Matthew Proano, Ryan Wong (group leader) 12/19/21

Audio Project

Our audio project implements rudimentary code of automatic audio transcription by trying to extract a melody and return its musical notes. This process is far from perfect because onset detection is extremely dependent on the right parameters, which may never exist and is subjective, so it is up to you to roughly set the right parameters for whatever pieces. We gave examples with our own parameters to show that our code, albeit roughly, does work. The pitch detection algorithm is a bit unreliable, and with more complex signals it falls apart, especially with polyphony. The code cannot decide whether one melody or a loud harmony is the melody it should be tracking. As an example for how well our code works, our first example kept up surprisingly well with the melody given the fast tempo and matched the melody almost note for note. However, the chord combined with the shimmering cymbals at the end doesn't have a melody, but the code always assumes there is some definite pitch being played. It then plays a collection of really low notes when it shouldn't. With other pieces, notes would be played when there isn't any noise or a recognizable melody. It is also worth mentioning that if the parameters aren't set correctly, and the more intricate the piece the more impossible it would be to set good parameters, it will fall apart completely. Certain notes wouldn't be caught or notes out of nowhere would be caught, the tempo would be too fast or slow, and sometimes even nothing would be played. One failed example was the Karplus Strong guitar playing a C scale from the course website (https://www.cs.bu.edu/fac/snyder/cs583/AudioSamples/KS.guitar.C.scale.wav). The result of this was a transcription of what seemed like completely random, very high pitched

notes. We think this may have something to do with the fact that it was created artificially using Karplus Strong and the overtones are more random and prevalent. We are not entirely sure, but it clearly seems to require a better pitch detection algorithm, which had decent success on piano audio samples. Another example included in our code is The Lick, and for some reason the pitch detection algorithm is not properly detecting f_0 and is returning a different harmonic.

The thought process for the code was to be able to feed in an audio signal to a function, and return the musical data that was retrieved. To do this we utilized the amplitude onset and pitch detection algorithms from the past homeworks. Originally, we wanted a way to automatically fine-tune these parameters, but the reality is that this is only possible through trial and error to find the correct onsets. In our testing we found that our algorithm fails when more complex signals are tried. So we tried implementing a moving average filter to help emphasize the melody and deemphasize the harmony and everything else, but realized it didn't really do much so we got rid of that. If we had time to improve this code, we would implement the steps laid out in Justin Salamon's website: https://www.justinsalamon.com/melody-extraction.html. Salamon essentially uses the de-facto method of melody extraction instead of the simpler mixture of onset detection and pitch detection. Instead of using merely volume throughout time as a means of splitting up the music to then identify as notes, their algorithm measures the entire frequency spectrum instantaneously via DFT. Then they identify the predominant noise, often the melody, with a salience function that keeps track of both the fundamental frequencies and the harmonic series and other improvements like "contour creation". Using the whole frequency spectrum essentially bypasses our problems by extracting the melody almost directly.