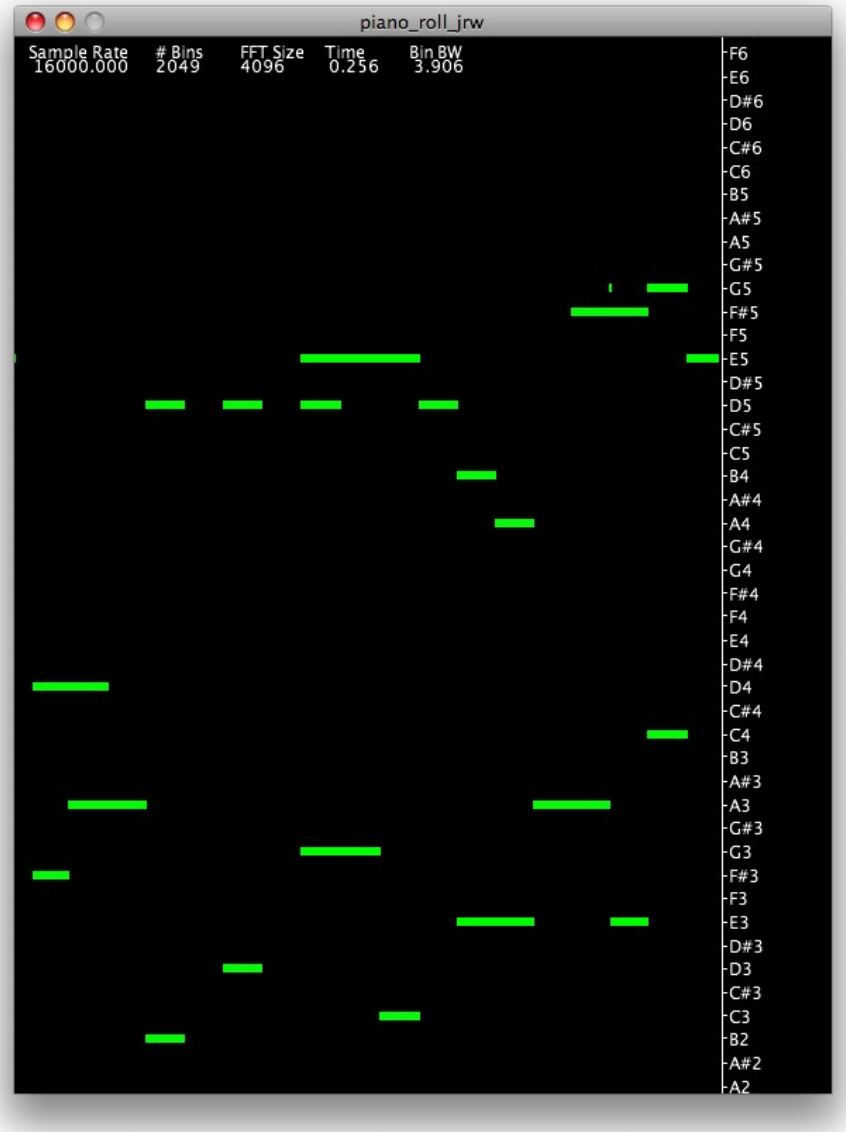
Piano Roll Program in Processing

Jonathan Ward



Methodology

To extract the fundamental notes played by the Bach piano piece, my program only looks at the FFT frequency bins that correspond to a musical note. This is not the best solution, however because of the logarithmic nature of musical notes. At lower frequencies, musical notes are much closer together than at higher frequencies. If a fixed frequency bin bandwidth is used (as in a normal FFT), a higher frequency bin will be more selective than a lower frequency been. The higher frequency bins have a higher effective Q.

While this scheme is not the best, I found it to work pretty well in practice. I tried making a

version that had the bin bandwidth increase with frequency, but ran into some trouble. The minim library includes a handy function called logAverages() that seems like it would really help with this. However, I found it difficult to set up to have the logarithmic bins be centered around musical notes. I eventually abandoned this effort because it did not seem to be the biggest stumbling block to achieve good performance.

To accurately capture each musical note, I used a two step filtering process. First, frequency bins must meet a minimum threshold before they can be suspected of holding a note. After this culling process, I scrutinize each suspected note to eliminate harmonics. For each suspected note, I look at the energy in the frequency bins of all of its harmonics. If the harmonic is not a substantial fraction of the fundamental, it is removed. After going through this process, I should be left with only the true notes being played.

To ease the requirements of the note labeling code, I mapped rounded integer note frequencies to their corresponding note names using a hashtable. Using a hashtable effectively gives me a transfer function that easily extracts a note name, given its frequency.

Room for Improvement

I encountered a few pitfalls on the assignment that I was unable to work around. To achieve the resolution necessary to accurately resolve low frequency notes, the program needs a large FFT window. The problem with a large FFT window is that it reduces the timing resolution. With this particular Bach piece, notes are sometimes played very quickly. In some cases, my program incorrectly lengthens the note. In others, it counts a sequence of rapidly repeating notes as one note.

I settled on an FFT window size of 4096 samples. This size yields an FFT bin bandwidth of around 4 Hz. 4096 samples at the 16 kHz sampling frequency of the midi file occupies about a quarter of a second. Because of the piano player's fast key twiddling, some of the notes seem to be played faster than that. This is a problem.

The large FFT window size is needed to differentiate between a B2 and a C3, which are only around 7.5 Hz away from eachother. If I reduced the window size to get more temporal accuracy, my FFT bandwidth jumps to 8 Hz, which blurs out the lower notes too much. If I lowered the FFT window size to yield better time resolution, the algorithm lost the ability to discern the lower frequency notes.

One way to fix this problem would be to have a high FFT window size for frequency accuracy, but not use a new block of data each time. If you slide the FFT window forward a little bit each time, you include data that has already been analyzed along with new data. This increases your temporal resolution while maintaining frequency resolution.

Unfortunately, I could not get this to work with the minim library. It seems to only be able to grab a whole new FFT buffer each time. It is probably possible to hack this to work by continuously rewinding the song for every FFT, but I don't think it is very easy to do this in realtime.

I also noticed that some of my recognized notes die away too suddenly. I can still hear the note being played, but it is no longer visible on the piano roll. One way to fix this would be to give a recognized note some "credit." If a strong fundamental tone is identified, it should still be identified as a note even if it falls below the minimum threshold a few seconds later. There could be a threshold below which an identified note has to fall for it to be disregarded.

Another challenge I discovered has to do with the percussive nature of the piano. Each note is struck with a sharp impulse that tends to create a lot of broadband energy centered around the fundamental. This energy can spill over into neighboring frequency bins, giving erroneous readings. Playing with the minimum threshold to eliminate this tends to filter out too many legitimate notes. One strategy to solve this is to prune the neighboring notes of a likely fundamental. I already prune the harmonics generated from the steady state piano string response, but I should also prune the broadband energy created from the transient response as well.

Conclusion

While I created a good piano roll interface for viewing the incoming musical notes, my identification of those notes is not very strong. It is often easy to get lost in the minutia of first solution one comes up with, painstakingly shoehorning it to fit. Knowing when to back up and look at the big picture and analyze the fit is hard, but necessary to evolve the solution to be the best it can be.