SYMPHONY - AUDIO PROCESSING

Kartik Vyas (B18CSE020)
Devin Garg (B18CSE011)
Aman Gupta (B18EE003)
Anmol Gupta (B18EE060)
Archit Dwivedi (B18BB005)

Abstract

Music – "an art form whose medium is sound, organized in time." is much more than that in the modern world. Now, to engineers, what's fascinating is the analysis of these audio signals (in addition to the music of course!). People who are currently learning music face the problem of getting a self-evaluation of their performance. To solve this issue, we enter into music processing (or audio-signal processing) in which we employ the tools of **Discrete Fourier Transform** (for frequency comparison) and **Short Time Fourier Transform** (for time delay analysis). Additionally, statistical correlation has also been used to get the similarity between data sets of frequencies. Over the course of this project, some ideas did not entirely work out and there were shortcomings which have been discussed in the subsequent sections.

1. Introduction

1.1 Motivation behind the project:

Do you get inspired to pick up any instrument after listening to moving melodies, upbeat baselines and pleasant harmonies, but you soon give up when you are nowhere close to the original composition just because your performance doesn't sound similar and you don't have any tool to measure your performance.

We began our journey to create Symphony a Signal processing tool which compares and quantize your performance to give you a good measure of similarity between your cover and original composition so that you can work on improving your score and are motivated enough to achieve perfection.

1.2 The problem statement:

The purpose is to create a unique and an efficient way to aid the composers by providing them a tool through which they can test their compositions against greatest compositions of all time. We would give a criteria based on similarity extent that can be judged with the help of Fourier Transform and its derivatives. We aim to create a dataset containing great music, from which the user can select the track or can even enter two inputs and the two tracks will be used for judging. In return, the user would get percentage based on one to one frequency matching and two images based on Short Time Fourier Transform depicting the significance of the frequencies being played on a particular time and thus indicating similarity between the two.

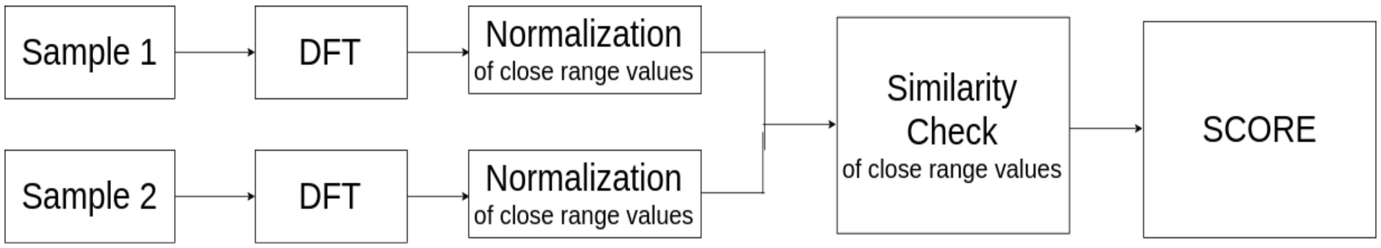
1.3 Extracting Audio File Information:

When processing audio signals, the primary base of knowledge required, is with regard to the composition of an audio file. On extraction of data from an audio file (using audioread() function of MATLAB© or scipy.io.wavfile.read() function of Python) we receive two quantities – *first*, the **sampling frequency** of the audio data and *second*, the **sampled data**.

Standard sampling rate ranges from 8kHz to 48kHz, the most common being 44.1 kHz. A sampling rate of 44.1 kHz would mean 44,100 samples are contained in every 1 second of audio. Sampling rate of 44.1 kHz allows maximum audio frequency of 22.05 kHz and since human ear can detect audio of only upto 20kHz frequency, a higher sampling rate of, say, 48 kHz wouldn't cause any appreciable improvement in audio quality.

2. **Audio Similarity Comparison (including Spectrogram Visualization):**
There are several mathematical constructs that have been used in this project. We have explored and chosen those algorithms that led us to our purpose of studying audio signals that evolves overtime. Now we will briefly touch upon those concepts one by one.

2.1 Workflow:


Sample 1
DFT
Normalization of close range values
Sample 2
DFT
Normalization of close range values
Similarity Check of close range values
SCORE

2.2 Fourier Transform:

We have used *Cooley and Tukey*[1] algorithm for fast fourier transform. The perk that we get in turn for using FFT algorithm instead of simple brute force algorithm is faster computational speed.
This algorithm was considered a great breakthrough in digital signal processing when it was invented back in 1965. That is because of accomplished optimization of conventional algorithm from O(n^2) to O(nlog(n)).
The below mentioned is the basic fourier transform expression.

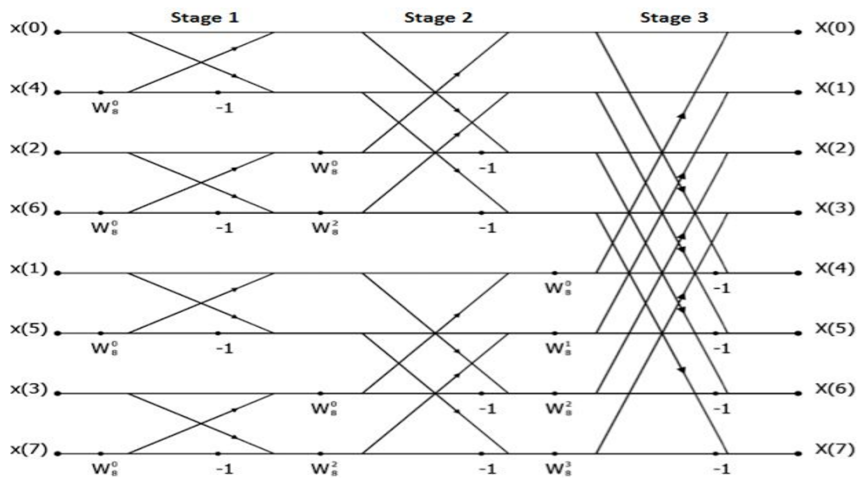
$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} nk},$$

But this has an implementation drawback. Nested looping of 'n' over 'K' would give us a time complexity of O(n*n).
Now we have taken the even and odd terms separately in order to simplify the expression.

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N}(2m)k} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N}(2m+1)k}$$

These two DFT are obtained for (N/2) domain.

$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of even-indexed part of } x_n} + e^{-\frac{2\pi i}{N} k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of odd-indexed part of } x_n} = E_k + e^{-\frac{2\pi i}{N} k} O_k.$$



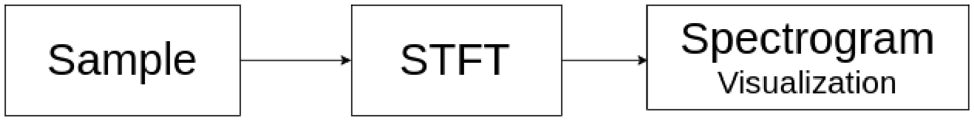
Now for deeper understanding let us take the help of an example, if N=8

N	**1000**	**10^6**	**10^9**
Complexity from Fourier Transform(N*N)	10^6	10^{12}	10^{18}
Complexity from Fast Fourier Transform(Cooley-Tukey)(N*logN)	10^4	$20*(10^6)$	$30*(10^9)$

Assuming O(1) operation takes 1 nanosecond, if N= 10^9, the time complexity from Fourier Transform is 10^18, which would take **31.2 years**. It is interesting to note that for the same N , the complexity from Fast Fourier Transform(Cooley-Tukey) is just **30 seconds**.

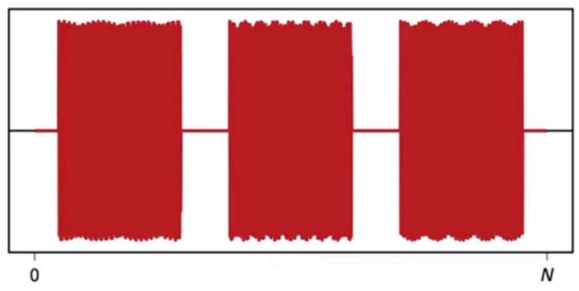
2.3 Normalization of close-range values:
While using statistical correlation to get the similarity between the frequencies played by the user and the frequencies in the original composition, to get the algorithm to work in a reasonable amount of time, we would need to limit the number of frequencies taken into consideration. While doing this, many nearby frequencies would be present which need to be eliminated to get a better data set of frequencies. In doing that, we need to determine the sensitivity factor to set the minimum gap between two contributing frequencies after which we would consider them separately.

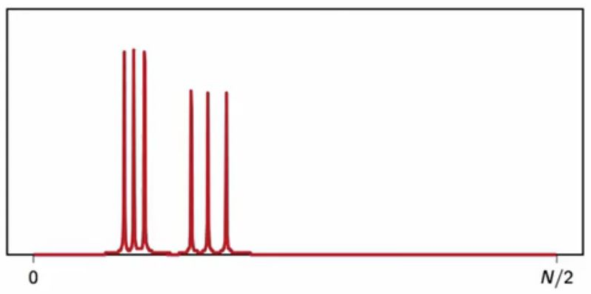
2.4 Short Time Fourier Transform-



Having seen the working of Fourier Transform, let us first see why wouldn't it serve the purpose in our case for not more than certain tasks.

Let us consider an example here-
Suppose we dial three DTMFs. When plotted, the audio signal would have three burst of sound each corresponding to the three DTMF that we just dialed.
Now when we move to frequency domain of the above signal, we will have 6 peaks- 2 corresponding to each dial. If we use FFT, it would simply give us all the frequencies occurring in the audio signal but still the order of dials will remain uncertain. Following are illustrations of the signal in time and frequency domain respectively.





Here, it is clear that time representation obfuscates frequency and frequency representation obfuscates time. For that very purpose- that is, to tell with definite certainty which DTMF is dialed in what order, we use STFT.
 This time varying spectral information can then be depicted using spectrogram. For the purpose of taking fourier transform as progressive time, we need to set a time window which can traverse accross the signal.

If window is long- frequency resolution in spectrogram is more. If window is short- frequency resolution in spectrogram is less.

2.5 Spectrogram Visualization

Spectrogram is a visual depiction denoting the frequen**cy** present in the audio track along with time. Taking the Fourier Transform over small time domains, we get the idea of STFT. The various Fourier Transform help us in getting the frequency domain as time progresses, frequencies being representatives of pitches, chroma can be obtained, these chroma are further used in construction if Spectrogram. The frequencies are represented on the Y axis, while the time on the X axis. The different colours and their instenisities depicts the contribution of frequencies at a particular time. An illustration has been used for better understanding.

3. Results and Observations

Similarity between two tracks has been obtained by two ways. On the basis of STFT, a visual representation has been introduced and on the basis of frequency matching, a numerical comparison has been obtained.

3.1 DFT of Signal

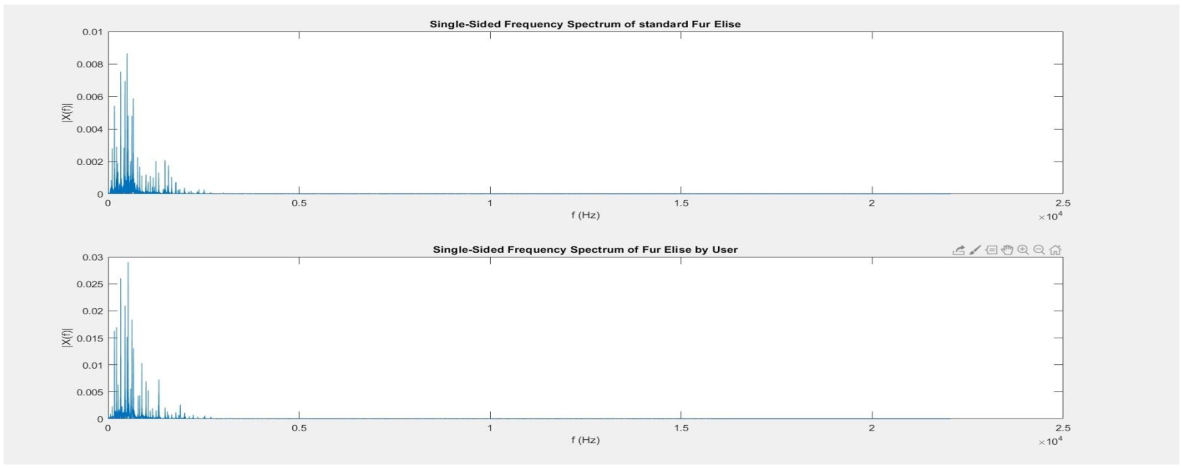
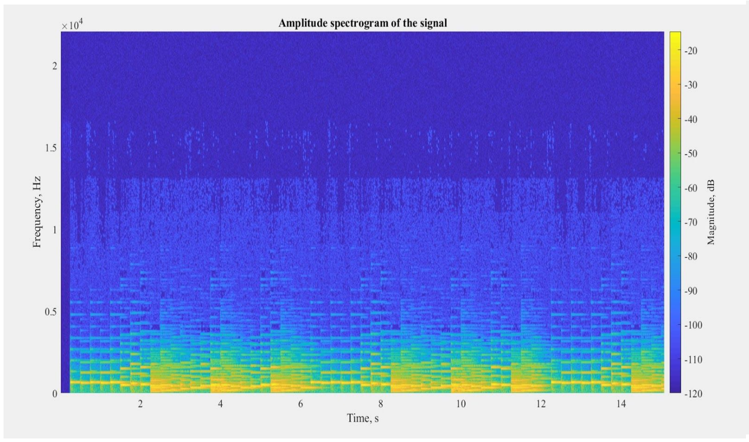


Figure x: DFT of Original Composition(above) ; DFT of Sample Signal(below)

3.2 Spectrogram Visualization



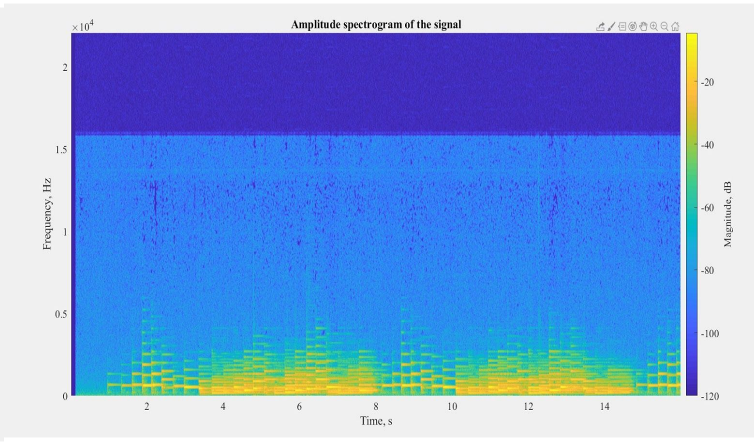


Fig 1: Spectrogram of Original Composition (Fur Elise)

Fig 2: Spectrogram of Sample (Fur Elise)

4. Conclusions and Limitations

4.1 Key Findings:

Similarity of signals was found in two respects - *first*, a more numerical approach which gave the similarity with respect to the frequencies being played by the user as compared to the original composition (utilizing the DFT of the signals) and *second*, a visual comparison of spectrograms of the two signals leading to a time delay understanding of the signals.

4.2 Shortcomings:

In Fast Fourier Transform(Cooley-Tukey), if samples N is not a multiple of 2, excess samples have to be added, these samples have amplitude zero. This results in large increase in space complexity, which leads to wastage of memory.

The correct range in which the data set is normalized and near by data are eliminated will vary in different signal and sensitivity you want from the algorithm.

4.3 Future Scope:

The parameters employed in normalizing the DFTs of input audio signals are twofold – *first*, the number of frequencies with considerable contribution to be taken into account and *second*, the sensitivity factor for normalizing the values i.e. the least gap for which the values should be considered different. These two factors (can be found for the maximum efficiency) using Linear Regression in future.
Also, a database could be maintained of the great compositions of all time to make it more convenient for the user to select a piece of music and compare his attempt against it.
Further, after a combined systematic analysis of the spectrogram and frequency domain representation of the signals we could venture into the genre-recognition area of audio processing.

5. References:

[1] https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm. Accessed: Nov 18, 2019.

[2] Lenssen, N. and Needell, D. "An Introduction to Fourier Analysis with Applications to Music," Journal of Humanistic Mathematics,
Volume 4 Issue 1 (January 2014), pages 72-91. DOI: 10.5642/jhummath.201401.05 . Available at: http://scholarship.claremont.edu/
jhm/vol4/iss1/5

[3] Chao Yang. “Week 7 19 4 Cooley Tukey and the FFT Algorithm”. (Sep 9, 2014). Accessed: Nov 16, 2019. Available: https://www.youtube.com/watch?v=WsJlJexWKPw

[4] LeisOS. “What is a Fast Fourier Transform(FFT)? The Cooley-Tukey Algorithm”. (Nov 27, 2017). Accessed: Nov 22, 2019. Available: https://www.youtube.com/watch?v=XtypWS8HZco.

[5] https://en.wikipedia.org/wiki/Short-time_Fourier_transform. Accessed: Nov 20, 2019.