Smart Sampling Synthesiser

Prachee Javiya, Kaushal Patil, Dhruvil Dave, Dhir Oza

Abstract

- We came up with an interesting goal - synthesising a sound sample and reproducing the timbre of the musical note.
- The harmonics of a wave file can be calculated by two methods-fft (Fast Fourier Transform) and Linear Regression.
- The latter gave us a vast environment to work with thereby allowing us to explore more and enhancing the quality of reproduced sound. We have used pyo python library to do the project. The inbuilt servers and audio drivers assist to synthesize the musical note.

SUMMARY

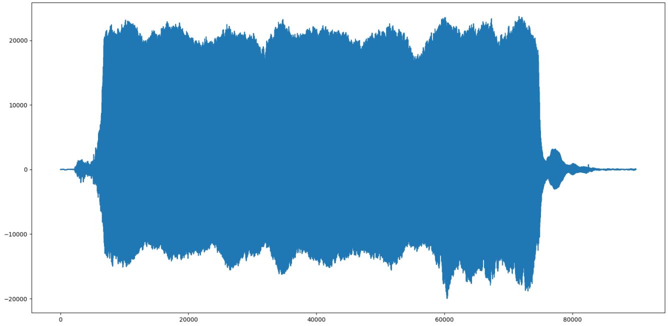
- An audio file (.wav format) is loaded and the sample indexes where the sample is in sustain is entered by the user.
- Using various algorithms the frequency of the input wave is calculated and shown to user out of which the user selects the frequency which looks accurate according to pitch of the sound.
- Then using Linear Regression,the weights first 40 harmonics of sine and first 40 harmonics cosine are calculated.
- Real-time values of the notes played by the user is taken by polling the midi input and sound is produced from the synthesiser according to it.
- To enhance the resemblance, the synthesiser features 2 extra oscillators of adjustable waveforms] and one frequency modulated oscillator (based on cos coefficients), an adjustable filter, harmonizers, distortion effect, reverb effect, chorus effect,.
- The final notes played can be adjusted as per requirement using the gui interface of synthesiser.

The entire procedure is divided into two parts

1) Training stage
2) Synthesis stage

Training - Stage 1:

The user loads a wavefile (a sample of few seconds of any recorded instrument). The programprocesses the wavefile and it is plotted. The user enters the limits of frequency(the part where itis in constant sustain gives a better output). The sound wave between that frequency is processed.



Then the frequency of that part is calulated using 4 different algorithms.

1. FFT(fast fourier transform)
2. 0 crossings
3. auto-correlation
4. HPS(Harmonic product spectrum)

Why 4 algorithms?

For instance , when the amplitude of the fundamental frequency happens to be less than the amplitude of higher harmonics of the wave, fft function fails to give an accurate output. Hence, the other three algorithms helps to determine the perfect frequency of the base audio file.

The program prints all the four frequencies and the user feeds the value which is most appropriate in the program.

Training - Stage 2:

- At this stage we have the frequency which is close to the pitch of the initial audio file.
- Using linear regression we calculate first 40 harmonics of sine and cosine wave and store them in a table.
- These harmonics are taken in a matrix and appended with the matrix of original audio file.
- The audio file is now trained and is sent to the synthesis stage.

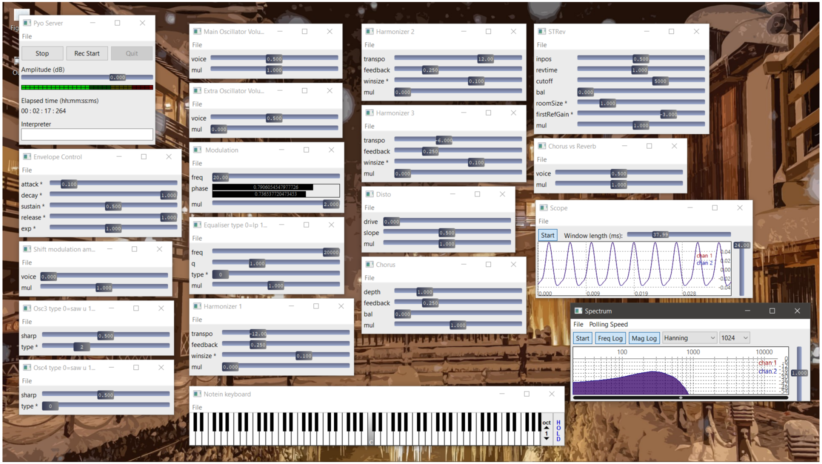
Synthesis stage:

- The sine and cos table has the values of coefficients of harmonics of sine and cosine wave respectively.The phase of cosine wave is 0.25 because period is scaled from 0 to 1.
- On keypress, real-time input of the note is read and its pitch and velocity is is fed into oscillator 1. The ADSR envelope sends the values of attack(0.1),Decay (1.0),Sustain (0.5),Release (1.0),Exp (1.0) to oscillator 1 simultaneously.
- Similarly the coefficients from cos table are fed into oscillator 2.
- For improving the qualtity of the sound further, 2 extra LFO oscillators are used which produces a periodic signal (square, sawtooth, triangular,pulse).
- The waves from oscillator 1 and 2 and LFO oscillators are sent to 'Mix' that combines the resultant waves from respective oscillators to produce a better sound.
- The final wave is then fed to a biQuad filter (Second order IIR Filter. It has two poles and two zeros. We have used this because of its coefficient sensitivness in higher order filters) which pipes the wave to 3 harmonizers and one distort.

Algorithms

1. FFT
2. Linear Regression ($x = \left(A^T A\right)^{-1} A^T b$)
3. Auto Correlation
4. Additive Synthesis
5. Zero Crossings
6. Harmonic Product Spectrum

User Interface



Pyo Server
File
Stop | Rec Start | Quit
Amplitude (dB) 0.000
Elapsed time (hh:mm:ss:ms)
00 : 02 : 17 : 264
Interpreter

Envelope Control
File
attack * 0.100
decay * 1.000
sustain * 0.500
release * 1.000
exp * 1.000

Shift modulation am...
File
voice 0.000
mul 1.000

Osc3 type 0=saw u 1...
File
sharp 0.500
type * 2

Osc4 type 0=saw u 1...
File
sharp 0.500
type * 0

Main Oscillator Volu...
File
voice 0.500
mul 1.000

Extra Oscillator Volu...
File
voice 0.500
mul 0.000

Modulation
File
freq 20.00
phase 0.7906054547977726 / 0.736537720473453
mul 2.000

Equaliser type 0=lp 1...
File
freq 20000
q 1.000
type * 0
mul 1.000

Harmonizer 1
File
transpo -12.00
feedback 0.250
winsize * 0.100
mul 0.000

Harmonizer 2
File
transpo 12.00
feedback 0.250
winsize * 0.100
mul 0.000

Harmonizer 3
File
transpo 6.000
feedback 0.250
winsize * 0.100
mul 0.000

Disto
File
drive 0.000
slope 0.500
mul 1.000

Chorus
File
depth 1.000
feedback 0.250
bal 0.000
mul 1.000

STRev
File
inpos 0.500
revtime 1.000
cutoff 5000
bal 0.000
roomSize * 1.000
firstRefGain * -3.000
mul 1.000

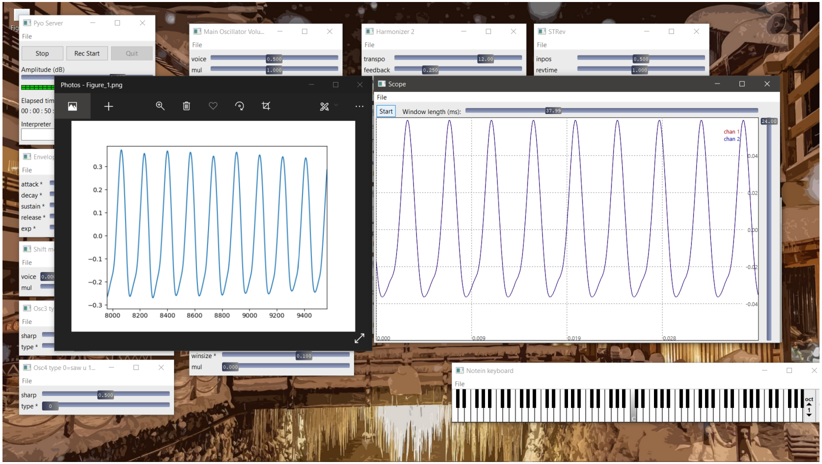
Chorus vs Reverb
File
voice 0.500
mul 1.000

Scope
File
Start Window length (ms): 37.99
24.00

Spectrum
File Polling Speed
Start | Freq Log | Mag Log | Hanning | 1024
1.000

Notein keyboard
File

Output



Pyo Server
Stop
Rec Start
Quit
Amplitude (dB)
Elapsed time
00 : 00 : 50
Interpreter
Main Oscillator Volu...
voice 0.500
mul 1.000
Harmonizer 2
transpo 12.00
feedback 0.250
STRev
inpos 0.500
revtime 1.000
Photos - Figure_1.png
Scope
Start
Window length (ms): 37.99
chan 1
chan 2
attack *
decay *
sustain *
release *
exp *
winsize * 0.100
mul 0.000
Osc4 type 0=saw u 1...
sharp 0.500
type * 0
Notein keyboard

Thank You

- Prachee Javiya (AU1841032)
- Kaushal Patil (AU1841040)
- Dhruvil Dave (AU1841003)
- Dhir Oza (AU1841108)