notGuitar

Joey Hall, Paul Chyz, and Jatin Chowdhury

March 9, 2018

1 Abstract

The idea for our project is to research, design, and implement a real-time Digital Signal Processing system that will take an input audio signal from a guitar, and output a signal with the same pitches and rhythms, but with the timbre of a saxophone. While most systems that allow melodies to be played with different timbres involve some form of MIDI conversion, and/or stored samples of instrument sounds to be used as the system output, our system will accomplish this timbral shift exclusively by manipulating the amplitude and frequency content of the input signal without using any stored recordings. Ideally, by the end of the semester, our system will be able to successfully convert single-note guitar melodies to the saxophone sound.

2 Description of the system

We plan to do this by using a non-linear filter bank to alter the overtone structure of the input signal, as well as replacing the amplitude envelope of the incoming guitar with that of the saxophone for the output signal.

- Sub-banding: We will be running our single note input from our guitar into linear bandwidth filters that will each output a harmonic frequency within the separate bands. The amount of filters will increase linearly toward the higher frequencies due to the number of harmonics within each of the higher octaves. Correct determination of the bandwidth of each filter should allow for no more than two harmonics from a given note to be present within a single sub-band of the filter bank. By comparing the power of the signal through each band of the auditory spectrum, we should be able to determine approximately which note the guitar is playing with enough precision to accurately implement the Overtone Control part of the system.
- Overtone Control: Manipulating the relative amplitude of the output of each sub-band filter will allow us to imitate the timbre of a different instrument. Guitars and saxophones have a unique harmonic structure, in that each overtone above the fundamental has a different amplitude relative to the fundamental. In other words for each instrument, the weighting of each overtone is unique. By storing the relative weighting of each overtone for a saxophone compared to a guitar, we can manipulate the amplitude of the guitar's input overtones so that the output's timber mimics that of a saxophone.
- Amplitude Envelope Modulation: Along with a unique harmonic structure, each instrument also has a unique amplitude envelope. Specifically, each instrument has a characteristic amplitude response with which it begins and ends each note, better known as an instrument's "attack" and "release." Together the characteristic attack and release of a instrument make up its amplitude envelope, which is a key component of it's timbre. By removing the amplitude envelope of the guitar from the incoming guitar sound, and applying the amplitude envelope of the saxophone to the generated output sound, our system should be able to accurately replicate the amplitude component of the saxophone's timbre.

In theory, since every melodic instrument has a unique harmonic structure and amplitude envelope, our system could be used to convert sound to from any melodic instrument to any other melodic instrument. In reality, our system only has enough memory to store the necessary timbral information for a few instruments, plus each instrument would also need some additional fine-tuning and calibration to be able to generate realistic instrument sounds.

3 Description of Possible Algorithms

- Sub-banding:The sub-banding filter bank will be composed of linear bandwidth Finite Impulse Response filters, than span the frequency spectrum of the guitar (about 80 Hz to 20 kHz), with a bandwidth small enough to allow no more than two harmonics in each sub-band regardless of the note being played. The output of each sub-band will be integrated to find the power in that band, and then the powers will be compared to determine which sub-band contains the fundamental frequency of the note. Since the fundamental frequency is typically the lowest frequency with any significant power output, our logic will determine the lowest frequency sub-band with any significant power output to be the sub-band containing the fundamental frequency.
- Overtone Control: As stated in the previous section, the output tone will be generated by manipulating the gain of each band of the sub-banding filter bank, depending on which sub-band contains the fundamental frequency of the note. The overtone amplitude weightings for the timbre of each instrument will be stored in a look-up table.

- Amplitude Envelope Modulation: There are three main parts to an instrument's amplitude envelope, the attack, the sustain, and the release.
  - Attack: The first part of modulating the attack of a sound is determining when the sound or "note" begins. After determining a threshold at which the note starts, the incoming samples will be multiplied by constants corresponding to the attack characteristic of the given sound. These coefficients will be determined from by analyzing recordings of the instruments, and will be stored in a look-up table.
  - Sustain: If sound is above the threshold for longer than the attack time, the sound enters a state known as the "sustain." For instruments with little to no damping the sustained amplitude is relatively constant, while for most others it tends to decrease with time. We were planning to leave the envelope coefficients for the sustain period at a constant amplitude to allow the dynamics of the melodic input to be translated through to the output.
  - Release: Once the signal drops below the threshold, the envelope enters the "release" state. The release envelope will act similarly to the attack envelope, in that it will function by multiplying the incoming samples by some coefficients stored in a look-up table, however the length and values of the release characteristic will be determined by the amount of time spent in the sustain part of the envelope. In this way, the envelope will react dynamically to the different types of notes (staccato, legato, etc.) that the input signal could contain.

4 Complexity Analysis

The main computational bottleneck will be the implementation of the sub-banding filter bank, since we will need to implement on the order of 100 individual FIR filters . Additionally, the computational load of the multiplication operations required to apply the weightings for the overtone series and amplitude envelopes could be a bit intensive. Since our system will inputting and outputting audio signal, both incoming and outgoing data rate must be 44.1 kHz. Since the DSP boards operate at 225 MHz [1], we should have time for 5,000 clock cycles of processing before encountering any issues with latency.

5 Major Challenges

The major challenges of designing and implementing our system include:

- Determining the correct bandwidth for the sub-banding filter bank, to allow for the maximum separation of harmonics into discrete bands regardless of the input note.
- Having enough processing power to implement an adequate filter bank.
- Efficiently multiplying the weights of the filters and the amplitude envelopes.
- Developing an envelope that can react differently depending on the length of the note being played, i.e. determining when a note ends.
- The maximum latency an audio system can produce before it becomes noticed by the user is about 17 milliseconds. Thus, the total latency of our system needs to be less than that in order for our system to wrok effectively in real-time.

6 Modeling, Simulation and Off-Line Prototyping

Our initial design will be shown with a block diagram describing the system. We will start collecting data by recording guitar and saxophone notes to use for testing. Our group has access to a guitar, a saxophone, and recording equipment that can be used during the data collection process. Testing and simulation will initially be done in MATLAB, with each member testing their components individually. We will create the bandwidth filters inside of MATLAB and test the filter bank with a recorded single note guitar sample, and will use separate test signals found online and from live recordings to check our MATLAB analysis of overtones and amplitude modulation. Once each member has completed their MATLAB simulation individually, we will implement and test all facets separately and then together on the DSP.

7 Human Factors

Musical instruments are a very subjective domain for listeners, as some things that sound good to one person may not gain the same approval from another person. People also have inherent bias in some scenarios that can influence their perception of things. To mitigate these human factors, the evaluation of our system will be a blinded test. With the listener unable to see the system in use, we will be able to switch the output between the output of our system and a recording of a real saxophone. This will allow the listener to compare the two different sounds without prior knowledge of each sound's source. If our system works perfectly, the listener would not be able to differentiate the real saxophone from the synthesized one.

8 Rough schedule

We plan to model each component of our system in MATLAB, and then implement and test each component on the DSP boards. Paul will be responsible for the overtone control component, Joey will be responsible for the sub-banding, and Jatin will be responsible for the amplitude envelope modulation.

We are planning to have our MATLAB prototyping finished by March 9th. We should have a working system on the DSP board by March 30th. The month of April will be spent debugging. We realize that spending 4 weeks on MATLAB prototyping might seem excessive, but we feel that it is needed to ensure that the algorithms for each component of our project are working, and since we will need MATLAB to generate the determine the weights to use for the weighted filter bank and the amplitude envelope. After Spring Break, we will still have 5 weeks to implement our system on the DSP board before our demonstration is due on April 23rd.

9 Final Test Set-Up

Our final test setup will include an electric guitar and a saxophone along with the DSK6713 board. The guitar will serve as our input source, and will be plugged directly into the DSK6713 input via a quarter inch to eighth inch cable. The output will require a set of headphones, where the listener will hear the real-time processed signal that will sound like a saxophone. Because an un-amplified electric guitar is so quiet, there is no need for significant separation between the guitar input and the listener. Additionally, we will have a saxophone for a live demo to compare the real instrument to our synthesized version. We have both a guitar player and a saxophone player in our group, which will allow us to have an exciting final test experience.

10 Board

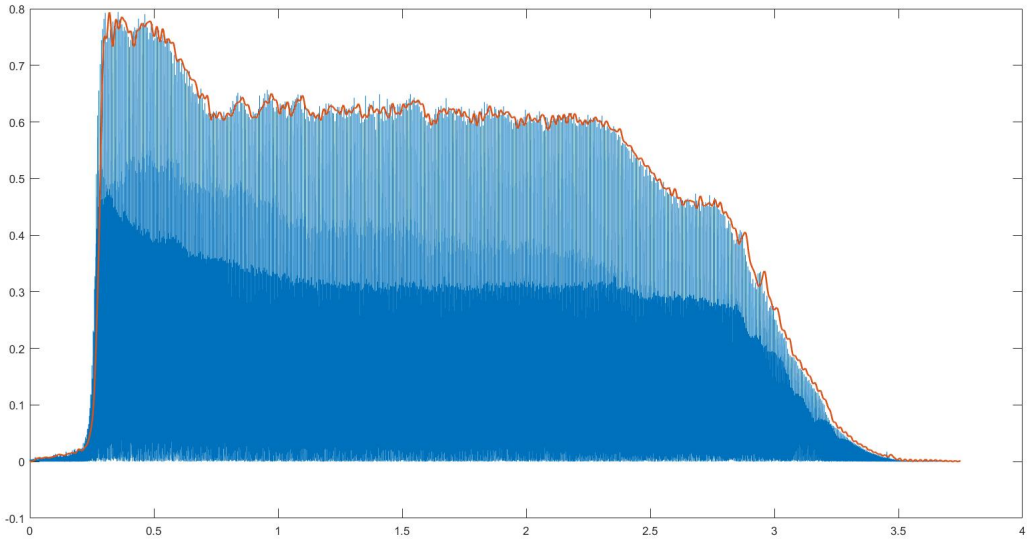
We are planning to use the DSK6713 for our project. For additional hardware we only require a guitar (which we have), as well as a microphone and some other recording equipment for acquiring data (which we also have).

11 Progress Report

11.1 Simulation Studies and Preliminary Results

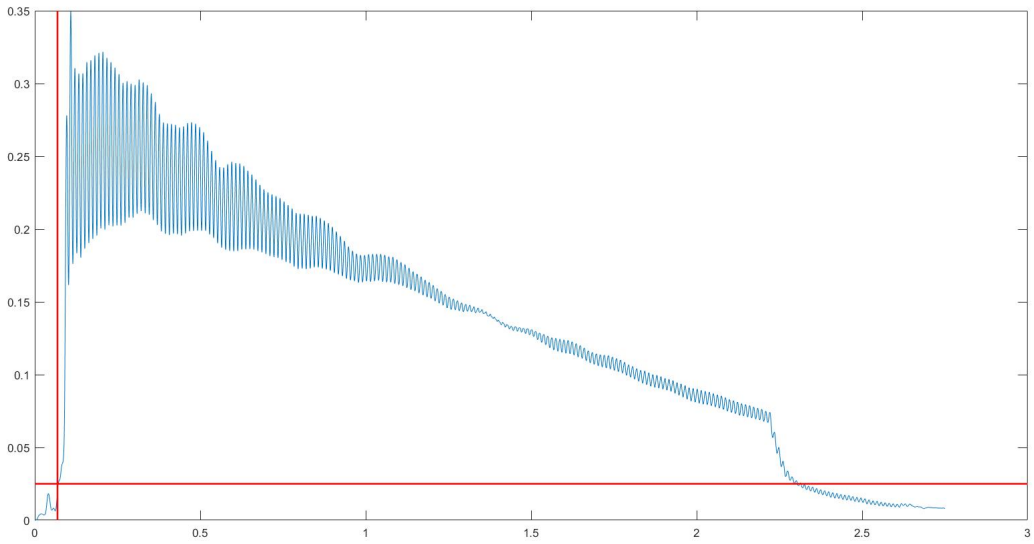
- Envelope Detection: In order to implement the amplitude envelope modulation described above, it is necessary to first characterize the amplitude envelope of the relevant instruments using a form envelope detection. We have implemented an envelope detection method described in [2] which consists of rectification followed by a peaking filter followed by a low pass filter. This envelope detection is an off-line process and the envelope values for the attack and release parts of the envelope have been stored in look-up tables to be used in real-time.

Figure 1: Envelope Detection for Saxophone Sample



- Note Onset Detection: In order to implement the attack part of the amplitude envelope modulation, it is necessary to determine when a note starts, so that the attack envelope can start to be implemented at that time. Following the process outline in [3] we will be implementing a modified form of the envelope detector described above and then using a threshold to determine the time at which a note starts. In off-line testing, this note detection algorithm correctly detected 100% of the notes in our test samples. This algorithm will soon be extended to also determine when a note ends.

Figure 2: Note Onset Detection for Guitar Sample



- Short Time Fourier Transform (STFT) Algorithm: In order to isolate and identify frequencies (ranges) in the input guitar signal, we must perform a STFT on each sample at the specified sampling rate. The trade-off being tested is that the algorithm needs to run fast enough to minimize delay in the output signal, while still having high enough resolution to produce quality sound. Currently we are testing at a sampling rate of 22 kHz (Half of industry standard 44.1 kHz) and adjusting the window length of the STFT in different iterations. Spacing between frequencies in our simulations have been found to be as small as 147 Hz, making this our current constraint on window length.

  – Algorithm 1: (MATLAB code) [4]
  One prospective problem with Algorithm 1 is the potentially large computation time when converting to C code for DSP board due to use of built-in MATLAB functions. If this proves to be an issue, we will test Algorithm 2.

```
function [stft, f, t] = stft_no_comment(x, wlen, hop, nfft, fs)
x = x(:);
xlen = length(x);
win = hamming(wlen, 'periodic');
rown = ceil((1+nfft)/2);
coln = 1+fix((xlen-wlen)/hop);
stft = zeros(rown, coln);
indx = 0;
for col = 1:coln
    xw = x(indx+1:indx+wlen).*win;
    X = fft(xw, nfft);
    stft(:, col) = X(1:rown);
    indx = indx + hop;
end
t = (wlen/2:hop:wlen/2+(coln-1)*hop)/fs;
f = (0:rown-1)*fs/nfft;
end
```

  – Algorithm 2: (to be tested next)
  Tomazic and Znidar [5] describe a STFT algorithm that we will implement once our testing of Algorithm 1 is complete and/or does not achieved optimal results in performance. This is a multi-pole STFT algorithm that should reduce our computation time by removing many, possibly unneeded, numerical operations.

- Overtone Weighting Algorithm: Effective manipulation of overtones relies on the generation of accurate lookup tables for overtone weighting. These weightings need to be determined for the relative amplitude of each guitar overtone compared to its saxophone counterpart. The lookup tables can be generated off-line and hard-coded onto the DSK6713, where they can be applied to the output of each frequency band to generate the manipulated output signal. After taking the FFT of both a guitar and a saxophone playing the same note in Matlab, we can determine the ratio of the magnitudes of each set of overtones, storing an array of multiplier values. As a result, applying these multiplier values to the FFT of a guitar note on the DSK6713 will manipulate the overtone amplitudes to mimic those of a saxophone.

Figure 3: Single-Sided FFT for Guitar Note and Overtone Peak Detection

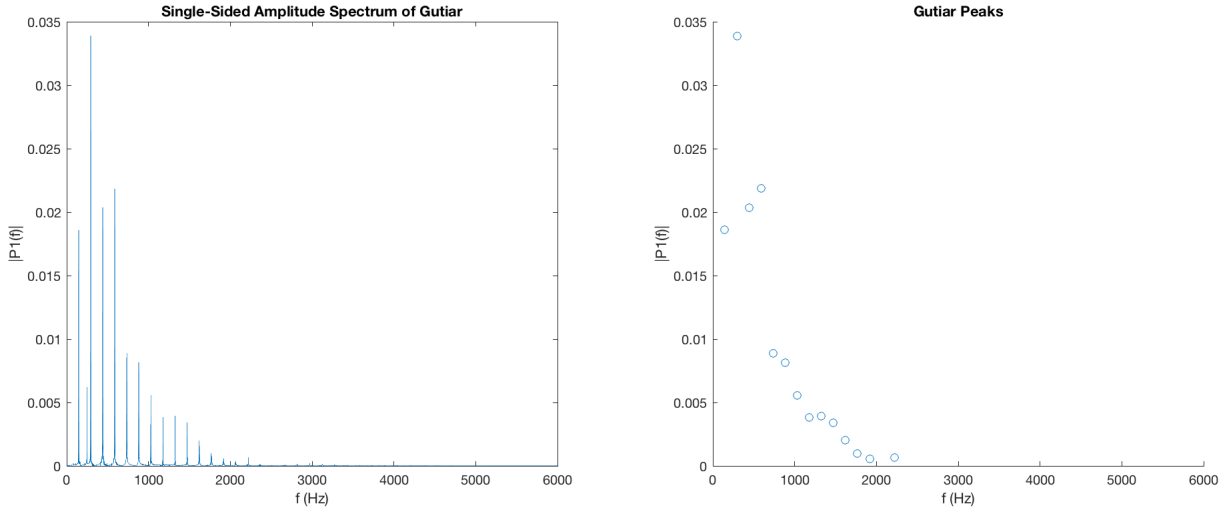
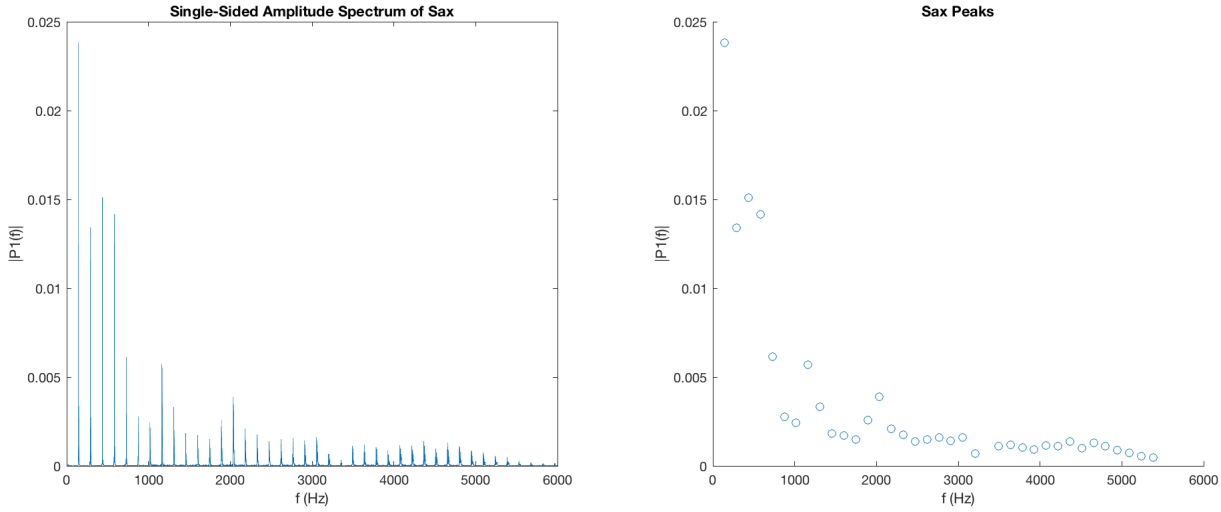


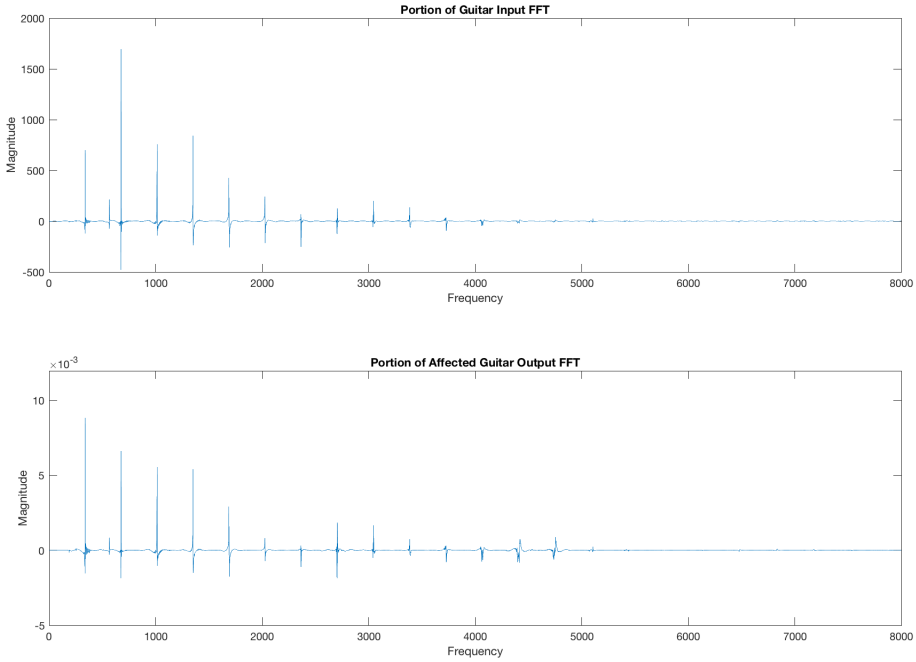
Figure 4: Single-Sided FFT for Saxophone Note and Overtone Peak Detection



- Overtone Weighting Implementation: To implement the overtone weighting algorithm, it is necessary to work off-line with many different audio recordings of guitar and saxophone notes to generate an accurate set of lookup tables. We have recorded three evenly spaced notes per octave on both guitar and saxophone for each octave available on guitar. By running this analysis in Matlab for each note, we can generate an average set of weightings for each octave, which we can then use as lookup tables for our real-time system. We can detect which octave contains the fundamental based on the magnitudes of each frequency band, and the system will access the appropriate lookup table for whichever octave contains the fundamental.

- Overtone Manipulation Proof of Concept: We have generated an off-line test of this overtone weighting algorithm for one note in Matlab. After applying the weightings to a recording of a guitar input signal, we found that the timbre of the guitar changed slightly, but not as much as we had hoped. The next steps to improve this are to fine tune the weighting value algorithm with controls like peak thresholds, which could provide a more comprehensive harmonic structure to manipulate. Additionally, we are considering the idea of making the weighings more intense, taking the overtone structure past that of a true saxophone. This overcompensation may provide the necessary changes to overcome the inherent harmonic structure of the guitar signal, allowing it to sound more like a saxophone.

Figure 5: Comparison of Guitar Input FFT and Manipulated Guitar Output FFT



11.2 Real-time DSP Studies and Preliminary Results

- Amplitude Envelope Modulation: Using the look-up tables determined by the envelope method determined above, as well as the note onset detection determined above, we have implemented the attack part of the amplitude envelope modulation from guitar to saxophone. The attack envelope modulation works as expected, and is computationally light.

11.3 Description of Set-up for Demo

The set-up for our demonstration is fairly straightforward. The will have an un-amplified guitar as input to the DSK6713 with a 3.5mm cable, and another 3.5mm cable as output from the DSK6713 to a set of speakers or a pair of headphones. The user will able to play the guitar and hear the saxophone output of the system in real time. We will also have some prepared guitar samples to use for demonstration in case some people don't want to play the guitar.

11.4 Summary of any Changes in Project Plan

The sub-banding method has changed from using self created linear bandwidth filters, to performing a Short Time Fourier Transform (STFT), which will act similar to a filter bank. We chose to switch to this method because if we were to use a filter bank, then each filter would have a different phase response and group delay, causing issues at each sub-band. The STFT allows us to simply perform the inverse in order to get our signal back successfully. This should prove to be computationally less complicated since we do not need to do any decomposition or reconstruction.

References

[1] http://c6000.spectrumdigital.com/dsk6713/revc/files/6713_dsk_techref.pdf. 2

[2] Cecilia Jarne. Simple empirical algorithm to obtain signal envelope in three steps. *CoRR*, abs/1703.06812, 2017. 3

[3] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, Sept 2005. 4

[4] https://www.mathworks.com/matlabcentral/fileexchange/45197-short-time-fourier-transformation--stft--with-m 4

[5] Saso Tomazic and S Znidar. A fast recursive stft algorithm, 06 1996. 4