Chord Recognition From DFTs of Down Sampled Audio Signals

Ethan Mandel
Electrical & Computer Engineering
Northeastern University
Boston, MA
mandel.e@northeastern.edu

***Abstract*— This paper presents a chord recognition method through analysis of a Discrete Fourier Transform (DFT) as well as a method for improving the calculation speed of said DFT. Research was conducted using a DE1-SoC and its hardware limitations inspired methods for reducing the number of calculations required to calculate a DFT. Down sampling the signal is investigated as a method to improve processing times while still maintaining enough meaningful data for accurate analysis. A novel method of chord identification from a DFT is presented involving the utilization of overtones and knowledge of western music chord structure. Down sampling of an audio signal may expedite chord recognition as well as other forms of musical analysis. This research was evaluated using individual chords played on an electric piano connected to a DE1-SoC.**

***Keywords*—** *chord recognition, DFT, DE1-SoC, music signal processing*

I. Introduction

Chords are one of the key elements of Western music. A common chord structure in music is a three-note chord known as a triad. The primary chords used in western music are the major and minor triad. The scope of this investigation covers these types of chords, but the methods described below can be extended into different types of triads as well as chords with greater or fewer notes.

Automatic chord recognition is useful in analyzing chord progressions within music as well as useful in automatic transcription of music. Many sophisticated methods of chord recognition exist. One of the most common methods is the use of a chroma vector which allows for real-time chord recognition using a DFT [1]. Other approaches focus on machine learning and Hidden Markov Models with chord labels as hidden values [2]. Additionally, there are other methods that focus on the use of the doubly nested circle of fifths where certain major and minor chord vectors are positioned near each other based on similarity so they can be analyzed with greater ease [3]. Many approaches use combinations of these elements and can identify chords within music with astounding accuracy.

This approach is most similar to the DFT chroma vector method, in which a DFT is used to analyze the power spectrum of the signal in the frequency domain. This paper is an investigation into both chord recognition and ways to reduce the number of calculations required to have a meaningful representation of a signal in the frequency domain. All chord identification and data collection were done using the DE1-SoC so references to processing speed refer to the DE1-SoC's processing speed specifically.

This paper is organized as follows. Section II presents the method for converting raw audio data into a frequency domain power spectrum, identifying notes from frequencies, and two methods for determining chords using the data generated from the first two algorithms. Section III provides experimental results for DE1-SoC processing speeds, as well as chord recognition accuracy using recordings of individual chords. Section IV concludes the paper and proposes future extensions of this research.

II. Algorithm Implementation

A. Algorithm Introductions

Three algorithms work in series to turn raw audio data into an identified chord. The first algorithm, audio to maximum frequencies, determines the power spectrum of the audio input and determines the 15 largest maxima as the 15 potential chord frequencies. These 15 frequencies are then converted into note values through the note identification algorithm. These note values are then analyzed and reduced to three notes through the chord identification algorithm. These three notes determine the chord. An improvement to the chord identification algorithm is also presented that improves accuracy.

B. Audio to Maximum Frequencies

To begin the conversion of raw audio data to the frequency domain, the number of audio data points and the sampling frequency are recorded and designated L and fs respectively. Using fs, a set of frequency values used in the DFT are generated. These frequencies are evenly spaced values between 0 and half of fs to prevent aliasing [4]. Nyquist sampling as shown below is the rate at which data need to be sampled at to prevent aliasing.

$$f_s = 2f_{max} \tag{1}$$

The Fourier Coefficients of the sound files are determined by taking the sum of the signal, x,multiplied by a complex sine wave for each data point in the signal as shown below.

$$a_k = \sum_{n=1}^{L} x(n) e^{-2\pi j(k-1)\left(\frac{n}{L}\right)} \tag{2}$$

The power at each frequency is determined by

$$P_k = \left(\frac{a_k}{L}\right)^2 \tag{3}$$

The method above shows the generation of all Fourier Coefficients for a given signal. Due to the processing limitations of the DE1-SoC, completion of this sum is processor intensive and time consuming. By calculating the Fourier Coefficients at wider intervals determined by a factor, v, processing time is reduced drastically while still maintaining the key components of the initial signal required to identify a chord. This reduction of meaningful Fourier Coefficients is expressed as

$$a_k = \begin{cases} 0, & k \,\%\, v \neq 0 \\ \sum_{n=1}^{L} x(n) e^{-2\pi j(k-1)\left(\frac{n}{L}\right)}, & k \,\%\, v = 0 \end{cases} \quad (4)$$

An example of this signal reduction is shown in the figures below.

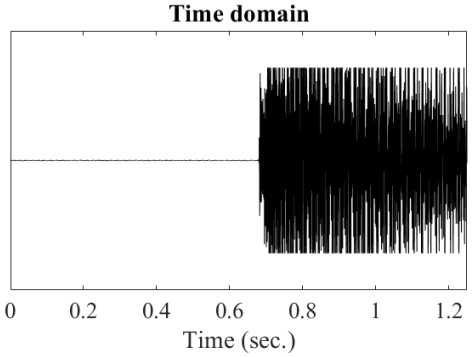

Fig 1. Time domain representation of signal

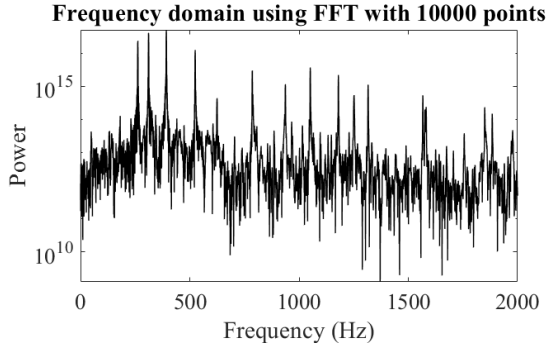

Fig 2. Frequency domain representation of signal with no Fourier Coefficient reduction (No down sampling)

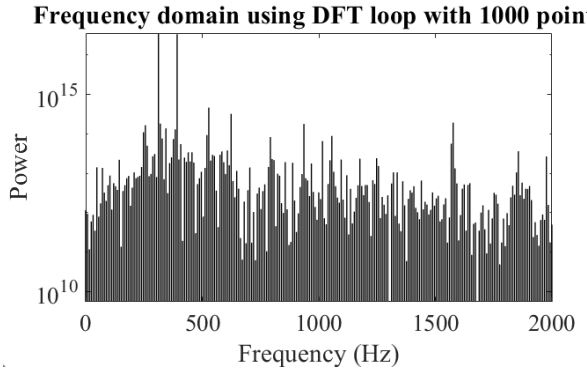

Fig 3. Stem plot of a frequency domain representation of signal with Fourier Coefficient reduction factor of 10 (1 in 10 Fourier Coefficients preserved)

The frequencies with the 15 largest power values are recorded as potential note frequencies and utilized by the next algorithm to identify which note corresponds to each frequency value. All frequencies below 27 Hz and above 4186 Hz were discarded as no note played on a piano can generate tones outside of this range although overtones may be outside of this range [5]. The list of frequencies is then sorted from least to greatest frequency for further evaluation.

C. Note Identification

Any note can be identified as a factor of 440 Hz which is the pitch standard. Western music divides sound in to 12 semitones which repeat at octave intervals (A, B-flat, B…) [5]. The formula used to determine the frequency of every note on a piano is expressed as

$$2^{\frac{i+(12*O)-49}{12}} \quad (5)$$

Where i representing the number of semitones from A=440Hz and O representing the octave number

The values generated by this formula represent notes that are perfectly in tune. Tuning, temperature, and other environmental factors affect the frequency emitted by the piano. Notes are infrequently played perfectly in tune; therefore, it is necessary to determine what note each frequency is by using equation (5). Each recorded frequency is paired to the closest frequency that satisfies equation (5). Although notes played by the piano may be slightly flat or sharp in their pitch relative to perfectly in tune notes, the method devised above correctly translates frequencies into their corresponding notes.

D. Chord Identification

With each frequency assigned a note value, a chord can be discerned from a set of frequencies. The initial algorithm for chord identification is to take the first 3 notes identified after the frequencies are sorted and set them as the three chord tones. These three notes are then compared against a list containing all major and minor chords and the notes within them. From this the chord is identified. This method is far less effective than the improved algorithm. The difference in these algorithms' performances can be seen in the results section.

E. Improvement to Initial Chord Identification Algorithm

The improved algorithm to determine the chord from a list of notes prioritizes which notes are chord tones based on two conditions. The first condition is note appearance, the number of times a note, regardless of octave, is present in the list. The second condition is frequency. Notes that make their first appearance at lower frequencies are given higher placement in the list. Appearance takes priority over note frequency but if two notes appear the same number of times in the list the note that appears with the lower frequency is placed higher.

Note appearance is taken into priority due to the presence of overtones that are generated whenever a note is played. When a note is played the fundamental frequency of the string can be heard with the most clarity and intensity, but integer multiples of this frequency can also be heard at a lesser intensity. [6] This

is due to the nature of standing waves which are fundamental in the design of both wind and string instruments. The overtones act as a check to confirm that a specific frequency is present. The more overtones that are present of a single note the greater probability that the note is correctly identified.

Note frequency is a secondary priority as chords in western music are built by building musical thirds on top of a single note [5]. The bottom note defines the chord. By prioritizing lower frequencies, there is a higher probability that the bottom note is identified as the first note of the chord.

Just as in the initial algorithm, the 3 notes are then compared against the notes of all major and minor chords to determine what chord is present in the audio data.

III. Results

A. Evaluation Method

Two separate algorithms are used to determine a chord from audio data: changing the signal from the time domain to the frequency domain with a DFT, and chord identification. The two metric that determine the efficacy of these algorithms are DFT calculation time on the DE1-SoC and chord identification accuracy at various DFT reduction factors.

B. DFT Reduction Algorithm

Increasing the DFT reduction factor yields fewer Fourier Coefficients thereby destroying parts of the initial signal. Without concern for the integrity of the signal, processing time is used as a metric to determine how quickly Fourier Coefficients are calculated. The times shown below are the averages of 100 trials at each DFT reduction factor.

Table 1: DFT Reduction Factor impact on processing time

DFT Reduction Factor	Processing time (s)
1 (no reduction)	177.48
2	92.63
3	61.72
4	46.35
5	37.07
10	17.68
15	11.8
20	8.85
30	5.91
40	4.41
50	3.53
100	1.76

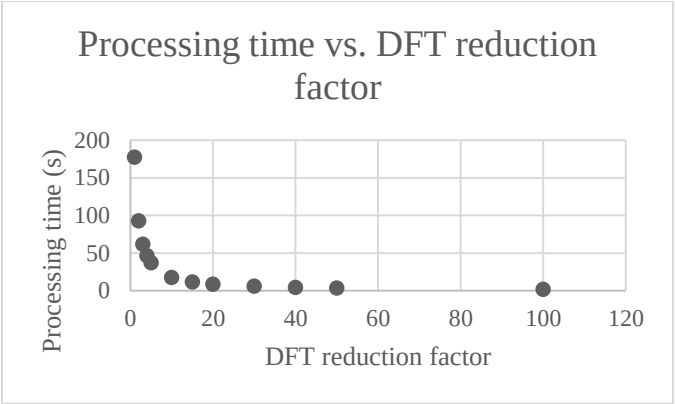

Figure 4: Processing time vs. DFT reduction factor

Though signal quality is important for identifying chords, a DFT with a reduction factor of 1 generates far more Fourier coefficients than necessary. By reducing the number of coefficients generated, processing time can be reduced without a drop in chord identification accuracy up to a point.

C. Chord Identification Algorithm

Two iterations of the chord identification algorithm were created for this investigation. These algorithms are tested at various DFT reduction factor values to also determine how DFT reduction affects identification accuracy. The first algorithm simply takes the first 3 unique notes in the list and checks them against all chords. The efficacy of this method is shown in the table below. These percentages were calculated from 240 trials, 10 different recordings of all 24 chords.

Table 2: DFT reduction factor on chord identification accuracy of initial algorithm

DFT Reduction Factor	Percentage of chords correctly identified (Original Algorithm)
1	100.0%
2	100.0%
3	100.0%
4	95.8%
5	91.8%
10	79.2%
15	45.8%
20	12.5%
30	<1%
40	<1%
50	<1%
100	<1%

The improved algorithm, which prioritizes note appearance first and lower note frequency second, is tested in the same manner as the algorithm above using the same recordings.

Table 3: DFT reduction factor on chord identification accuracy of improved algorithm

DFT Reduction Factor	Percentage of chords correctly identified (Improved)
1	100.0%
2	100.0%
3	100.0%
4	95.8%
5	95.8%
10	91.7%
15	58.2%
20	33.3%
30	20.8%
40	8.7%
50	<1%
100	<1%

Both the initial and improved algorithms display the same performance at low DFT reduction factors. With a reduction factor of 10 or greater the drop off in performance becomes apparent. DFT reduction factors above 20 fail to identify chords 99% percent of the time with the initial chord recognition algorithm. By contrast, the improved algorithm was able to identify chords with some level of accuracy up to a DFT reduction factor of 40. At all levels of DFT reduction the improved algorithm outperformed the initial chord recognition approach with the most substantial improvements appearing at DFT reduction factors of 10 and 15.

IV. Conclusion

In this paper a method for reducing DFT calculation times as well a method for identification of a chord from a list of notes has been presented. The down sampling of the signal has proven to be highly effective in maintaining enough important frequency information at low DFT reduction factors to properly identify chords with near perfect accuracy. At higher DFT reduction factors too much of the signal is destroyed for any meaningful analysis to be performed on the signal. While holding the DFT reduction value constant, the improved chord identification algorithm proves to noticeably outperform the initial chord identification algorithm. A combination of reducing the number of Fourier Coefficients calculated and implementation of the improved chord identification algorithm allows for quick identification of chords from audio data.

This method of chord identification proved highly effective at low DFT reduction factors, but at the cost of processing time. This method is unable to handle real-time analysis of chords in music due to processing limitations. Improved DFT calculation algorithms would reduce calculation time without destruction of the signal to the point at which analysis of the signal is meaningless.

This method of down sampling for musical analysis proved to be highly effective at low DFT reduction factors. Implementation of the algorithms presented above in more sophisticated chord identification algorithms that utilize chroma vectors may allow for significantly reduced processing time for real-time chord recognition systems.

V. References

[1] T.Fujishima: “Real-time chord recognition of musical sound: A system using common lisp music,” Proc. ICMC, pp. 464-467, Oct. 1999

[2] K. Lee and M. Slaney, "Acoustic Chord Transcription and Key Extraction From Audio Using Key-Dependent HMMs Trained on Synthesized Audio," in IEEE Transactions on Audio, Speech, and Language Processing, vol. 16, no. 2, pp. 291-301, Feb. 2008, doi: 10.1109/TASL.2007.914399.

[3] A. Uemura and J. Katto, "Chord recognition using Doubly Nested Circle of Fifths," 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 2012, pp. 449-452, doi: 10.1109/ICASSP.2012.6287913

[4] A. V. Oppenheim and A. S. Willsky, Signals and Systems. Englewood Cliffs, NJ: Prentice-Hall, 1983J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892.

[5] Loy, Gareth, Musimathics: The Mathematical Foundations of Music, Cambridge, MA: MIT Press, 2006.

[6] L. Oudre, Y. Grenier and C. Févotte, "Chord recognition using measures of fit, chord templates and filtering methods," *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, USA, 2009, pp. 9-12, doi: 10.1109/ASPAA.2009.5346546.