CS 378 Programming For Performance: Project 2

Sai Avala EID: ssa766
Vaibhav Gupta EID: vg6663

October 13, 2014

1 Introduction

For this projct we were required to use an FIR filter in order to mix stereo audio. The stereo audio is mixed and expects speakers at -30 and 30 degrees relative to the listeners head. We had to use head-related transfer functions (hrtf) to take the stereo audio and render it for bianural playback through headphones.

2 FIR Filter

In order to take the stereo audio and render it for bianuaral playback through headphones, we used an FIR Filter. The left output channel is the sum of the left hrtf FIR filter for -30 degrees applied to the left channel and the left hrtf FIR filter for 30 degrees applied to the right channel. This is also applies to the output channel. We choose our coefficient file from the passed in arguement and then read the values into a map such that we could quickly access the coefficents for any desired environement (i.e. Angle -80, Elevation 0, 'R'). Then, we proceeded to read in the Left and Right mono audio files from the arguements. We applied the filter as per the design described above and merged the output streams into a stereo audio file.

The original function consisted of a nested for loop. For every "i", up til the size, and then for every j up till FIR_SIZE=199, y[i] = h[FIR_SIZE - j] * x[i + j - FIR_SIZE]. This was the standard loop. To speed up the FIR Filter function, I vectorized both the inner and outer loops. The vectorization of the inner loop allows for speed up in computing each value of y[i], and then the outer loop is vectorized so that I am computing y[i], y[i+1], y[i+2], and y[i+3] values at the same time. The main computation takes place in the inner loop. Take the arrays "h" and "x". Represent the y values we want to compute as columns, such that each element multiplication of x and y is columnar. We load in 8 values of "x", and shuffle them to match the right value of "h" to multiply with. Sum down the four columns, and that sum will be the values of y[i], y[i+1], y[i+2], and y[i+3] respectively.

With both the "l" and "r" outputs, we had to merge them into one stereo file. The vectorization of the "merge" function takes in both the "l" and "r" inputs and an "o" output array. We load four vales of "l" and "r" and shuffle them such that we get two arrays: l0 r0 l1 r1, and l2 r2 l3 r3. All eight of these values are then stored into the output, "o", array. During the process of converting from floatToInt, we vectorized the function such that we loaded four values from the input at each iteration. We used a mask to determine which indices were less than the minimum value (-32768.0) and which were greater than the maximum value (32767.0), and replaced the values which were out of the range with their respective clipped value. We were not able to vectorize the actual casting.

3 HRTF Demo

For this section of the assignment, we read the filter using the same approach as above, and then proceeded to read a mono audio file. We read in the input with a given angle as well as a sample *.pcm* file. The overall function that we use to apply the filter for the HRTF Demo is *apply_surround()*. This function applies a random angle to the FIR Filter at an interval of every 5 seconds. *apply_surround()*. Basically what happens is that we have an array of angles, we loop through and every 5 seconds choose a random angle from the array, and have the FIR Filter use that. We then write out the filtered audio into one stereo audio file.

4 Results

The results of our run times are below:

```
Absolute Times with Vectorization:
Part1: 2.300888 seconds
Part2: 1.082839 seconds
```

The data is the first 15 seconds of the 01.L.pcm and 01.R.pcm files provided the samples. The run time of Part2 makes sense because we are processing half the amount of data that is processed in Part1. The times that we provide do not count for IO.

5 Instructions

Please run the following commands to view our code and replicate our results. As the tar file does NOT include the sample data which was too large to add. If you want to get everything (including the sample data) please clone the repo.

```
git clone https://github.com/hellovai/cs378-update.git vg6663-programming
cd vg6663-programming/project2/
make
make test1
make test2
```