CHROMAZAM: SONG IDENTIFICATION THROUGH CHROMAGRAM

Steven Belitzky Christopher Palace Albert Peyton

sbelitz@u.rochester.edu cpalace@u.rochester.edu apeyton@u.rochester.edu

ABSTRACT

We designed an algorithm like that of the popular app “Shazam”; upon recording a 5 to 20 second clip from anywhere in a song in our database, our algorithm is able to accurately identify the song from our database. In contrast to the “Shazam” algorithm, which uses spectrogram analysis for song identification, our algorithm uses chromagram analysis to accurately identify songs. In theory, our method provides some advantages over the “Shazam” algorithm; such as being able to identify a song regardless of the key, and without the exact waveform being saved in our database. Upon completion of our program, we will test how accurate our algorithm can identify a song of variable input length and distortion to compare its correctness to the “Shazam” method.

INTRODUCTION

Chromagram

A chromagram is a condensed form of the spectral information of a given waveform. It converts the different frequency responses across all audible octave bands into their pitch classes: their note names given by western music theory over time. As seen in the following figure, the chromagram is obtained by taking the spectrogram of a signal, and multiplying it, using matrix multiplication, with a conversion matrix. A visualization of the process for calculating the chromagram, as well as a visualization of the conversion matrix itself can be seen in the right margin.

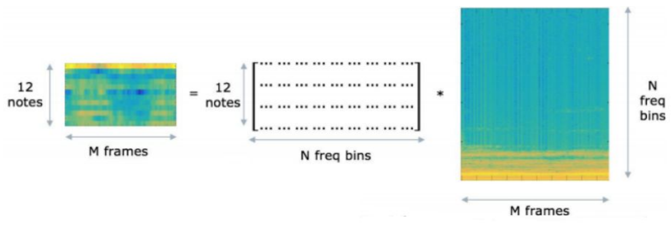

Figure 1. This outlines the chromagram conversion process. [1]

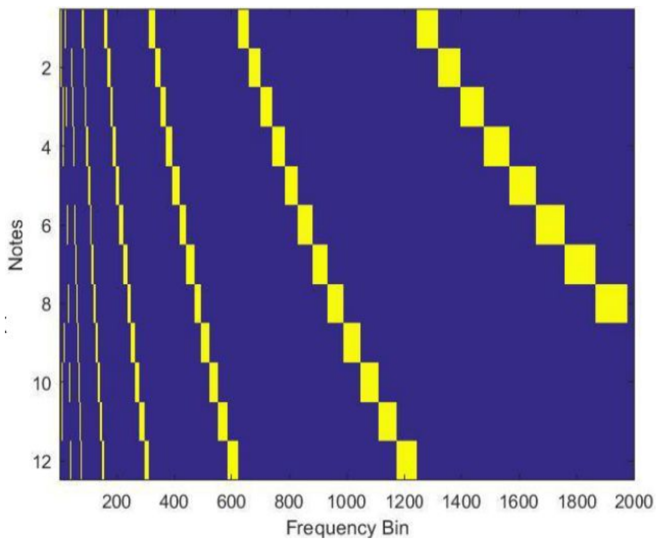

Figure 2. This is a visualization of the conversion matrix used to calculate the chromagram from the spectrogram.

Shazam

“Shazam Entertainment, Ltd. was started in 2000 with the idea of providing a service that could connect people to music by recognizing music in the environment by using their mobile phones to recognize the music directly. The algorithm had to

be able to recognize a short audio sample of music that had been broadcast, mixed with heavy ambient noise, subject to reverb and other processing, captured by a little cellphone microphone, subjected to voice codec compression, and network dropouts, all before arriving at our servers. The algorithm also had to perform the recognition quickly over a large database of music with nearly 2M tracks, and furthermore have a low number of false positives while having a high recognition rate." [2]
Our algorithm is similar to Shazam's as it identifies a song from a database given a short recording from any point in the song. The main difference is that our algorithm will attempt to accurately identify songs using a chromagram opposed to Shazam's method of using a spectrogram.

METHODS

The construction of our program consisted of two main phases: designing the algorithm and testing said algorithm. During the testing step we were able to tweak and adjust our algorithm to make it as accurate as possible at identifying songs.

Algorithm

The algorithm used for our project is broken up into two main steps: Preprocessing and Processing. The Preprocessing step analyzes each song or piece individually and converts it into a form where the processing step can compare songs to each other and correctly identify the input.

Preprocessing

The preprocessing step consists of multiple parts, the first of which is to check if the signal is stereo or mono. If the signal is mono the algorithm skips this part, else it converts the stereo signal to mono and moves on to the next part. Then we take the Short Time Fourier Transform of the signal. This step is done using the spectrogram function.

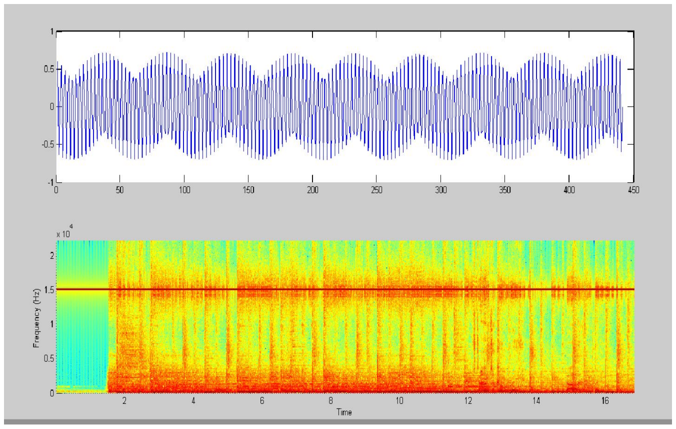

Figure 3. Above is a photo of a waveform and its frequency information o

Once the spectrogram is found for the signal, the appropriate chromagram conversion matrix can be formed and multiplied with the spectrogram. This produces the appropriate chromagram of the signal.
The next part of preprocessing is to take the most present note at each frame. This is done by creating a vector of the largest value at each frame and then comparing each value with each note to determine which note is most present.
After a vector of each note is calculated, a vector of the difference or interval between each note and its successor is formed; this allows us to identify the song regardless of key. This vector initially has values between -11 and 11. In order to ensure that key or original version does not play a role in our algorithm, we add 12 to any values that are negative. The song is now represented by a vector of integers between 0 and 11, which can be compared to other pieces that have gone through the same preprocessing method.

Processing

After preprocessing is run on each song in our database and on the input recording, they can go through processing and be compared against each other in order to correctly identify the song.
The main concept behind the processing step of our

algorithm is the Euclidean Distance formula, it is outlined in the image below:

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}.$$

Figure 4. This is the Euclidean Distance formula

Our algorithm takes the vector that represents first song in the database and lines the beginning of it up with the vector that represents the recorded sample. The euclidean distance is calculated then the vector that represents the recorded piece is moved one frame over in the vector that represents the song within the database. This continues until the end of the vector that represents the recorded sample is lined up with the end of the vector that represents the song in database. Our algorithm saves the smallest euclidean distance as a way to represent how close this song resembles the song in the database.
This process repeats for each song and a single euclidean distance value is saved to represent each song in our database. The song with the smallest euclidean distance value from the previous step is identified as the song in the recorded sample. Our program's accuracy is discussed in the section below.

Tests and Results

We initially tested the functionality of our Chromazam algorithm by running it with songs that are typical of today's popular music. These songs consisted of multiple instruments playing simultaneously, with multiple melodies, rhythms, counter-melodies and harmonies all occurring at the same time. We first tested our algorithm with 5 songs of this type. Upon this initial test our algorithm yielded 0% rate of accuracy revealing that our algorithm is ineffective with songs of this type. We believe this to be because these 5 popular lead guitar, piano melody or really any melodic part songs contained frequency information that spanned the entire spectrum and ultimately increased the the intensity of chromagram as a whole, making it impossible to calculate the most present note at each point.
Upon seeing these results, we compiled a database of 10 simpler, more or less monophonic songs such as a solo violin, tuba, trombone, vocal, and piano pieces. Some of these pieces still had some form of accompaniment, however, all 10 of the pieces consisted of a single melody line that stood out significantly from any other part. While testing with these 10 pieces, our algorithm was actually quite effective. We tested these by varying the length of the recorded signal (5 second, 10 second, 15 second, and 20 second long inputted signals) and by adding varying amounts of white gaussian noise to the inputted signal (using input signals with a SNR ranging from -15 to 30).
As you can see by the results below, with simple monophonic songs, our algorithm works more effectively with longer input signals. Furthermore, our algorithm is effective when the SNR is high but ineffective when the SNR is low.

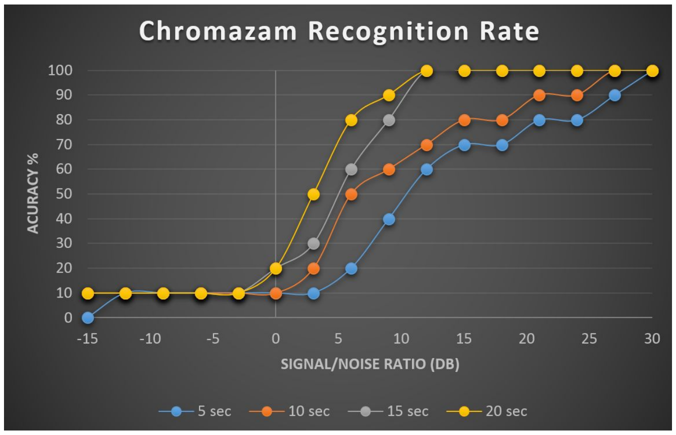

Figure 5. This is our results with varying input length and signal-to-noise ratio.

Below is a similar test run on the Shazam algorithm. As you can see, the Shazam algorithm works better than ours when noise is added but they are comparable when using input recordings with high signal-to-noise ratios.

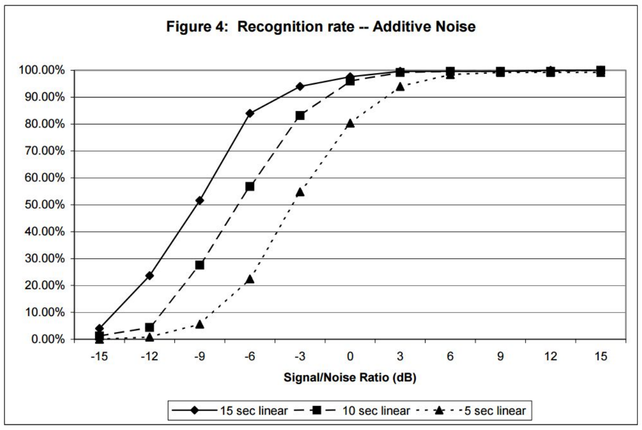

Figure 6. Shazams results conducting a similar test to that which we did

.

CONCLUSION

Our current algorithm is not comparable to Shazam's because it does not have the ability to accurately identify most polyphonic songs. When multiple melodic or harmonic lines are present in the inputted waveform, our program is overwhelmed by the amount of activity in the spectrum and is unable to distinguish the most prevalent pitch classes. That being said, we were successful at creating an algorithm that can successfully and efficiently identify monophonic songs from a database using a chromagram. We were also successful at identifying a song regardless of its key, something the Shazam app does not allow. In conclusion, we were quite overzealous in our proposal and were under the impression that we would be able to accomplish a lot more than we did, but in most ways our project was a success and we have still have the ability to develop and further improve our algorithm.

Future Work

For our future work with this project we plan to implement a source separation algorithm which will be able to extract different parts from a piece to be analyzed. These parts could be anything; the vocals, This will solve the problem of extraneous information overwhelming the processing. When this problem is solved and our algorithm will be able to account for polyphonic pieces, we will be able to increase the number of songs in our database and construct hash tables to store data and run our program as efficiently and effectively as possible. We could also implement a noise removal algorithm into our code so it will be able to accurately identify songs at lower signal-to-noise ratios.
Another idea that recently occurred to us was to only analyze the low frequencies of a track (20-250Hz) that way we cut out a lot of higher frequency harmonics that may have made our program less efficient.

References

[1] Y. Zhang, "Chromagram Representation for Musical Signals", University of Rochester, 2017.
[2] Shazam Entertainment, Ltd, "An Industrial-Strength Audio Search Algorithm", 2003.