**Super Sound Regenerator (SSR)**

**Group member:** Junbin Ma

**Introduction:**

Sound effects on Human voice is very popular and a lot of companies makes great money on doing so. However, the “cost” of applying sound effects is quite lot for both the hardware and software, and usually after applying the effects on real time or audio files, the sounds will be "unnatural" because those software/hardware usually treat sounds as discrete data, which for sure, not the case for real people’s sound (and most sounds in audio files).

**The challenge here is:**

**Challenge branch 1:**

1: we cannot trade the quality of sounds for the efficiency, because otherwise the SSR products will be meaningless (or just act like the other products);

Approach -> Using FFT conversion to make the audio file from discrete data to continuously function/data, and it will almost lose no accuracy (which based on how many bits the hardware can support).

2: the FFT is not very slow at the software side, however, when it comes up to large data FFT conversion, the software will be slower for sure because the two-dimensional loop takes O(n^2) operating times; the FFT in hardware side may act a little slower for small FFT conversions because it will take some time for data transfer from software to hardware or memory map I/O, but since the total amount of data for the FFT to deal with is constant, the hardware side will be faster for larger segmentation of the total data.

Approach -> Try to find a relative large segmentation number N for FFT do the conversion. (Since 1.5n^2 + 1000000n will be faster than 2n^2 + n for a large n.)

3: the data cannot be insanely large to make the performance seemingly apparently tend to hardware side, since the sample number in the audio file is not infinite!

Approach -> In my approach, the data number is 2^16, which is a very “good” number but still not large enough to make the performance act “favor of” SSR, but this number 65536 for almost the best quality wave file 48kHz is already a segmentation greater than 1.36 secs!

**Challenge branch 2:**

1: hard to test for the real sound, hard to tell whether the effects are “good”, “bad”, “better” or “worse”!

2: lots of bugs software side and hardware side! (see bugs&solution.docx)

3: the understanding of high/low filter is limited! We usually take the frequencies to be the “filter” and the real data to be “filtered”, however, for the higher purpose of my project, we need to do something directly on the frequency data!