REALIZATION AND COMPARISON ON UKULELE SOUND SYNTHESIS MODELS MUMT 618 FINAL PROJECT

Yinan Zhou
Music Technology, McGill University
yinan.zhou@mail.mcgill.ca

ABSTRACT

The goal of this project is to build and compare different models that synthesize ukulele sound. The body and air response of a ukulele is first measured. I then implemented both Karplus-Strong algorithm and digital waveguide model. To improve the result, a timbre control filter and a pluck position filter are added before the input signal is fed into the loop. The results of the two methods are compared in terms of spectrum and spectrogram. The effects of different summation locations in digital waveguide model are also studied. Finally, I wrote a script to play MIDI files using the generated ukulele sound.

1. INTRODUCTION

Ukulele is getting increasingly popular due to its small size and its easiness to learn. Musicians and music enthusiasts can use synthesized ukulele sounds to reduce the cost of music production. Another advantage of synthesized sound is that it has unlimited frequency range because the instrument is virtual. This project implements two algorithms to produce ukulele sound.

Many techniques have been developed to generate musical sound. In 1983, Karplus-Strong (KS) algorithm was first proposed [3]. This simple and efficient algorithm produces a rich and natural plucked-string sound. Julius Smith and David Jaffe then explained its relationship to physics and proposed several extensions of the method [1]. Later, Julius Smith introduced the digital waveguide theory based on physics [5]. In 1998, Matti Karjalainen and others showed the derivation from digital waveguide to an extension of Karplus-Strong algorithm [2].

The first aim of this project is to build a model that can generate ukulele sound, and to play a MIDI file with it. The second aim of this project is to compare the results of Karplus-Strong algorithm and digital waveguide model. Although the two methods are equivalent with the linear and time-invariant (LTI) assumption, the resulting sounds are slightly different. The results of digital waveguide models with different summation locations are also compared. The spectrums and spectrograms are analysed to evaluate the similarity and difference.

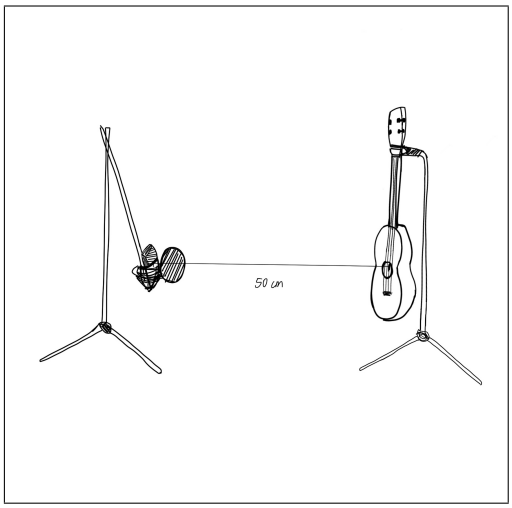

Figure 1. Recording Setting

2. METHODOLOGY

2.1 Body-to-Air Response

The classic model of plucked string instruments includes four components: excitation, waveguide, body response, and air response [4]. Due to the LTI assumption, body and air response can be commuted together and used as an excitation signal.

The first step of this project is to measure the body-to-air response of a ukulele. The setting of the recording is illustrated in Figure 1. The ukulele measured here is a general concert ukulele. The body-to-air response is recorded via the following steps:

1. Suspend the ukulele on a guitar stand with a height-adjustable neck so that nothing touches the body;
2. Damp the strings;
3. Place a R∅DE NT1-A microphone 0.5 meters right in front of the soundhole;
4. Use a drumstick to tap on the bridge in the horizontal direction, and record the sound;
5. Trim the audio to get the short tap part.

2.2 Timbre Control Filter

Figure 2 shows the spectrum of the real ukulele sound at G4 in the range of $0Hz$ to $6kHz$. It is worth noting that the signal nearly has no frequency components above $4kHz$. Thus, a timbre control filter with low-pass characteristics is added before the excitation signal is fed into the pluck posi-

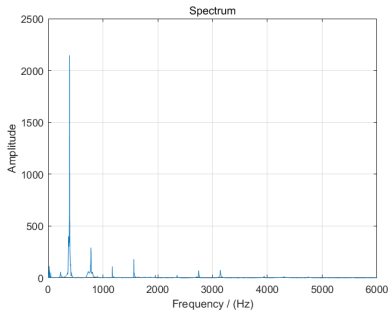

Figure 2. Spectrum of Ukulele at G4

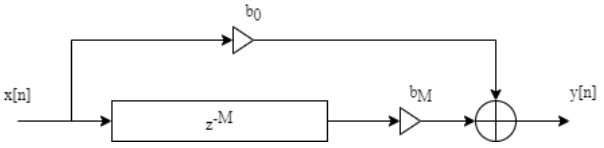

Figure 3. Pluck Position Filter

tion filter. The low-pass filter designed here is a Chebyshev filter.

2.3 Pluck Position Filter

The pluck position filter $P(z)$ simulates the standing wave pattern that the nth partial and its integer multiples will not be excited if the string is plucked at $1/n$th the distance from one end. The general pluck position of a ukulele is around the soundhole, which is approximately $8cm$ from the bridge. The string length of a concert ukulele is $38cm$. Therefore, n is rounded to 5 here.

The pluck position filter is actually a feedforward comb filter. It can create M notches at equal frequency intervals of f_s/MHz, where f_s is the sample rate. Thus, to simulate the pluck position, the signal is fed into the comb filter of order M shown in Figure 3. The order M is a fraction of the total delayline length. The transfer function can be expressed as

$$P(z) = b_0 + b_M z^{-M}. \tag{1}$$

2.4 Digital Waveguide Model

Digital waveguide model simulates the vibration of a string from a physics standpoint (Figure 4). It consists of two delaylines that represent two opposite travelling waves. The string of a ukulele has terminated ends on both sides. The displacements of the travelling waves will be inversed at both ends. Thus, the values are multiplied by the reflection coefficients after the waves reach the end. Here, the reflection coefficients are set to -1.

In addition, a string attenuation filter is needed so that the string vibration could decay over time. In this project, I used a one-zero filter, whose transfer function is

$$G(z) = \rho[(1 - S) + Sz^{-1}]. \tag{2}$$

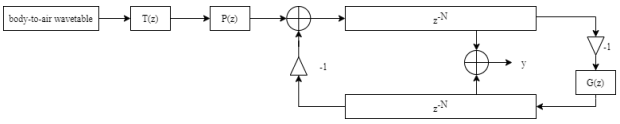

Figure 4. Original Digital Waveguide Model

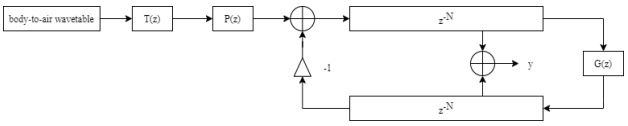

Figure 5. Lumped Digital Waveguide Model

In this filter, the loss factor ρ controls the overall decay rate. It can be obtained with the desired decay time. In this project, the value is set to 0.995 based on experiments to make the output not decay too slowly or too fast. However, the decay produced by ρ is not frequency dependent.

The stretch factor S simulates the frequency-dependent damping effect. It is a value between 0 and 1, which is used to adjust the relative decay rate for high frequencies in contrast to low frequencies. Stretch factor S is set to 0.5 here, to get the fastest high-frequency decay.

Moreover, the string attenuation filter can be lumped with the left multiplier (Figure 5). The transfer function of the string attenuation filter then becomes

$$G(z) = -\rho[(1 - S) + Sz^{-1}]. \tag{3}$$

Now, filter $G(z)$ represents both losses and inversion that happen at the left end of the string.

The total delay in the loop determines the frequency that corresponds to the pitch of the resulting signal. In this model, two delaylines provide a phase delay of $2N$ samples in total. The phase delay of the string attenuation filter is half a sample. Thus, the phase decay of the loop is $2N + 0.5$ samples. The desired phase decay can be expressed as f_s/f_0, where f_0 is the desired fundamental frequency of the output. Therefore, the delayline length N can be expressed as

$$N = \frac{\frac{f_s}{f} + 0.5}{2} samples. \tag{4}$$

One problem is that the length of the delayline has to be an integer. Thus, the delayline length must be rounded, which will cause tuning problems, especially for high frequencies. To address this problem, I implemented all-pass interpolation to allow a fractional delayline length.

The output is the sum of the two delaylines at a certain read location, in this case, the soundhole.

2.5 Karplus Strong Algorithm

Owing to the LTI assumption, the two delaylines can be combined into one. Besides, two reflection coefficients can also be lumped together and cancel each other out. As a result, a delayline with double the length above and the original string attenuation filter are left in the feedback loop (Figure 5).

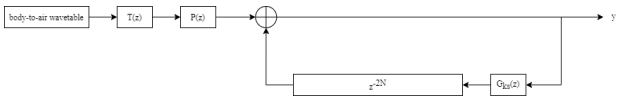

Figure 6. Karplus Strong Algorithm

The resulting model is an extended form of Karplus-Strong model. The original Karplus-Strong model implements a two-point average to simulate the frequency-dependent decay. The transfer function is

$$H(z) = \frac{1}{2}(1 + z^{-1}). \tag{5}$$

The transfer function $H(z)$ is the same as $G(z)$ when ρ is set to 1. In this project, I set ρ the same value as above. Thus, the transfer function becomes

$$G_{ks}(z) = 0.995 \times \frac{1}{2}(1 + z^{-1}), \tag{6}$$

which is the equivalence of the original string attenuation filter in the digital waveguide model.

Another difference is that the original Karplus-Strong algorithm feeds white noises into the delayline. In the case of the ukulele, I used the filtered body-to-air response as the input of the delayline.

2.6 MIDI Implementation

To extract MIDI information, I utilized MIDI toolbox 1.1 (2016) [6]. MIDI toolbox is a collection of MATLAB functions to analyse MIDI files. More detailed information on this toolbox can be found on the GitHub page.

First, the function $readmidi()$ is used to read a MIDI file to a notematrix. The onset, duration, and velocity of each MIDI event can be obtained by extracting the corresponding column vector from the notematrix. Then, I implemented the function $midi2hz()$ to convert MIDI note numbers to frequencies. All the information is then fed into the two models as parameters.

Velocity describes the force with which a note is played. With larger velocity, the note produced will have larger amplitudes. When generating each note, therefore, I multiplied the amplitude of the body-to-air response by the velocity parameter to simulate different forces. The output is the summation of each note.

3. RESULT AND DISCUSSION

3.1 Body-to-Air Response

The spectrum of the body-to-air response obtained is shown in Figure 7. Figure 8 shows the body-to-air response after editing, which is the input signal of the system.

3.2 Perception

Both resulting sounds perceptually resemble the real ukulele sound within the frequency range of a ukulele. However, when the desired frequency is not in the ukulele

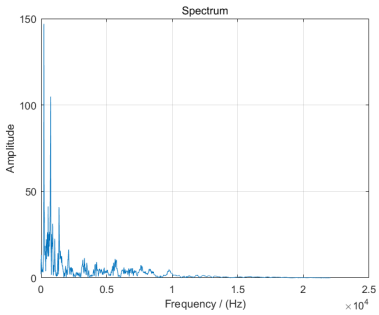

Figure 7. Spectrum of Body-to-Air Response of Ukulele

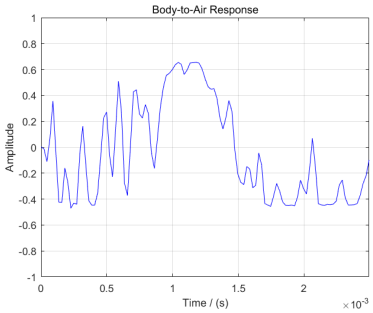

Figure 8. Input Signal of the System

frequency range, the results sound like a guitar for low frequencies and an xylophone for high frequencies.

Furthermore, although the two models are equivalent with the LTI assumption, the resulting sounds are slightly different from each other. The audio examples can be found via this link: `https://github.com/yinanazhou/ukulele-sound-synthesis`.

The sounds analysed below are all $G4$ notes, which is the 4th open-string of a ukulele.

3.3 Spectrum

A spectrum is a concise way to demonstrate the frequency component distribution of a musical signal. The spectrums of the two models are shown in Figure 9 and Figure 10.

As shown in Figure 2, the amplitudes of the third and fifth harmonic of the real ukulele sound are relatively lower. Compared to digital waveguide model, the spectrum of Karplus-Strong algorithm is more similar to that of the real sound in this manner. However, the amplitudes of the second harmonics of both models are relatively large.

Besides, both generated sounds have nearly no frequency components below the fundamental frequency, which correspond to the body resonances. Two reasons probably cause this. First, I tapped the ukulele with a drumstick made of wood. It would be better to measure the body-to-air response with a professional impulse force hammer made of metal. Second, the recordings contain room reverberation because the studio is not a standard studio.

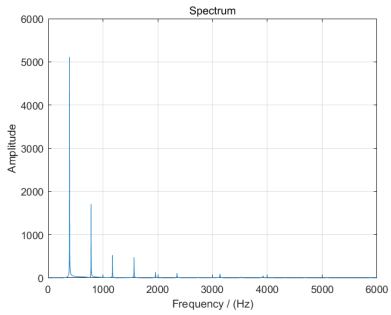

Figure 9. Spectrum of Digital Waveguide Model at G4

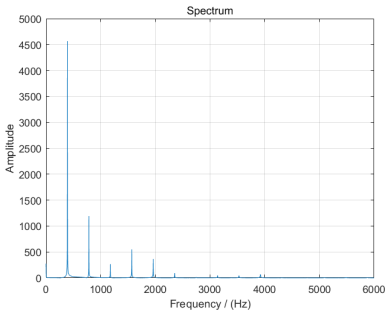

Figure 10. Spectrum of Karplus-Strong Algorithm at G4

3.4 Spectrogram

To compare the similarity over time, this project plots 3D spectrograms, also known as waterfalls. A spectrogram is a visual presentation that shows how various frequencies of a signal evolve over time. To calculate spectrograms, Short-Time Fourier Transform is calculated, with a hamming window of size 256 samples and hop size of 1 sample. The spectrogram of the real ukulele sound at G4 is shown in Figure 11.

Figure 12 and Figure 13 show the spectrograms of the two models without the timbre control filter. The resulting sounds from both models without coloration have more medium and high frequencies than the real sound. Thus, it is necessary to implement a low-pass filter before the wavetable is fed into the loop.

Figure 14 and Figure 15 presents the spectrograms of the sound generated based on digital waveguides and the sound produced using Karplus-Strong algorithm, both of which are colored by the low-pass filter. In terms of spectrogram, Karplus-Strong algorithm is also better than the digital waveguide model. In the beginning, the resulting sound of digital waveguide model contains more high frequencies. Besides, high frequencies in the result of digital waveguide model decay slower than those in the other two.

3.5 Summation Location of Digital Waveguide Model

In the signal shown in Figure 14, the summation location is set around the soundhole, which is the standard recording position of a ukulele.

Figure 16 and Figure 17 are the spectrograms of the signals generated by digital waveguide models, in which the sum is respectively made at the left and right end of the string. Both signals have more high frequencies than the result shown in Figure 14. It is worth noting that when the sum is made on the left end, the summation location of the output is the same as that of Karplus-Strong algorithm. Nevertheless, the outputs are different.

Figure 18 shows the spectrogram of the output of digital waveguide model, in which the summation location is in the middle. In this case, high frequencies decay the fastest. However, compared to the result of Karplus-Strong algorithm, there are still more high frequency components in the beginning.

4. CONCLUSION

This project implements both Karplus-Strong algorithm and digital waveguide model to synthesize ukulele sound. A system is then built to play MIDI files with the resulting sound of two models.

The results of Karplus-Strong model more resembles ukulele sound, both perceptually and analytically. As for the digital waveguide model, the location of the summation is critical to the quality of the output. The best result is got when the summation location is right in the middle. Moreover, the results of digital waveguide model and Karplus-Strong algorithm are different even when summing at the same location. Thus, it is possible that the difference between the two models is caused by the structure of the loop.

In further research, a more accurate body-to-air response should be measured and used as the input. In addition, a string vibrates in both vertical and horizontal directions. The vibration of one string also affects other strings on the instrument. The coupling effects between directions and between different strings can also be simulated.

5. REFERENCES

[1] David A Jaffe and Julius O Smith. Extensions of the karplus-strong plucked-string algorithm. *Computer Music Journal*, 7(2):56–69, 1983.

[2] Matti Karjalainen, Vesa Välimäki, and Tero Tolonen. Plucked-string models: From the karplus-strong algorithm to digital waveguides and beyond. *Computer Music Journal*, 22(3):17–32, 1998.

[3] Kevin Karplus and Alex Strong. Digital synthesis of plucked-string and drum timbres. *Computer Music Journal*, 7(2):43–55, 1983.

[4] Gary P. Scavone. Commuted waveguide synthesis. https://www.music.mcgill.ca/ gary/618/week5/node11.html, 2004-2020.

[5] Julius Orion Smith. *Music applications of digital waveguides*. Number 39. CCRMA, Department of Music, Stanford University, 1987.

[6] Petri Toiviainen and Tuomas Eerola. MIDI toolbox 1.1. https://github.com/miditoolbox/, 2016.

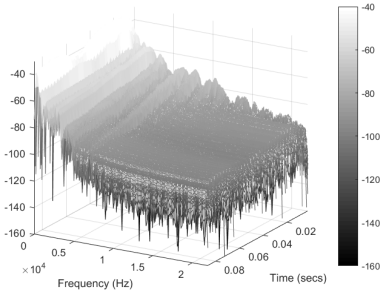

Figure 11. Spectrogram of Ukulele at G4

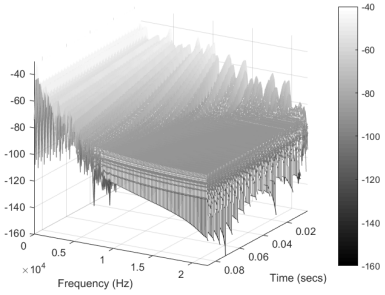

Figure 12. Spectrogram of Digital Waveguide Model at G4, without coloration

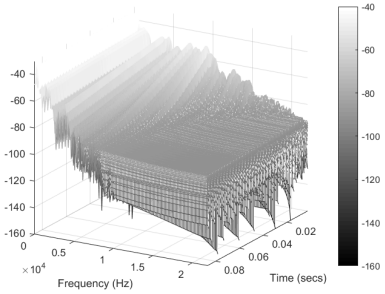

Figure 13. Spectrogram of Karplus-Strong Algorithm at G4, without coloration

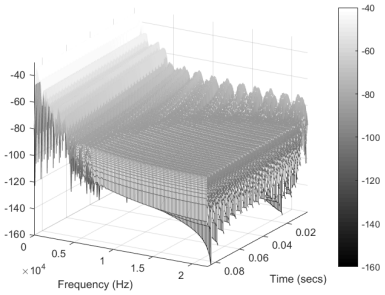

Figure 14. Spectrogram of Digital Waveguide Model at G4

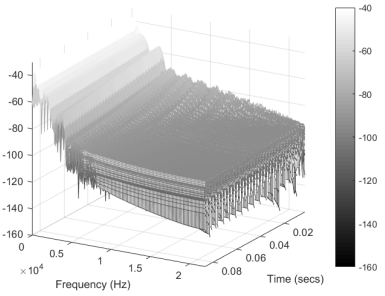

Figure 15. Spectrogram of Karplus-Strong Algorithm at G4

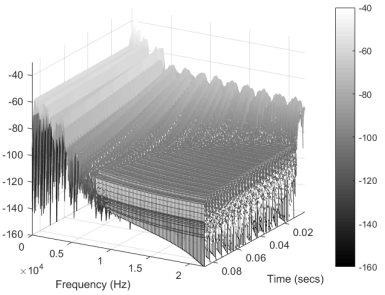

Figure 16. Spectrogram of Digital Waveguide Model, Summation at the Left End

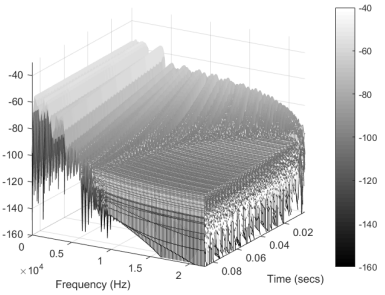

Figure 17. Spectrogram of Digital Waveguide Model, Summation at the Right End

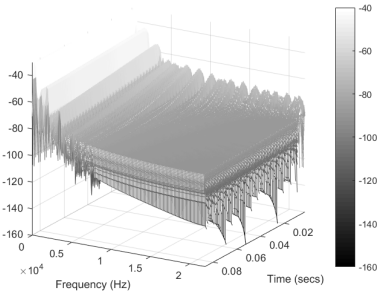

Figure 18. Spectrogram of Digital Waveguide Model, Summation in the Middle