Guitar Simulator - An Application of Real-Time Signal Processing

Tianyou Li, tl2965
Tianshu Wang, tw2119
Yu-Lin Shen, yls247

New York University

Abstract

Digital Signal Processing has been a promising field, by which we could simulate various sound effects, and this technique helps us achieve breakthroughs in many domains, for example, generating animation sound in movies, music, and so on. In our work, we apply digital signal processing-related knowledge to implement a range of sound effects to simulate what guitar sounds like, and produce a user interface for them.

Table of Contents

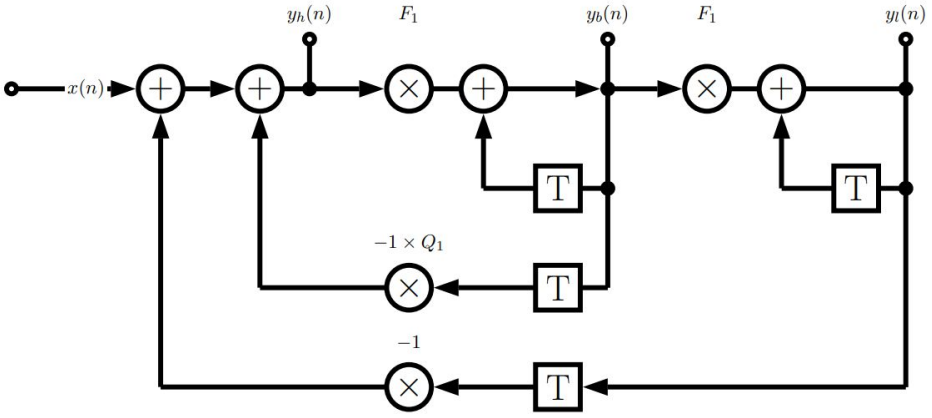
1 Introduction

Designing an application for various guitar effects is our primary goal in this project. Instead of having a real guitar, we produce sounds from signals. Typically, we classify effects from the perspective of how a sound is translated by a specific effect. Therefore, we have time varying (wah-wah), delaying (chorus, flanger, vibrato), modulation (ring, tremolo), non-linear processing (distortion) and some special effects (reverberation). On top of this, we build our user interface, in order to ease the pain of operating our system through a terminal, where users could interact with our system in a more user-friendly way. Overall, we team up to build a practical guitar effect simulator, hence users could access different guitar-like features either through a microphone or a keyboard.

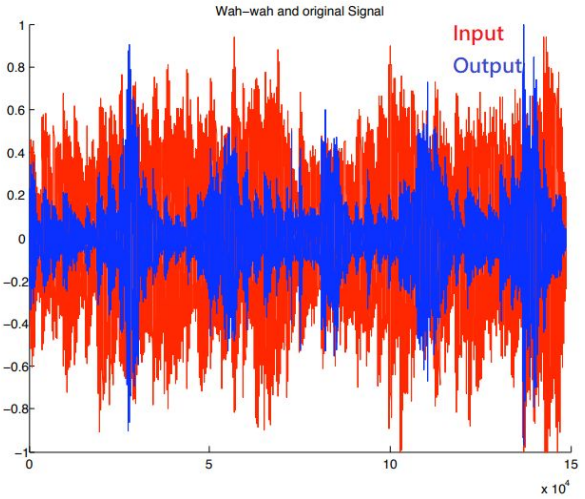
2 Sound Effects

2.1 Time varying

For time-varying effects, we tried the wah-wah effects. Based on [1], it is the signal filter that synthesis the previous effect and a time-varying effect. This filter contains a lowpass filter signal, bandpass filter signal, and highpass filter signal. It is named wah-wah due to the sound similarity on the "wah" sound [2]. The algorithm functions and the signal input & output are listed below [1]:



Signal structure of wah-wah (source from [1])



Input & output comparison of wah-wah (source from [1])

$$
\begin{aligned}
y_l(n) &= F_1 y_b(n) + y_l(n-1) \\
y_b(n) &= F_1 y_h(n) + y_b(n-1) \\
y_h(n) &= x(n) - y_l(n-1) - Q_1 y_b(n-1)
\end{aligned}
$$

Algorithm of wah-wah (source from [1])

The note and the explanation of the variables are listed below:
(a). x(n): input signal
(b). Q1: 2 times damping (d)
(c). yh(n): the output of the highpass filter, which source from the x(n), yl(n-1), and yb(n-1)
(d). yb(n): the output of the bandpass filter, which source from the yh(n), and yb(n-1)
(e). yl(n): the output of the lowpass filter, which sources from the yb(n), and yl(n-1)
(f). d: damping, a ratio on how many previous bandpass filter signal is been used in the high pass filter signal
(g). fc: the circular sequence, can be pre-defined
(h). fs: input signal sampling rate
(i). F1: 2*sin(pi*fc/fs), except fc, all other parameters are fixed

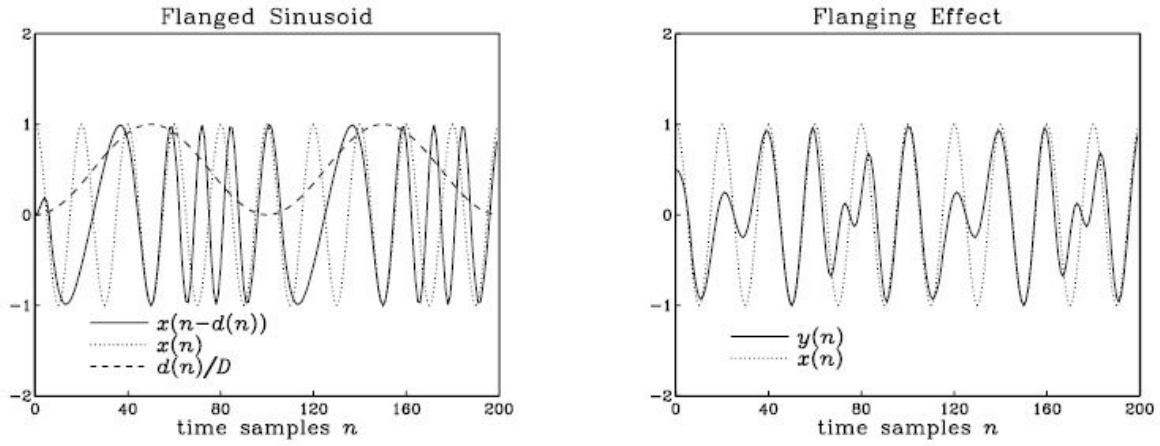
Basically, in one round, the input signal will be used to generate yh(n), then generate yb(n), and generate output yl(n). We will finally use the yb(n) as the output.

The wah-wah effect has been actually used in the 1960s for pop music, which was played by electric guitar and controlled by the padel [3]. This pedal effect had been mostly used in the 1980s to 1990s [2]. Back to the time in England, people can hear the effect in most of the songs. Even some of the performers use it in the whole album while releasing.

2.2 Delay

We chose chorus and flanger effects to implement for delay effects. They work slightly differently when it comes to their delay range, as chorus is range from 10 to 25ms, and flanger is in 0 to 15ms [1]. However, they can be described by one equation, y(n) = x(n) +

ax(n-d(n)) [7], where d(n) = D/2 * (1-cos(2 pi Fd n)). Intuitively, the equation produces an effect that one signal is superimposed by a variant of itself with a certain length of delay. The figure from [7] also demonstrates this as below.



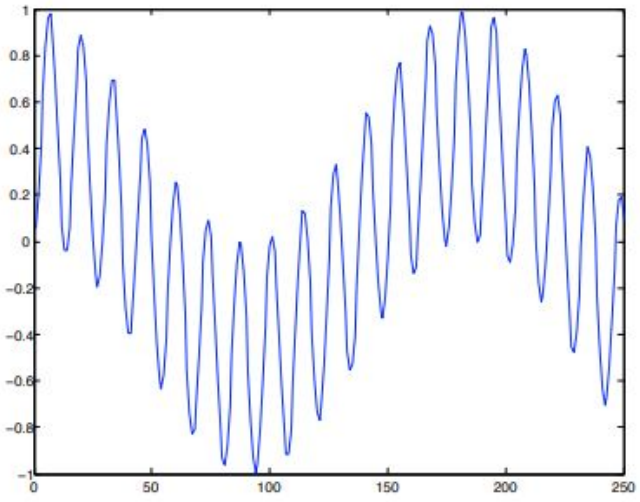
flanger effect (from [7])

2.3 Modulation:

For modulation, we have implemented two filter effects: ring modulation and tremolo.

2.3.1. Ring modulation:

Ring modulation is the synthesis effect from the input signal and the signal from the sine function. The output signal and the algorithm function are listed below:



Output signal of ring modulation (source from [1])

$$y(n) = x(n) * m(n)$$
$$m(n) = \sin*(\pi * \theta(n) * (Fc/RATE))$$

Algorithm function of ring modulation (source from [1])

(a). y(n): output signal

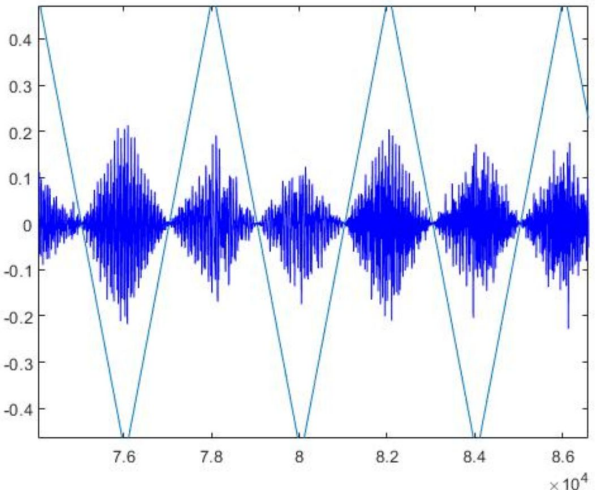
(b). x(n): input signal

(c). m(n): periodic signal generated from sine function

(d). theta: the incremental index, increased by time
(e). Fc: fixed used frequency parameter
(f). RATE: input signal sampling rate
We used the filter to generate a periodic effect. This is mostly used in the science movie that shows up as the robot sound [1].

2.3.2. Tremolo modulation:

Tremolo is also a periodic filter. Originally, it synthesises with the sine function and generates the sound wave. But, here, we use the much more simpler version. We only multiple the input sound with a circular sound wave (discrete) and output the wave file. This is called the tremolo with the ring modulation in [1]. It is just like multiple input signals with the triangular function. The algorithm function is listed below:



Output of Tremolo modulation with ring modulation (source code from [1])

$$y(n) = x(n) * Fc$$

Algorithm function of Tremolo modulation with ring modulation (source from [1])

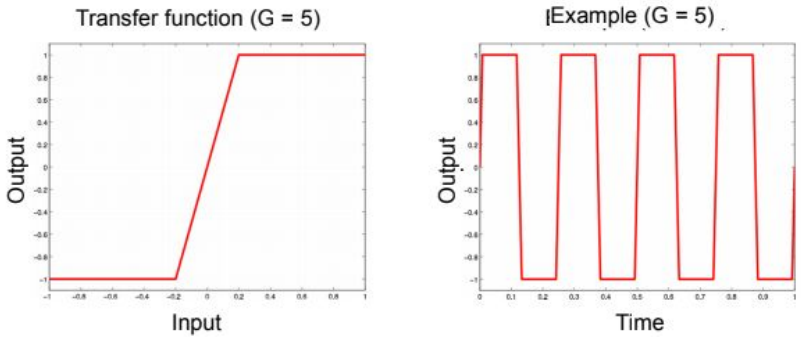
This is also a well-known usage on electric guitar and controlled by pedal as well.

2.4 Non-linear processing

The non-linear processing includes overdrive, distortion and fuzz effects. These concepts can be treated as the same thing and the implementations are identical. These effects are non-linear processing and memoryless which means that the current signal is independent to the previous signals and we don't have to save the signal of a certain period of time into a buffer.
Various kinds of clipping methods were used in our project: hard clipping, soft clipping, soft clipping exponential, half-wave rectifier, full-wave rectifier.

For hard clipping, the formula is as follows. If Gx is larger than or smaller than a threshold, we simply cut the higher or lower signals and keep the maximum and minimum values.

$$f(x)=\begin{cases}-1 & Gx\le -1\\ Gx & -1<Gx<1\\ 1 & Gx\ge 1\end{cases}$$

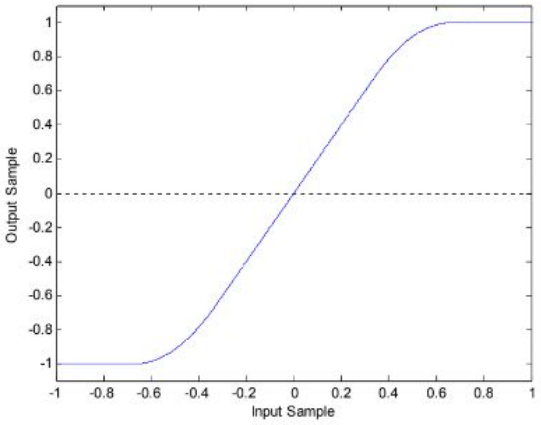


For soft clipping, it provides a smooth transition from linear to nonlinear, creating round corners at the peak of the waveform. In general, soft clipping produces a smoother, warmer sound, whereas hard clipping produces a bright, harsh, or buzzy sound.

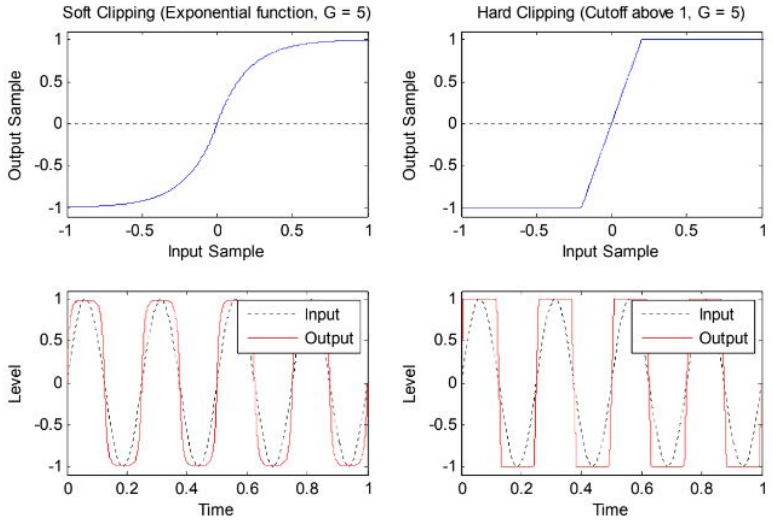
Notice: G = 2

$$f(x)=\begin{cases}2x & 0\le x<1/3\\ 1-(2-3x)^2/3 & 1/3\le x<2/3\\ 1 & 2/3\le x\le 1\end{cases}$$

Characteristic input/output curve for a quadratic distortion.

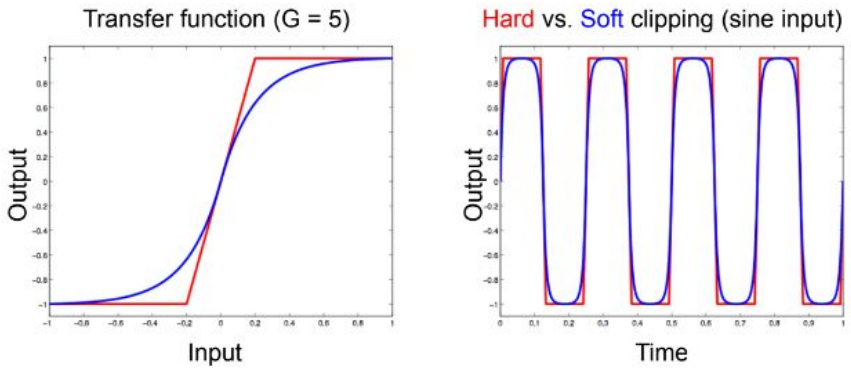


The comparison of hard clipping and soft clipping is as follows.



For soft clipping exponential algorithm, the curve looks like this. Exponential function grows gradually more nonlinear as input gain increases.

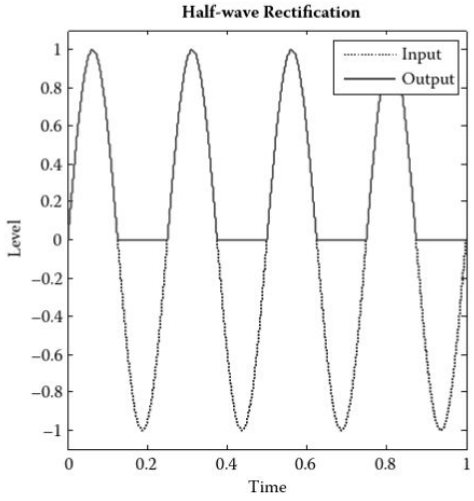
$$f(x) = \mathrm{sgn}(x)\left(1 - e^{-|Gx|}\right)$$

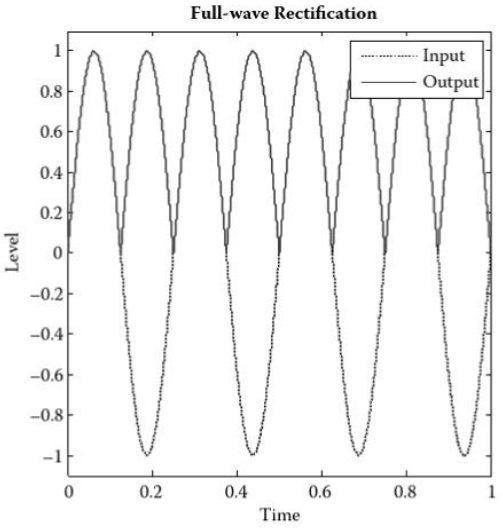


Half-wave rectification and full-wave rectification belongs to asymmetrical functions. A lot of sounds produced by real guitars don't behave like symmetrical waves, and that's why rectification functions come to our help.

$$f_{half}(x) = \max(x, 0)$$

$$f_{full}(x) = |x|$$



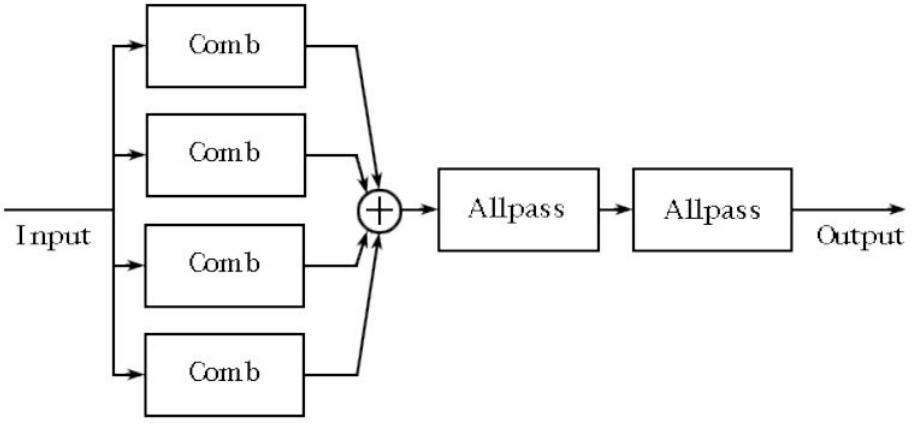


2.5 Reverberation:

Based on the class slide, this is the sound reflection from the source input signal to multiple spaces. It operates a series of delayed and attenuated effects to generate the output. In our proposal, we built three different kinds of reverb: Schroeder Reverb, Moorer Reverb, and Convolution Reverb.

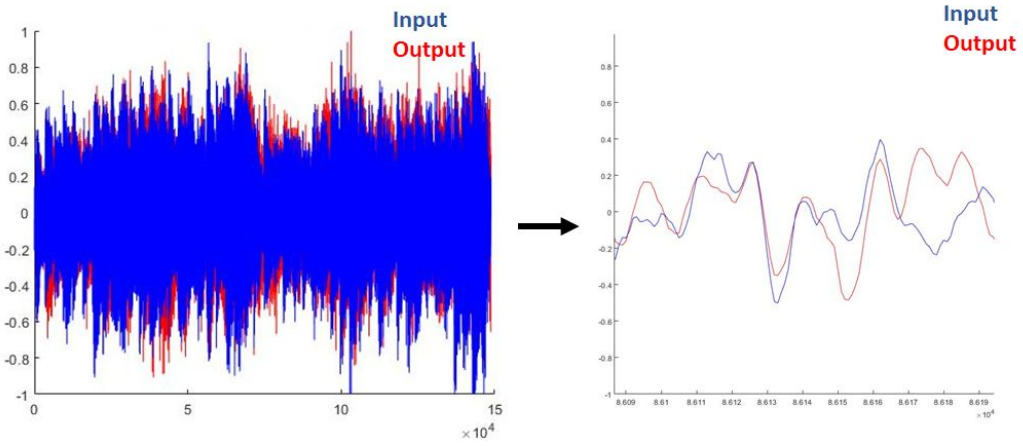
2.5.1 Schroeder Reverb:

For the well-known schroeder reverb example, it first run through four Comb filters parallelly and then combine to run through two All pass filters, which look as below.



Typical example of Schroeder Reverb(source from [1])

In our proposal, we didn't use the Comb filters, instead, for the simpler version, source from [1], we only replace the Comb filter to All pass filter and make it to 6. The signal result is listed below:

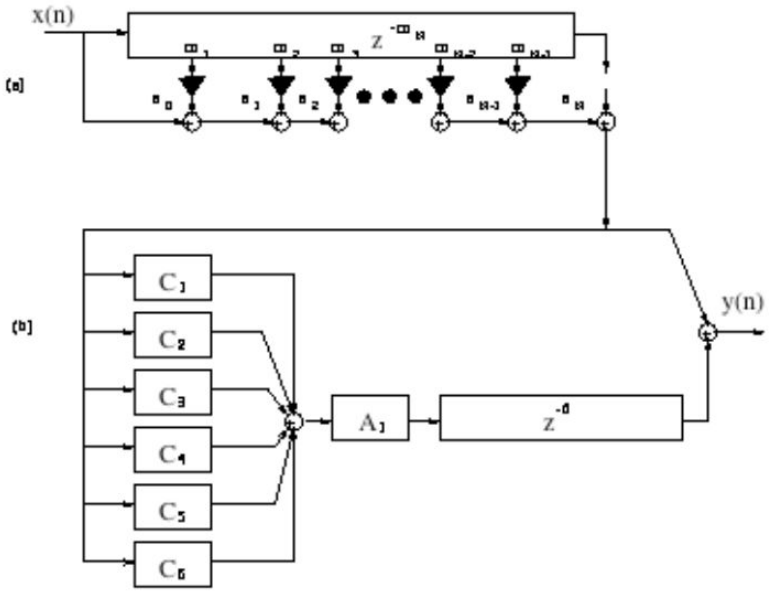


Input & output comparison of Schroeder Reverb (source code from [1])

Here are some notes for the All pass filter [4] and the comb filter [5]. The former is the filter that outputs the signal with the same gain factor but modifies the phase, which depends on the difference of the input frequencies. The latter is the filter that is processed by adding a delayed version of itself.

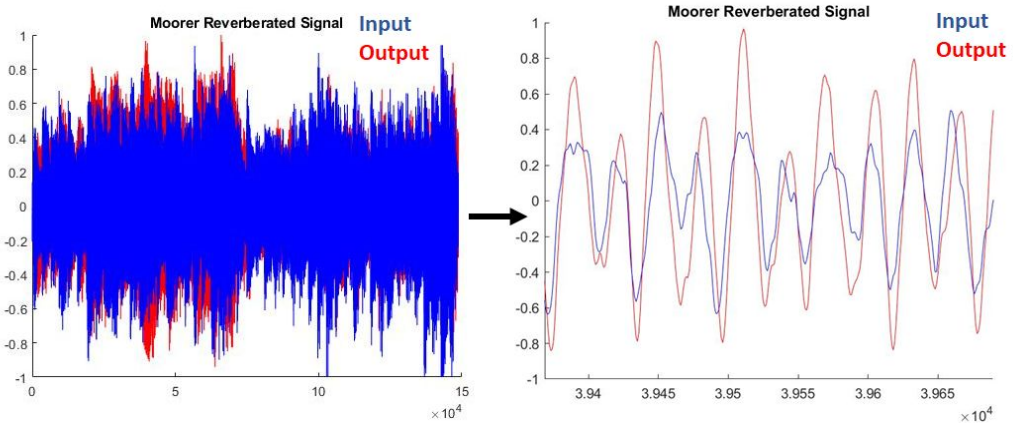
2.5.2 Moorer Reverb

We then move on the Moorer reverb, which is built upon Schroeder [1]. The Moorer Reverb has two part: the first part (b in below graph) is combined with the different parallel Comb filters and run through an All pass filter. The second part (a in below graph) is the tapped delay lines that simulate the reflections and then forwarded to part b [1]. The rough scratch is listed below [1]:



Sample structure of Moorer Reverb (source from [1])

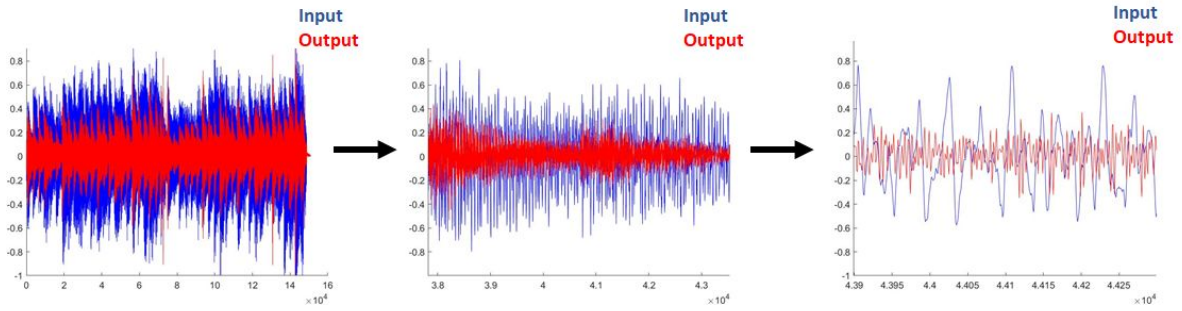
Here, for the simpler version, we only build part b in this proposal as [1]. According to [1], each filter has its own effects. For paralleled comb filters, which stand for Ci above, we apply them by different delay lengths. It represents the reflections from different obstacles. And for the All pass filter (Ai), it helps to add the density of the reflection. This graph has a really similar effect with Schroeder but with different structure. The output is listed below:



Input & output comparison of Moorer Reverb (source code from [1])

2.5.3 Convolution Reverb

If you want to generate a specific sound output, we believe that this is the most straightforward method, which is simply convoluted with the desired signal. Basically, this can use any signal to synthesis the output. In our experiment, we use the guitar click to convolute the original signal input and use the discrete convolution function to generate the output. Below is the output graph that we have:



Input & output comparison of Convolution Reverb (source code from [1])

To explain this, basically, we have the original input signal (x) and the desired synthesis input signal (h) as well. We first process the the fft function on both the signal with the both length of log2(length of x + length of h -1). Then, convolute both x & h and input to the ifft function. Eventually, output the real part of the synthesis signal.

According to [6], the technique have various applications, including simulate machine, electronic music and real space audio.

2.6 Phase Vocoder

We implemented an effect called robotization as a demonstration of phase vocoder. The phase vocoder is based on the short-time Fourier transform which operates only a small segment of the input signal, giving a snap-shot of the frequency content of the signal at a particular moment in time. In the program, we collect the sound signals in small windows and apply a fast Fourier transform operation for that window. By modifying the frequency content in each frame, many new effects are possible that are not easily implemented in the time domain.

Robotization is a very simple application of phase vocoder. It applies a constant pitch to the signal while preserving the vocal formants that determine vowel and consonant sounds, resulting in a robot-like monotone voice that is nonetheless very intelligible.
Following the FFT at each frame, the phase of every frequency bin is set to zero, while the magnitude is left unchanged.
Suppose the value of frequency bin k is a+jb, then the output value following robotization will be sqrt(a^2+b^2).

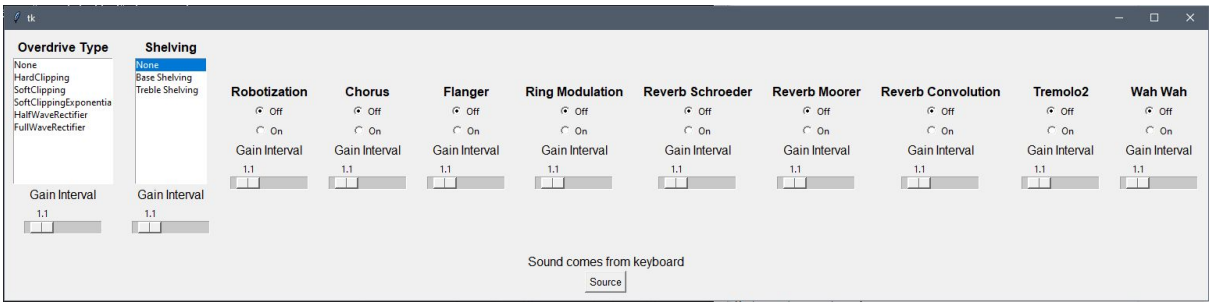
This procedure should be repeated identically for each bin of each frame. By preserving the magnitude, the overall shape of the spectrum remains the same, preserving the vocal formants. But by regularizing the phase phase information, each frequency component will effectively restart from zero phase on each hop rather than connecting smoothly from one hop to the next. This causes a constant audible pitch that depends on the hop size. In general, the pitch of the robot voice can be determined by f(robot) = f(s)/H, where f(s) is the sample rate and H is the hop size in samples.

The sound of the robotization effect also depends on the window size. In general, moderately sized windows (roughly 256 to 1024 samples) produce the most striking effect. Very small

windows reduce the clarity of the output, where longer windows attenuate the robot-like quality by passing through more of the signal's original pitch.

3 Results

The overall GUI is built like this.



Different types of effects are provided with a name and a switch. Some of them also have some options of various implementing methods that will result in different sound effects. Under each effect there is a gain controller to modify the amplitude.

The sound signals could come from the microphone as well as being generated by the program. We use a bunch of keys from C4 to C5. Users may choose the effects that he wants and hear the output sound from the speaker. Different sound effects can be processed serially.

A demonstration video is provided in this link if interested:
https://stream.nyu.edu/media/Real+time+sound+processing+and+guitar+simulator/1_yib6aj72

4 References

[1]. basic filtering lets, "Digital Audio Effects," Cf.ac.uk. [Online]. Available: http://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf. [Accessed: 20-Dec-2020].
[2]. Wikipedia contributors, "Wah-wah pedal," Wikipedia, The Free Encyclopedia, 19-Dec-2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Wah-wah_pedal&oldid=995071714. [Accessed: 20-Dec-2020].
[3]. W. Wah, "Marion, Jean Guy Bruno," Edu.au. [Online]. Available: https://ses.library.usyd.edu.au/bitstream/handle/2123/10578/Marion%2C%20Bruno%20-%20Wah%20Wah.pdf. [Accessed: 20-Dec-2020].
[4]. Wikipedia contributors, "All-pass filter," Wikipedia, The Free Encyclopedia, 19-Dec-2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=All-pass_filter&oldid=995078480. [Accessed: 20-Dec-2020].
[5]. Wikipedia contributors, "Comb filter," Wikipedia, The Free Encyclopedia, 19-Dec-2020. [Online]. Available:

https://en.wikipedia.org/w/index.php?title=Comb_filter&oldid=995210910. [Accessed: 20-Dec-2020].
[6]. Wikipedia contributors, “Convolution reverb,” Wikipedia, The Free Encyclopedia, 12-May-2020. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Convolution_reverb&oldid=956250672. [Accessed: 20-Dec-2020].
[7]. S. J. Orfanidis, Introduction to signal processing: International edition. Upper Saddle River, NJ: Pearson, 1995.
[8]. J. D. Reiss and A. McPherson, Audio effects: Theory, implementation and application. Boca Raton, FL: CRC Press, 2014.