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.
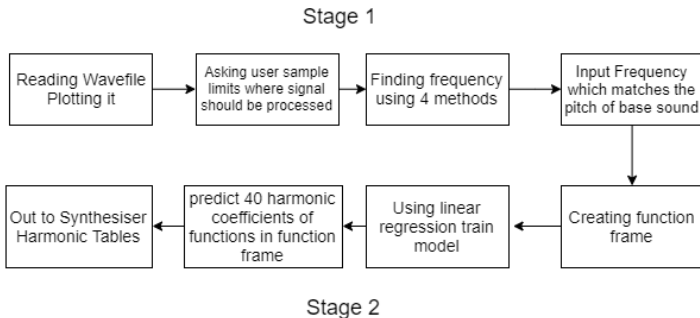
SUMMARY

- 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

Block Diagram

Training Stage

Stage 1

```
┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐
│ Reading Wavefile│──▶│ Asking user     │──▶│ Finding         │──▶│ Input Frequency │
│ Plotting it     │   │ sample limits   │   │ frequency       │   │ which matches   │
│                 │   │ where signal    │   │ using 4 methods │   │ the pitch of    │
│                 │   │ should be       │   │                 │   │ base sound      │
│                 │   │ processed       │   │                 │   │                 │
└─────────────────┘   └─────────────────┘   └─────────────────┘   └─────────────────┘
```

```
┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐   ┌─────────────────┐
│ Out to          │◀──│ predict 40      │◀──│ Using linear    │◀──│ Creating        │
│ Synthesiser     │   │ harmonic        │   │ regression      │   │ function frame  │
│ Harmonic Tables │   │ coefficients of │   │ train model     │   │                 │
│                 │   │ functions in    │   │                 │   │                 │
│                 │   │ function frame  │   │                 │   │                 │
└─────────────────┘   └─────────────────┘   └─────────────────┘   └─────────────────┘
```

Stage 2

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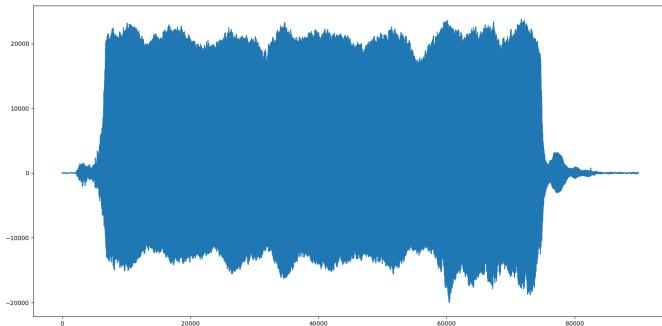
Then the frequency of that part is calulated using 4 different algorithms.

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.

- 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.
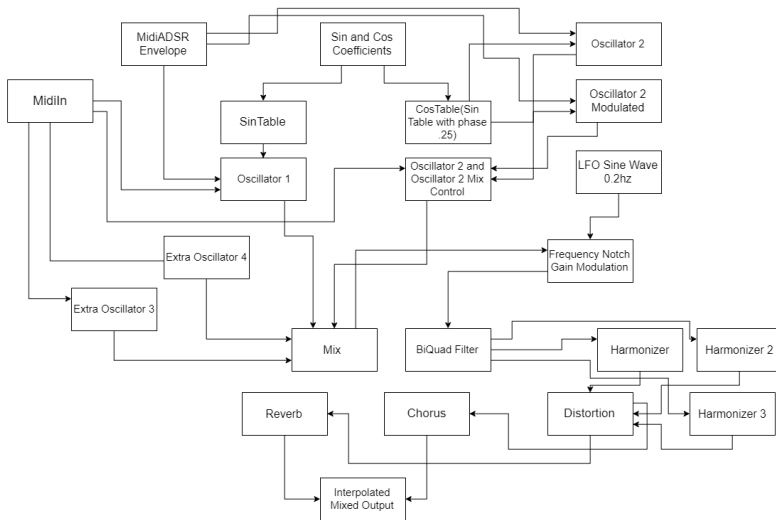- The audio file is now trained and is sent to the synthesis stage.
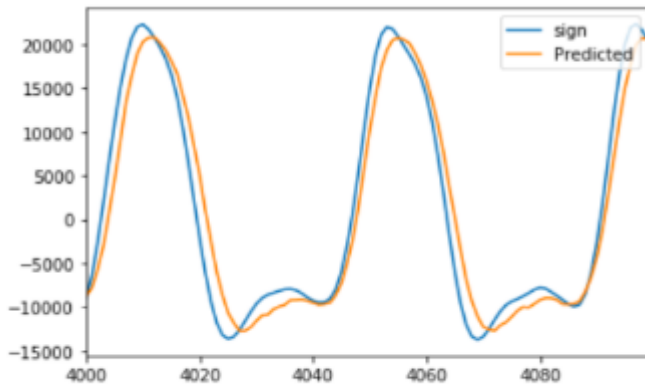
Original and Predicted Signal

Synthesis stage:

Block Diagram

- 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.

1. FFT
2. Linear Regression $\left(x = \left(A^T A \right)^{-1} A^T b \right)$
3. Auto Correlation