Music Classification with Signal Processing Techniques

Christian Johnson
Electrical Engineering Department
United States Coast Guard Academy
New London, Connecticut 06320
Email: Christian.S.Johnson@uscga.edu

Daniel Nusraty
Electrical Engineering Department
United States Coast Guard Academy
New London, Connecticut 06320
Email: Daniel.Y.Nusraty@uscga.edu

Joshua West
Electrical Engineering Department
United States Coast Guard Academy
New London, Connecticut 06320
Email: Joshua.C.West@uscga.edu

***Abstract*—This paper explores the application of digital signal processing techniques in classifying and comparing music files. A method for quantifying similarity between two audio signals is developed utilizing the Fourier Transform to analyze frequency composition. This method is applied in order to generate music recommendations based on a single song. The study investigates the effectiveness of this approach through experimental procedures involving a diverse collection of music files. Results demonstrate promising outcomes, with the produced playlists organized in logical orders based on the similarity of songs. The paper concludes by discussing the implications fo the research, and suggesting avenues for future exploration in personalized music recommendation systems.**

I. Introduction

With the advent of music subscription services, many people have become accustomed to automatically generated playlists, tailor-made to their own personal taste. These services provide such playlists at the push of a button, analyzing the user's listening history to continuously recommend similar songs. Providers such as Pandora, Spotify, and LastFM maintain massive databases containing contextual information on millions of songs in order to present recommendations that match each listener's unique taste. For those who prefer to maintain their own local music library however, the options for tailored music recommendations are dramatically reduced. This paper seeks to explore the application of digital signal processing techniques in order to implement similar music recommendation functionality that will work with local music files.

II. Theory

A. Background

Music analysis is based on a commonly used function known as the Fourier Transform. A Fourier Transform is used to transform time-domain audio information into a frequency-domain representation. Sound in the time domain is represented as a waveform, where amplitude is a function of time. This amplitude represents vibrations in the air, which the human ear interprets as sound. The frequency domain, however, represents sound as magnitude based on frequency, which is more relevant than time information when analyzing music. For a computer to process this frequency signal, the amount of information present in the signal must be reduced through a process called *sampling*. Sampling replaces the continuous signal with a discrete representation; an array of values at (typically) evenly spaced indeces of time. Stated here without derivation, the Discrete Fourier Transform, or DFT, is most commonly used to transform analog signals into digital frequency representations. See [1] for more detailed information.

$$S(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi}{N}kn} \tag{1}$$

$$S : \mathbb{R}^N \to \mathbb{C}^N \tag{2}$$

Equation (2) shows the DFT as a functional map from real to complex values. Each point within the DFT output represents a point in an N-dimensional complex vector space, where the magnitude and phase of that point represent the content of time-domain signal at frequency $\frac{2k\pi}{N}$. These values depend on a concept known as *sampling frequency*, which refers to the space between each sample of a signal.

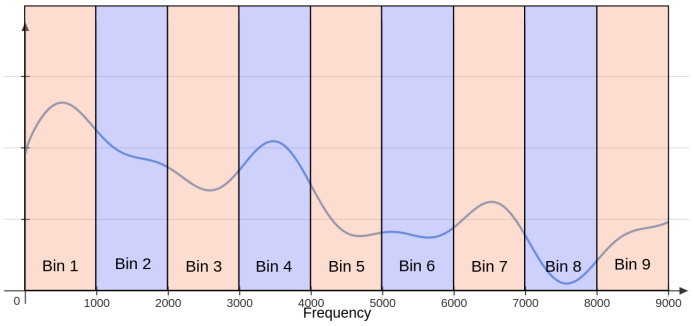

Fig. 1. Example Frequency Distribution Between Bins

Each 'bin' of $S[k]$ represents a collection of frequencies, $\frac{-F_s}{2N} + \frac{kF_s}{N} < f < \frac{F_s}{2N} + \frac{kF_s}{N}$. This relationship is shown in Fig.(1), which illustrates that $S[1]$ does not directly correspond to a specific frequency value; in order to find the DFT output for, e.g. 20Hz, one must determine which bin this frequency will fall into.

1) The FFT: In its original form, the DFT is computationally expensive and complex, comprised of $\mathcal{O}(N^2)$ operations.

An algorithm known as the Fast Fourier Transform (FFT) dramatically simplifies this complexity, reducing the expense to $\mathcal{O}(N*log(N))$ operations. It does this by exploiting symmetry within the DFT. Equation (1), the basic representation for the DFT, is periodic about N, i.e. $X_{k+lN} = X_k$ for any integer l. The FFT uses this relationship to break the DFT into even and odd components.

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x(n) e^{-j2\pi kn/N} \\ &= \sum_{m=0}^{N/2-1} x(2m) * e^{-j2\pi km/(N/2)} \\ &+ \sum_{m=0}^{N/2-1} x(2m+1) * e^{-j2\pi km/(N/2)} \end{aligned} \tag{3}$$

This particular implementation is known as the Cooley-Tukey FFT, and is one of the most widely used algorithms in the world.

B. Analyzing Music

In order to analyze and classify music, it is important to understand how it is structured. Fundamentally, music is composed of pressure waves vibrating at different frequencies and amplitudes. In western music, the most common method for organizing these frequencies uses what are known as scales, which group frequencies and assign them alphanumeric designations based on 'relative tone'. Relative tone refers to how the human ear interprets a note. For example, the notes A1, A2, and A3 are denoted as 55, 110, and 220 Hz respectively. These notes all 'sound' like the same note, in different octaves, and scales are groupings of these repeated notes. More information on this topic is presented in [2]. Fig.(2) shows 2 octaves of the C Major scale. Examining the

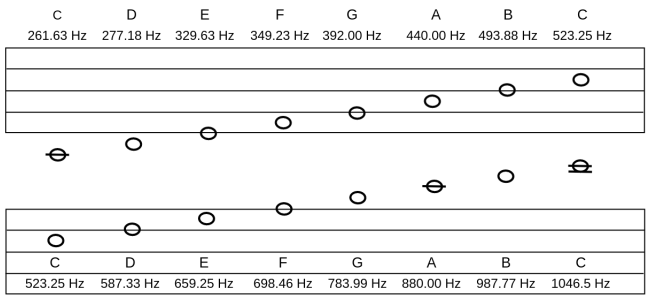

Fig. 2. Concert B Flat scale, C4 to C6

frequencies of this scale, and their distribution, it is apparent that the notes are not distributed evenly - music is instead arranged on a logarithmic scale. This conceptual understanding is what leads to a method for comparing different songs. Recognizing these scales as segments into which to divide the audible frequency spectrum, presents a possible metric through which to 'classify' or represent a song. The prevalence (or relative magnitude) of these different scales and the notes that make them up, represents information about a song's frequency makeup that can serve as a point of comparison. This assumption, that an audio signal's frequency makeup can be used to assess similarity to other signals, is the singular premise upon which this paper is based. Music recommendation is based upon the concept of 'similarity', and this paper asserts that frequency composition in terms of scale structure in music is an effective method with which to compare different songs. Based on this assertion, it becomes important to determine which frequencies, that is, which *scales* are most important when analyzing music. Table (I) shows 8 octaves of notes and their frequency values, and what becomes clear is that at higher frequencies, individual notes are spaced further apart than they are at lower frequencies. Based on this information, it is reasonable to assume that notes toward the lower end of the audible frequency spectrum will be more prevalent, and therefore more important to distinguish between for any system that seeks to characterize musical frequency. Therefore, defining frequency ranges that will characterize most musical pieces will feature much wider ranges at the high end of the spectrum, as individual notes are spaced further apart. Using frequency values that are similar to those shown in (I), a reasonable characterization range can be defined from 0-60Hz, 61-240Hz, 241-500Hz, 501-2000Hz, and 2001-8000Hz. There are few instruments that produce frequencies over 8000Hz, which makes this a reasonable cutoff in order to reduce the data that must be processed. These frequency ranges represent common groupings of frequencies that share musical and instrumental characteristics, which can be used to characterize a song.

III. Procedure

A. Processing a Single Audio File

Digital audio files are a collection of amplitude information arranged based on time. This information is sampled from the original continuous audio source, typically at a rate of 44,100 Hz (Meaning that there are 44,100 samples making up a single seconds worth of audio information). This format is incredibly useful for a computer which must play recorded audio, since the data will simply tell a computer the *amplitude* at which to vibrate a speaker diaphragm, and the time at which to do so. It is less applicable to examining and classifying the underlying composition of the audio however, and must be tranformed using the FFT to a more applicable format. Taking the FFT maps the sampled data from amplitude in terms of time to amplitude in terms of frequency, as discussed in section (II-A).

B. Interpreting Data

In order to use the frequency information generated by the FFT to compare two songs, it is important to generalize the information. Algorithm (1) demonstrates a method for transforming the FFT output data into a generalized representation. The algorithm describes the process of taking the FFT of an audio file and *segmenting* it. Segmenting the FFT data consists of seperating the data into frequency bands based on the values discussed in section (II-B). Within each band, the FFT bin

Note	Octave 0	Octave 1	Octave 2	Octave 3	Octave 4	Octave 5	Octave 6	Octave 7	Octave 8
C	16.35 Hz	32.70 Hz	65.41 Hz	130.81 Hz	261.63 Hz	523.25 Hz	1046.50 Hz	2093.00 Hz	4186.01 Hz
D	18.35 Hz	36.71 Hz	73.42 Hz	146.83 Hz	293.66 Hz	587.33 Hz	1174.66 Hz	2349.32 Hz	4698.63 Hz
E	20.60 Hz	41.20 Hz	82.41 Hz	164.81 Hz	329.63 Hz	659.25 Hz	1318.51 Hz	2637.02 Hz	5274.04 Hz
F	21.83 Hz	43.65 Hz	87.31 Hz	174.61 Hz	349.23 Hz	698.46 Hz	1396.91 Hz	2793.83 Hz	5587.65 Hz
G	24.50 Hz	49.00 Hz	98.00 Hz	196.00 Hz	392.00 Hz	783.99 Hz	1567.98 Hz	3135.96 Hz	6271.93 Hz
A	27.50 Hz	55.00 Hz	110.00 Hz	220.00 Hz	440.00 Hz	880.00 Hz	1760.00 Hz	3520.00 Hz	7040.00 Hz
B	30.87 Hz	61.74 Hz	123.47 Hz	246.94 Hz	493.88 Hz	987.77 Hz	1975.53 Hz	3951.07 Hz	7902.13 Hz

TABLE I
FREQUENCIES OF MUSICAL NOTES

Algorithm 1 Characterizing a Music File

function SEGMENT(File) ▷ *File should be an .MP3 file*
  $x \leftarrow$ .MP3 file audio data
  $X \leftarrow FFT(x)$
  $a =$ [Array of Frequency Breakpoints]
  $A =$ Index Breakpoints from a ▷ *Uses (4)*
  $B \leftarrow$ FFT Data segmented based on A
  for Segment in B **do**
    max←FFT bin with the highest magnitude
  list←all max values ▷ *A list of complex numbers*
  return list

(or index) with the highest magnitude will represent the most prominent frequency in that range. It is important to recognize that the FFT is normalized, which means that the indeces will not correspond to individual frequencies, as shown in Fig. (1). When segmenting the FFT, the frequency breakpoints must be converted to an index value using (4), where N represents the number of samples in an FFT signal, f representes the frequency value, and r represents sampling rate. This formula uses these values to determine the bin into which a specific frequency will fall.

$$\text{Bin Index} = \frac{f * N}{r} \tag{4}$$

This segmented FFT groups together several series of frequencies which make up similar ranges. Finding the bin with the largest magnitude within each segment produces an array of complex numbers, which presents a way to identify which frequencies are most prominent within a given audio file. Using the frequency breakpoints as defined in section (II-B) will produce a 5-element array, where each element is a complex number representing the most prominent frequency in a segment. The 5-element array stands as a characteristic representation of a single song, which can be used as a point of comparison. In order to determine the similarity between two songs, one must use these characteristic arrays and determine how similar those are. The Euclidean Distance Formula, (5), provides a simple method for determining the physical distance between the endpoints of two vectors.

$$d(\hat{x_1}, \hat{x_2}) = \sqrt{(x_1 - x_2)\overline{(x_1 - x_2)}} \tag{5}$$

Thus, the similarity between two songs can be approximated from each song's characteristic representation.

C. Storing Comparison Data

Section (III-B) shows how a song can be characterized through its frequency data and compared to other songs. This comparison takes the form of a distance metric, representing the proximity of the primary component frequencies for each song. The larger this distance metric, the less similar the two songs are. In order to construct a system which can handle more than two songs, it is necessary to construct a method for storing and accessing data. Algorithm (2) demonstrates

Algorithm 2 Saving Song Characterizations

function CHARACTERIZE(Directory) ▷ *Directory should contain .MP3 files*
  if D does not exist **then**
    Initialize empty dictionary D
  for File in Directory **do**
    C=SEGMENT(File) ▷ *C is the Characteristic Array*
    if File not in D **then**
      Append C to D, with File as key
      COMPARE(File)
  return D

the process for characterizing a directory with multiple .MP3 files and saving their characteristic arrays into a dictionary structure provides fast access and lookup time. Algorithm (3) demonstrates the process for comparing 2 songs together and saving their comparison data. The comparison data produced by this function is a list of smaller 3-element lists, where the 3 elements are 'Song 1', 'Song 2', and the 'Distance' between them. Using these two functions and the data structures that they create, a program creates a playlist from a filename taken as an argument. This program searches the list C produced by Algorithm (3), searching for any element that contains the input file, then creating a list sorted by distance. Removing all except the top 20 songs produces a 20 song playlist, composed of the songs that are most similar to the input.

IV. RESULTS

Because of the qualitative nature of music, it is challenging to evaluate the system's actual performance. In order to examine how well the system performs, it is necessary to track the results produced using a relatively diverse collection of music files. Using a dataset of around 21 songs, comprised of several songs each from about 7 artists, the system was consistently able to recommend logical sets of songs. One example output for this system is shown in Table (II), producing songs that

Algorithm 3 Saving Song Comparisons

Require: .MP3 file that is present in D
  function COMPARE(File)
    $X_1 \leftarrow$ Characteristic Array of File
    if C does not exist **then**
      Initialize empty array C
    for OtherFile in D **do** ▷ *OtherFile is every file other than the one passed to the function*
      $X_2 \leftarrow$ Characteristic Array of OtherFile
      D$\leftarrow$ distance between X_1 and X_2 ▷ *Uses* (5)
      Initialize empty list L
      Append to L: File, OtherFile, D
      if list containing File and OtherFile not in C **then**
        Append L to C
    return C

are similar to 'Bridge Over Troubled Water', by Simon and Garfunkel.

TABLE II
COMPARISON OUTPUT - BRIDGE OVER TROUBLED WATER

Song	Distance
MrsRobinson.mp3	17833.007342
ElCondorPasa.mp3	18187.548751
AHazyShadeOfWinter.mp3	30014.763253
StickSeason.mp3	33366.085974
TheViewBetweenVillages.mp3	39553.522808
TheBoxer.mp3	42437.789784
HeadingSouth.mp3	42465.180760
HomewardBound.mp3	45135.042716
MrForgettable.mp3	47201.975596
Someday.mp3	47231.605537
⋮	⋮
NeverGonnaBeAlone.mp3	176914.812287
IfTodayWasYourLastDay.mp3	178549.798450
AmericanIdiot.mp3	188508.497539
MeinHerzBrennt.mp3	218351.289405
BreakingTheHabit.mp3	285998.454949

This table displays song name and the 'Distance' metric, representing the similarity of the two songs. The top 10 songs are those that the program determined most similar to the input song, which is a good assessment. Those displayed at the top of the table are the same genre as the input, and many are written by the same artist. Conversely, the last rows in the table show the songs that the system found to be least similar, featuring songs with a dramatically different genre. The program consistantly produces similar results, generating playlists that are as similar to the initial song as the limited dataset allows. Performing the same operation on 'Breaking the Habit' by popular rock artist Linkin Park produces the output shown in Table (III). Songs by other rock artists appear prevalent at the top of this table, while less similar songs appear toward the bottom.

On the whole, the system produces remarkably reliable data. The top 10 songs in each playlist are a good match

TABLE III
COMPARISON OUTPUT: BREAKING THE HABIT

Song	Distance
LosingMyLife.mp3	167792.443135
AwakeMySoul.mp3	177931.103088
IWillNotBow.mp3	195567.314209
WatchTheWorldBurn.mp3	219692.835879
IWillWait.mp3	219933.052561
It'sNotMyTime.mp3	224138.133323
21stCenturyBreakdown.mp3	229613.249743
Numb.mp3	234717.128844
IntoTheNothing.mp3	240318.799362
ScarboroughFair.mp3	240868.302800
⋮	⋮
AnthemOfTheAngels.mp3	300251.679696
Burn.mp3	302620.855625
DuHast.mp3	309728.869723
AmericanIdiot.mp3	316130.617928
NeverGonnaBeAlone.mp3	331907.954410

for the input song, and represent a close match to the genre, tone, and musical quality. These topics are not discussed in depth in this paper, but more information can be found in [2], [3], or [4]. One interesting anomaly the appeared regularly across a range of trials was the 'median' data. There were a consistant subset of songs that appeared toward the middle of the table, regardless of the input song. This may simply be a factor of the dataset, and a potential lack of size or diversity. Further research opportunities could explore this phenomena, or explore other features that could be used to classify a song, such as tempo or frequency distribution, see [5] for more information.

V. CONCLUSION

This paper has explored the application of digital signal processing techniques to the classification and comparison of audio files, and the efficacy of these techniques for generating song recommendations. Analyzing the frequency content of audio signals and comparing them based on distance metrics proved to be an effective method for organizing songs, and demonstrates the application of frequency analysis for song recommendation. Experimental results show promising outcomes, with songs sorted in a seemingly logical order based on musical attributes, genre, and general composition. While the qualitative nature of music analysis presents challenges in evaluating system performance, this approach offers a promising foundation for further research and development in personalized music recommendation systems. Future work could involve refining the algorithm to provide results that better differentiate between extremely similar songs, improving the speed of the system, and exploring additional features in order to enhance the accuracy and relevance of music recommendations.

VI. Bibliography

References

[1] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 4th ed. Upper Saddle River, NJ: Pearson/Prentice Hall, 2007, literaturverz. S. 1053 - 1066.

[2] M. Gotham, K. Gullings, C. Hamm, B. Hughes, B. Jarvis, M. Lavengood, and J. Peterson, *OPEN MUSIC THEORY*, Jul. 2021. [Online]. Available: https://viva.pressbooks.pub/openmusictheory/

[3] W. Hongdan, S. SalmiJamali, C. Zhengping, S. Qiaojuan, and R. Le, "An intelligent music genre analysis using feature extraction and classification using deep learning techniques," *Computers and Electrical Engineering*, vol. 100, p. 107978, May 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622002506

[4] D. Silver, M. Lee, and C. C. Childress, "Genre Complexes in Popular Music," *PLOS ONE*, vol. 11, no. 5, p. e0155471, May 2016. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0155471

[5] Y. Meng, "Music Genre Classification: A Comparative Analysis of CNN and XGBoost Approaches with Mel-frequency cepstral coefficients and Mel Spectrograms."