Noise Resilient Piano Homophonies Recognition

Abhipray Sahoo, Eric Kang, Chenxi Liu, John Yan

Acknowledgements to Richard Baraniuk, Eva Dyer, and Christoph Studer

Introduction

Automatic Music Transcription (i.e notating a piece of music automatically) has been a hot topic of research in signal processing for decades now. Fundamentally, it requires the detection of note intervals and their pitches. While this problem has been solved for monophony (single line of melody), it is still largely unsolved for polyphony (multiple lines of independent melody). Our project aims at accurate and robust detection of homophonies (single melody line with chord line). We explored several existing implementations , augmented and combined different approaches. Our constraints were: single instrument (Piano), beats per minute less than 180 and minimal reverberation.

Results

We tested our implementation on a variety of piano songs, created digitally as well as recorded from a Grand Piano. Our metric for accuracy is defined to be:

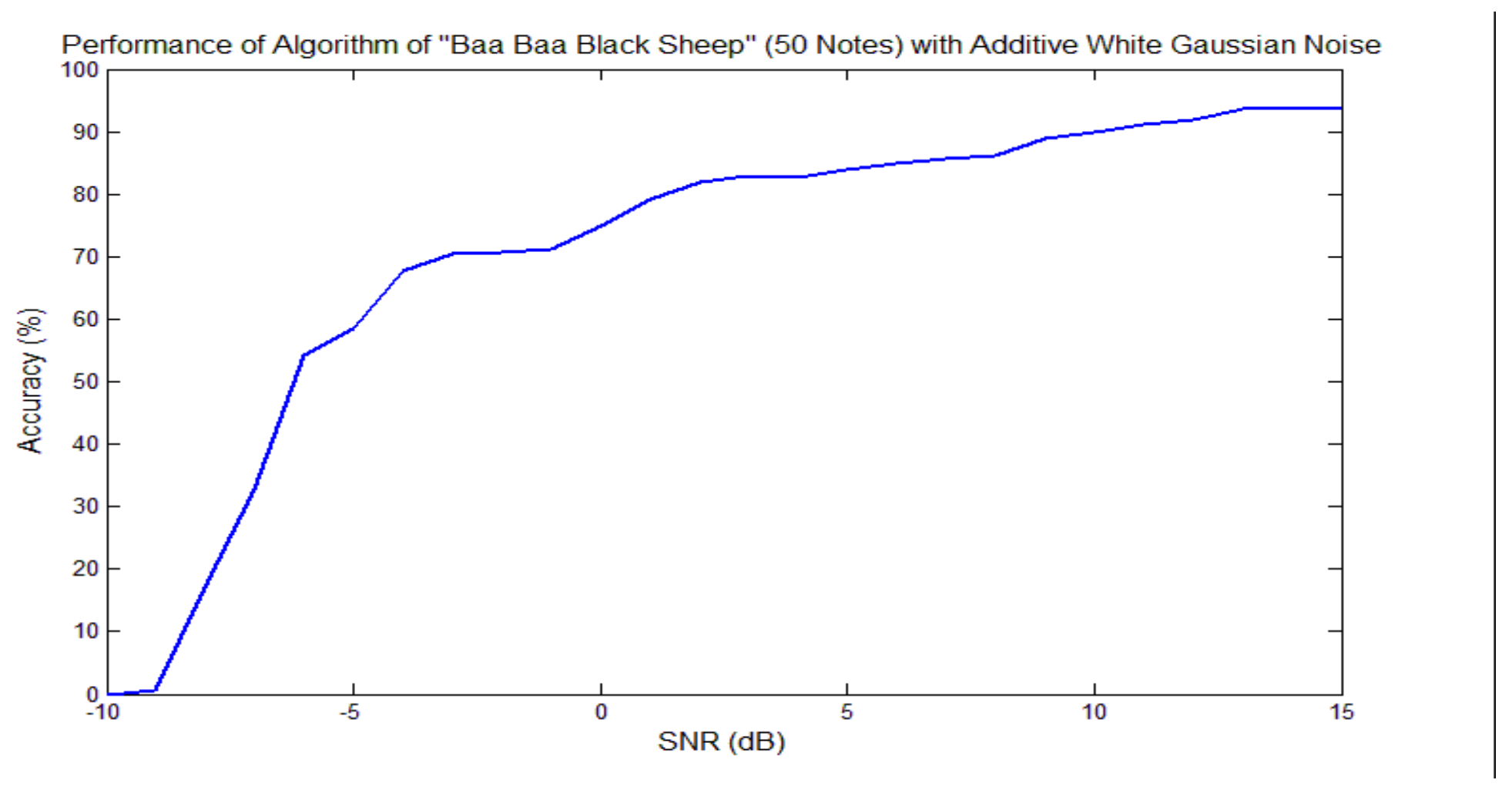
$$Accuracy = 100 * (N - edit\_distance(x, y))/N$$

Where x is the correct string of notes/chords and y is the one created by our implementation; N is length of x; edit_distance gives the number of changes (insertions, substitutions or deletions) to go from string x to y.

The average accuracy on testing eight different songs was:

- 92% on songs with single melody line
- 86% on songs with single chord line
- 82% on songs with homophonies

One of the features of our implementation is noise-resiliency. With additive Gaussian noise, the performance is maximum above SNR of 5 dB.



References

Fujishima, Takuya, “Realtime Chord Recognition of Musical Sound: a System Using Common Lisp Music” Stanford University.

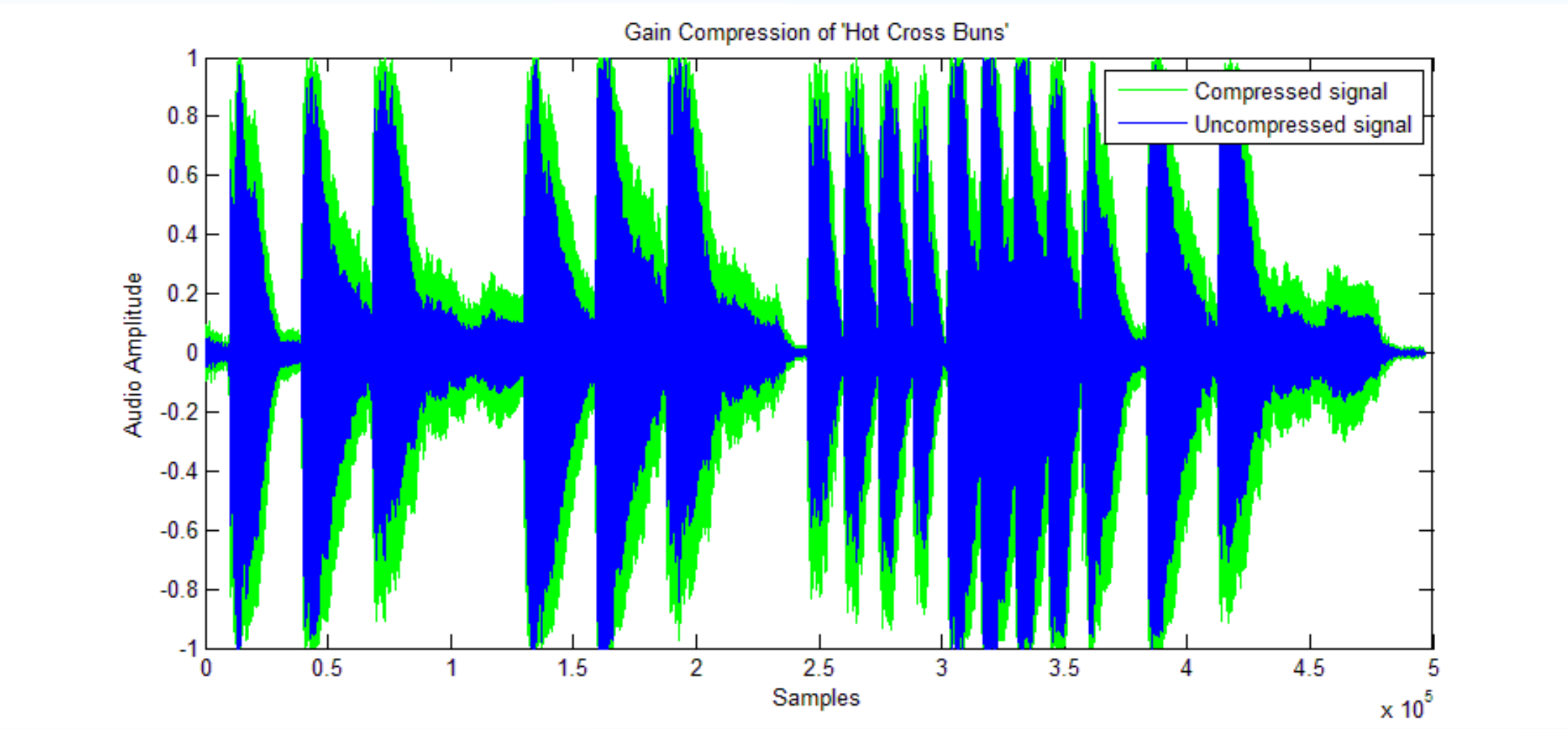
Joan Serra, Emilia Gomez, Perfecto Herrera and Xavier Serra, “Chroma Binary Similarity and Local Alignment Applied to Cover Song Identification”

Implementation

Gain Compression

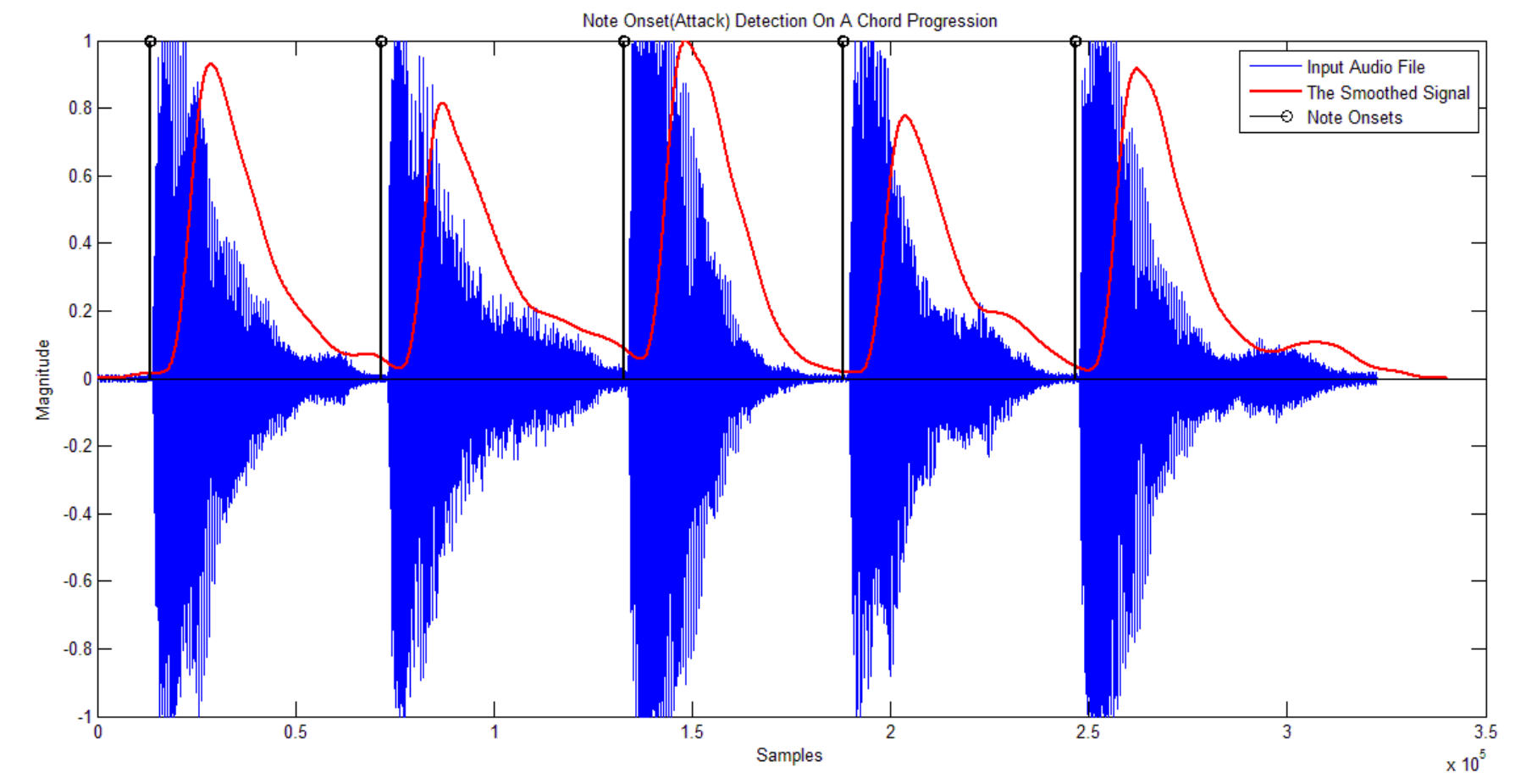
First, we prepare the audio signal for analysis by doing a gain compression. This flattens large peaks in our signal while amplifying the rest. It therefore allows us to "clean up" the signal especially when the pianist plays certain notes softly and other notes loudly.

$$y[n] = 1 - (1 - |x[n]|)^2 * sign(x[n])$$



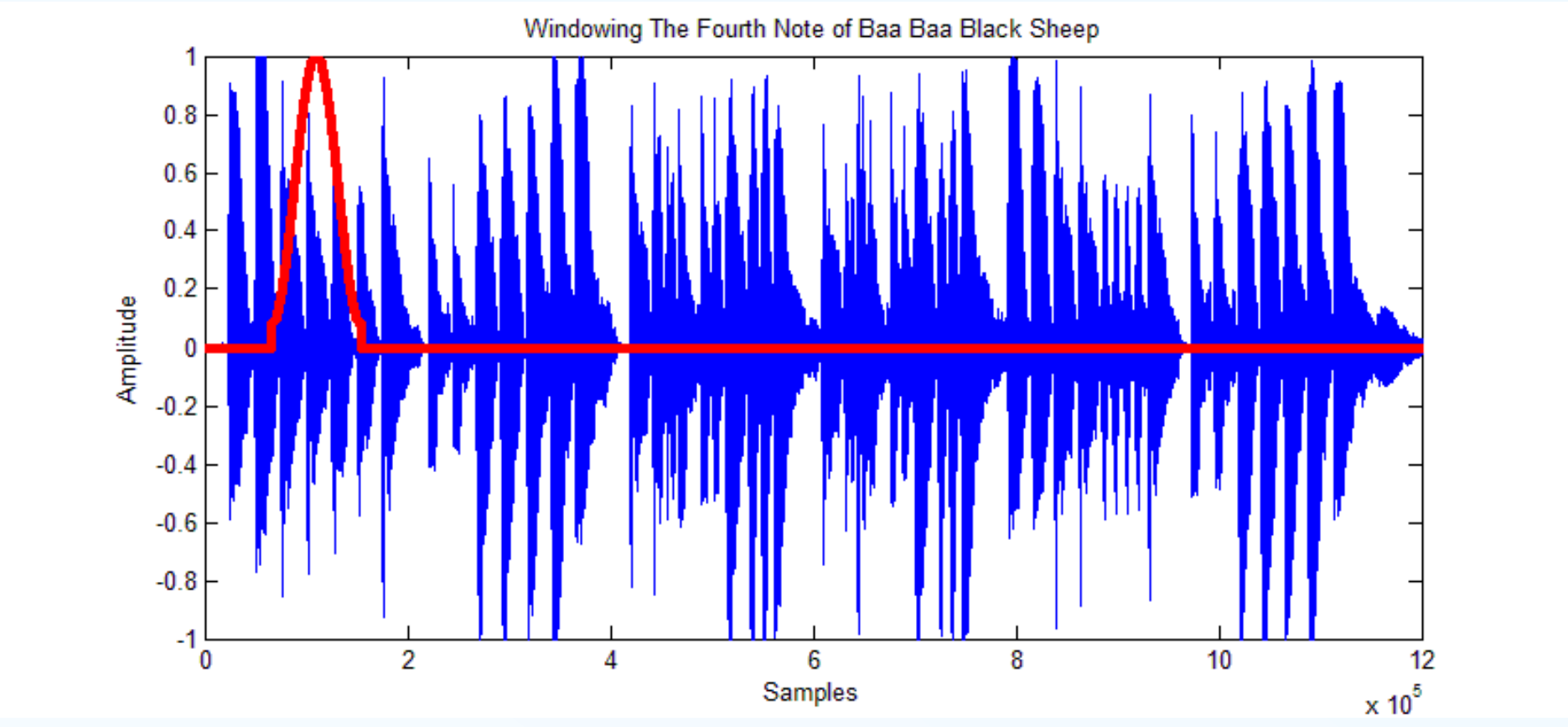
Note Onset Detection

To record the onset of each note in the song, we search for sharp increases in amplitude of the signal in the time domain. We achieve this accurately by smoothing out the signal . Our algorithm utilizes a moving average (low-pass) filter– a boxcar filter convolved with itself three times. Filtering gets rid of high frequency oscillations and leaves behind only the peaks which represent the start of notes. We find the locations of these peaks to find note onsets. Because the convolution of the signal with our filter shifts all peaks by the length of the filter, we add a correction term for the actual location. Once we have note onsets, we can also generate the bpm (beats per minute) of the song.



Fourier Analysis Of Notes

To extract out the note from the song, we window the song signal with a hamming window. We take a Fast Fourier Transform (FFT) of the note signal to study it in frequency domain. We detect ten candidate frequencies with the greatest power along with their corresponding power. This information is passed into the HPCP algorithm for further processing.

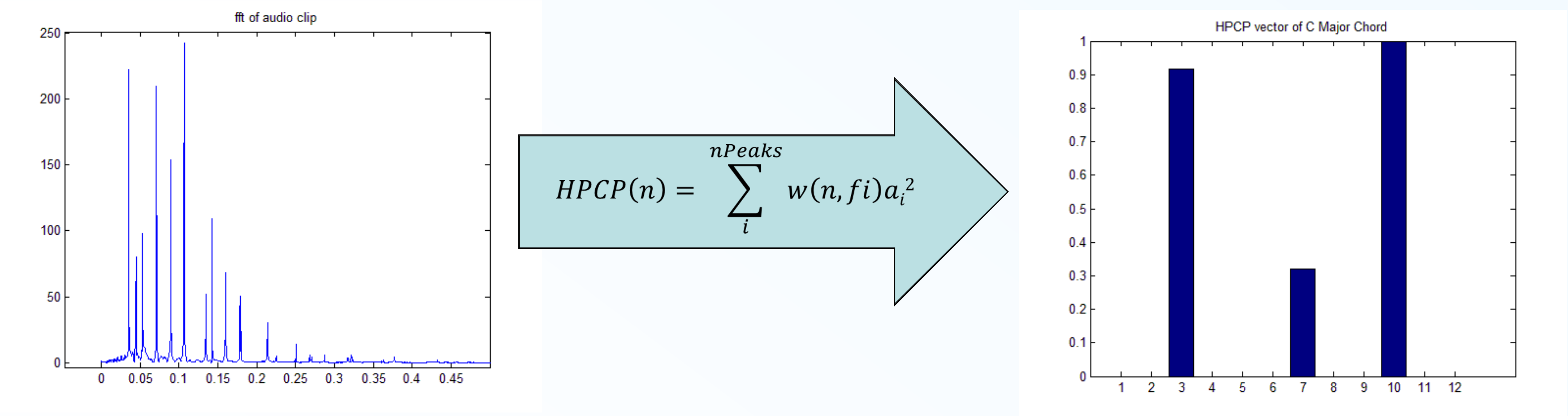


Each peak we detect in the Fourier transform corresponds to the harmonics of whatever piano key was being pressed. By finding the largest peaks in the frequency domain, we are able to much better “guess” exactly what note was played.

Pitch Classification

The Harmonic Pitch Class Profile (HPCP) algorithm was first proposed by Fujishima from Stanford in the context of a chord recognition system. The HPCP algorithm produces a vector of length 12, representing the 12 notes in an octave. The vector is normalized to representing the likelihood that the corresponding note is actually in the audio.

The inputs required are the peaks in the frequency domain, represented by the pair (f_i, a_i), where f_i is the location of the peak frequency and a_i is the corresponding amplitude. Each of the 12 pitch classes is then compared with every recorded peak to calculate a certain “correlation” determined by a $\cos^2$ weighting function and amplitude a_i. By analyzing the coefficients(which represent the correlations), we are able to decide which note(s) are actually present in the audio clip.



$$HPCP(n) = \sum_{i}^{nPeaks} w(n, fi)a_i^2$$

Above is an HPCP example of analyzing a C Major chord (C, E, and G). The HPCP algorithm returns a vector of length 12, each representing the relative intensity of that pitch class, with 12 representing A and 1 representing A#. We can see clearly from the figure on the right that C(3), E(7), and G(10) are the most significant results.