# AUDify

Sanskar Mani(B18CSE048)
Siddhant Attri(B18CSE052)
Divya Singh(B18EE014)
Arushi Midha(BI8BB006)
Vedanti Alaspure(B18EE054)

**Abstract**

AUDify is a song recognition platform, which is vastly reliable, flexible and fast in its approach. It is built to filter out background noises from a piece of audio recorded using an inefficient microphone and subject to compression, distortion and ambient disturbing noises. The filtered audio which is relatively free from noise is compared to a plethora of soundtracks stored in a pre-existing database. This is done by juxtaposing fingerprints extracted from both the audio sources, that is, the unidentified recorded piece of audio and the songs stored in the database, against each other. TheIn our everyday life, we listen to a number of songs on a daily basis. In such a case, finding a song that you connect with, but not knowing where to look for its original soundtrack, is a common scenario. Intrigued by this scenario and taking inspiration from the extremely handy song recognition app 'Shazam', we decided to work on AUDify. AUDify aims at identifying a song by analyzing a short audio of it recorded by the user. The algorithm focuses on recognizing the song from its recorded audio segment, which would most likely be distorted due to background noise and compressed, owing to the fact that the recording would be done using a tiny in-built microphone on a mobile phone. Moreover, the algorithm targets to perform the recognition as fast as possible with a low number of false positives, that is, with the minimum scope of error in detection. To successfully enable song recognition, we utilized the following processes in the same order as mentioned- sampling, followed by quantization. Next, the Discrete Fourier Transform was done using a Fast Fourier Transform technique. Fingerprints, which are extracted hash tokens from the audio samples, were then matched with the fingerprints of the songs present in our database and then the result was computed.

To implement the concept of fingerprinting using Shazam algorithm, we referred to the paper, "An Industrial-Strength Audio Search Algorithm" by Avery Li-Chun Wang, Shazam Entertainment, Ltd.

obtained song matches are retrieved in the descending order of similarities found, that is, the song with the maximum number of matching fingerprints is displayed first, followed by the other matching songs in a decreasing order. Thus, the desired song can be found from the list of possible matches.

## 1. Introduction

to

, the signal is quantized.

### 2.2 Pulse-Code Modulation(PCM)

PCM is used to digitally represent a sampled analog signal. Conversion of a multiple channel stream (stereo) is done to a single-channel stream (mono) by averaging the values from the left and right streams.

## 2. Audio Fingerprinting

### 2.1 Sampling and Quantization

Analog signals are continuous signals which implies that even in an infinitesimally small time interval, it stores infinite information. For the audio to be analyzed and stored, it needs to be digitalized. This is achieved by sampling the signal, that is, converting it from analog to digital. According to the Nyquist-Shannon theorem, for a sampled signal to be reconstructed, the sampling frequency should be greater than or equal to two times the frequency of the sampled sig**From digital sound to frequenci**nal. So, the sampling frequency was chosen as 22100 kHz, with the assumption that most songs do not contain frequencies greater than 10500 kHz.

This is done using a Discrete Fourier Transform.

**2**To store the loudness which has an infinite variety of volume in a continuous signal to a discrete variation

### 2.3 es

The obtained discrete-time signal is converted into a frequency signal for the analysis of frequencies in the audio and hence, to achieve an accurate implementation of the algorithm. **.3.1 Discrete Fourier Transform**

$$X(n) = \sum_{k=0}^{N-1} x[k] e^{-j(2\pi kn/N)}$$

In this formula:

- N is the size of the window: the number of samples that composed the signal
- X(n) represents the nth bin of frequencies
- x(k) is a kth sample of the audio signal

A bin of frequencies is the smallest unit of frequency the DFT can compute. The size of the bin(frequency resolution) equals the sampling rate of the signal divided by the size of the window(N).

### 2.3.2 Windowing

Windowing is done to analyze the most necessary parts of the frequency signal obtained by DFT by bifurcating it into small segments. The signal audio(t) is multiplied by a window function(t). This Window function produces spectral leakage, that is, the introduction of

artificial frequencies inside the audio signal. For our audio signal, we used a hamming window function which was optimal to reduce spectral leakage and handled the noise.

#### 2.3.3 Fast Fourier Transform

DFT is too time-taking as it requires approximately 2*N^2 operations. Therefore, a faster approach called Fast Fourier Transform is done to reduce the time complexity from O(N^2) to O(N). FFT uses the Cooley- Tukey algorithm which involves division of the N-sample window into two parts and then computing the FFT recursively for each. In the end, the resultant FFT is computed from the previous two FFTs.

### 2.4 Spectrogram filtering

Spectrogram filtering is done to obtain the spectrogram of the audio signal in order to extract fingerprints from it. The spectrogram from FFT is filtered to extract only strong frequencies in the spectrum which denote the loud notes. For each FFT result, 512 bins are divided into 5 logarithmic bands. For each band, the strongest bin of frequencies are kept. Then, the average value of these 5 powerful bins is calculated. Only the bins that are above this average value remain in the filtered spectrogram.

### 2.5 Storing Fingerprints (and using the spectrogram to search for the song)

A database of filtered spectrograms for many songs is precomputed and stored. A short recording (minimum 2-3 seconds) of an unknown song is done and its filtered spectrogram is computed. This 'small' spectrogram is compared with the 'full' spectrogram of each song in the existing database by superposition. The filtered spectrogram provides a large number of frequency points. Hence, it would be too time-consuming to compare each point one by one. So, instead, we compare a set of points that are collectively termed as a target zone.

### 2.6 Search and scoring the fingerprints

The search is performed by generating an address structure during the fingerprinting of the recorded audio. In order to create the addresses, we need to create an anchor point per target zone. The address is of the format: ["frequency of the anchor";" frequency of the point";" delta time between the anchor and the point"]. Each address from the record is used to search in the fingerprint database for the associated couples: ["absolute time of the anchor in the song";" ID of the song"]. This search returns a large number of couples, let's say N.

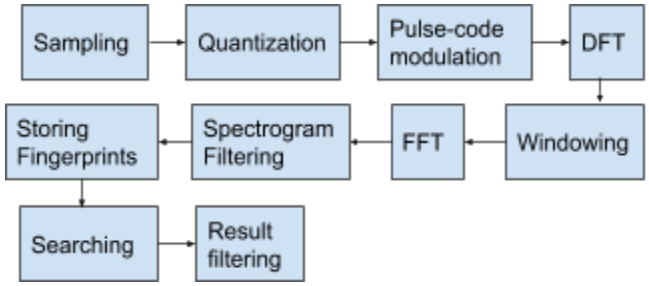
### 2.7 Result filtering and time coherency

The search is performed by generating an address structure during the fingerprinting of the recorded audio. In order to create the addresses, we need to create an anchor point per target zone. The address is of the format: ["frequency of the anchor";" frequency of the point";" delta time between the anchor and the point"]. Each address from the record is used to search in the fingerprint database for the associated couples: ["absolute time of the anchor in the song";" ID of the song"]. This search returns a large number of couples, let's say N.

The following process needs to be done for all the remaining songs:

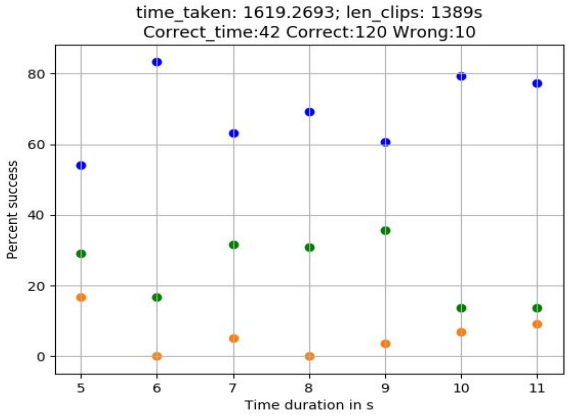
- For each address in the record, we get the associated value of the song and we compute delta = “absolute time of the anchor in the record” – “absolute time of the anchor in the song” and put the delta in a “list of delta”.
- It is possible that the address in the record is associated with multiple values in the song (i.e. multiple points in different target zones of the song), in this case, we compute the delta for each associated values and we put the deltas in the “list of delta”.
- For each different value of delta in the “list of delta” we count its number of occurrence (in other words, we count for each delta the number of notes that respect the rule “absolute time of the note in the song = absolute time of the note in record + delta”).
- We keep the greatest value (which gives us the maximum number of notes that are time coherent between the record and the song).

From all the songs, we keep the song with the maximum time coherent notes. If this coherency is above (the number of note in the record)*( coefficient) then this song is the right one.




## 3. Results and Observations

The result computed is as follows:

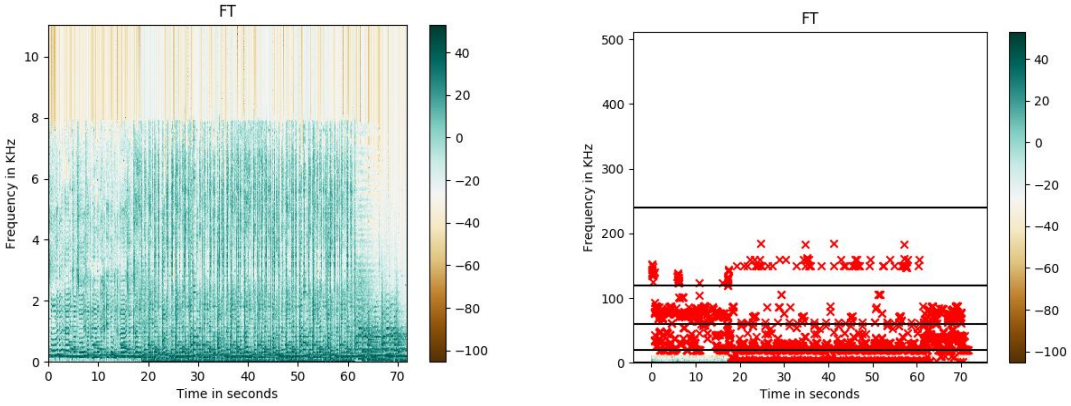



*Correct(blue) = Percentage of matches where the absolute difference in the time predicted by the algorithm and the time where the given song began in the actual song is not within the tolerance, but the fingerprint matches.*

*Correct_time(green) = Percentage of matches where the absolute difference in the time predicted by the algorithm and the time where the given song began in the actual song is within the tolerance, that is, 1 second, and also the fingerprint matches.*

*Wrong(orange)= Wrong matches*

Therefore, 162 correct matches were found for 172 samples of random clips extracted from the database. The spectrogram after DFT and the filtered spectrogram after spectrogram filtering for the song (Neovaii - I Remember) are:




## 4. Conclusions and Limitations

162 correct matches were found for 172 clips. Therefore, the accuracy of the algorithm is nearly 94.18 %. The algorithm has the following limitations:

- A clip of the original song should already be present in the database for identification. Songs cannot be identified by simply singing or humming them.
- Since we have assumed the maximum frequency to be encountered in a song sample to be nearly 11000 kHz for faster results; in the worst possible scenario, if a frequency higher than that is encountered in the audio sample, it could lead to aliasing.
- If we analyze parts of the song that have very weak frequencies( such as the end or beginning), we could end up with wrong values of strong frequencies( during spectrogram filtering), since the mean of these low frequencies would be very less.

AUDify could have further enhancements and can be used for the detection of the genre of a song using a filtered spectrogram. Moreover, this algorithm can be utilized to check for song plagiarism. Accuracy can be enhanced by using a better setup and sampling at 44100 Hz and time complexity can be reduced further by using downsampling. Since human ears follow a psychoacoustic model, low sounds in a 'raw' song are artificially increased to make them perceptible to the human ear as compared to the shriller sounds. Decreasing the frequencies of such low sounds would enable us to get the original 'raw' song and hence, enhance accuracy.

## 5. References

[1] Wang, Avery. (2003). An Industrial-Strength Audio Search Algorithm.
[2] Coding Geek website: http://coding-geek.com/how-shazam-works/