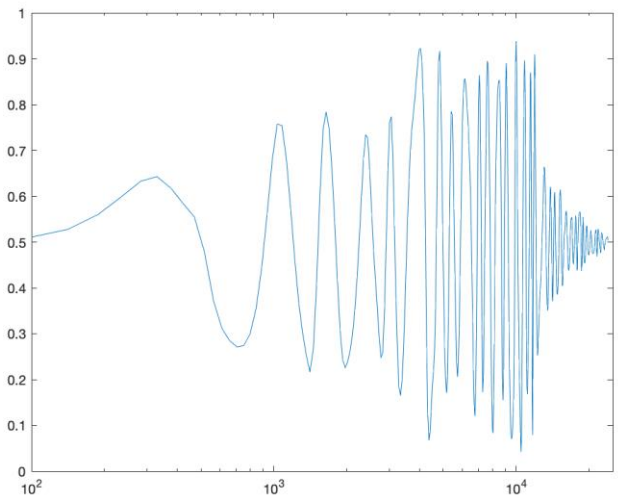


Figure 4 Decorrelation filter for left and right channel

## 5. Performance

Testing the plugin myself, I found the plugin works pretty well with pop music, creating immense sense of immersive feeling and enveloping. It is a less effective with classical music though. I think it is because of the way it extracts main components more rely on the difference of intensity rather than difference of time. Several stereo recording techniques such as the AB miking, on the other hands, are creating time delay rather than loudness gap. So, the algorithms might falsely classify some of the main contents into noise, thus narrowing the orchestra's image. Still, it extracts the ambience components and successfully construct the sentiment of the venue, and creating reverb around the listener.

Another drawback is that the computing power it required is relatively large. Since it using two FFT and five IFFT during each block. For future improvements, use of more light-weighted transform function or using frequency blocks chopped by filters instead of bins could be ideal.

Overall, it is a robust and effective tool, and has achieved the goal of the project. The flexibility it creates and enables audio engineers to looking at the stereo signal from the viewpoint of spatial relations really enhance my knowledge about music technology, and extend my imagination of the ability to manipulate audio.