ISSN: 2320-5407 Int. J. Adv. Res. 6(12), 690-702

ISSN NO. 2320-5407

Journal Homepage: - www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI: 10.21474/IJAR01/8189

DOI URL: http://dx.doi.org/10.21474/IJAR01/8189

RESEARCH ARTICLE

STUDY ON FRAMEWORK OF AUDIO FINGERPRINTING AND SHAZAM'S WORKING ALGORITHM.

Mandla Vamshi Krishna[1] **and Dasika Moukthika**[2]**.**

1. Sreenidhi Institute of Science and Technology.
2. Maturi Venkata Subba Rao Engineering College, Department of Computer Science.

……………………………………………………………………………………………….....

Manuscript Info

…………………….

Manuscript History

Received: 10 October 2018

Final Accepted: 12 November 2018

Published: December 2018

Keywords:

audio,fingerprinting,sampling,frequency, combinatory hashes, analog & digital signal.

Abstract

………………………………………………………………

Inquisitive to know the songs and vanquishing the idea of remembering many songs lead to the innovation of this most anticipated music identification application called Shazam. Many music identification applications have been developed, but some of the most important features which are applied in Shazam makes it distinguishable from others. Shazam is an adaptable music quest engine and it identifies audio file based on the music played or recorded with the concept of fingerprinting. Fingerprint summarizes the whole audio file. Audio fingerprinting is used to match the song from database using limited number of bits. It's computational time is quick and easy with few faulty positives and excessive identification rate. By using fingerprinting mechanism, there is no requirement of any details of the song like singer , movie name to identify the song. Fingerprint is extracted from a framework after progressing through specific stages. Shazam uses fingerprints to identify the song which is obtained by analog to digital conversion, sampling, quantizing ,pulse coded modulation ,down sampling .Mainly the working algorithm of shazam uses combinatory hashed time and frequency pattern or topological scrutiny of the audio and finally querying for match of hashes in the database. Shazam ignores the noise added to the music while recording when you try to identify a song in a noisy environment. Shazam is significantly expandable. These concepts will be divulged in this paper and gives the readers a fundamental overview of Shazam's working algorithm.

……………………………………………………………………………………………….....

Introduction:-

As we all know, Sound is a vibration with different frequencies and it is decrypted by our ears and it is seen as a sinusoidal waveform by a computer. Things which make sound musical are loudness ,timbre and pitch(musical note) which means that music is an ordered sound. Human perception of loudness depends upon the frequency of the sound. We do not perceive exact same sound when it's been played by a singer, a piano or a guitar because each has its own timbre for a given pitch (musical note). The sound produced has lowest frequency as well as highest frequency (more overtones).A music can be seen with a spectrogram which is a 3 dimensional graph, on X-axis and Y-axis we have time and frequency respectively where high sounds are on very top and deep sounds are on very

Corresponding Author:-Mandla Vamshi Krishna.
Address:-Sreenidhi Institute of Science and Technology.

bottom and on the other axis we have amplitude of frequency which indicates how loud the frequency is and in general, A spectrogram is a voiceprint (in our context it is a representation of a music/audio file). The musical song is composed of several singers and instruments. To work fundamentally system needs to take this large amount of data and reduce it down to very small amount of data (reducing the information density) which is done by taking the fingerprint of the audio. In order to operate the data in systems, analog signal which is continuous both in amplitude and time is modified into discrete(electronic/digital) form which could be managed by electronic devices. For this continuous to digital conversion sampling is used. Shazam is a flexible audio search engine. It resembles Google which means when you give Shazam a small clip of audio then it gives you back the audio or probably the metadata related to that audio file. Shazam is noise/distortion tolerant. For example, if we type any misspelled words in Google it gives you the results correcting that based on previous results. Similarly, Shazam ignores the noise when you try identify a song in a noisy environment.

audio fingerprinting:-

An audio fingerprinting is an acoustic classification technique. The term Acoustic describes the properties of the sound waves. Audio fingerprinting is an appressed content-based mechanism that epitomizes an audio recording. It identifies the audio file independent of its format and links the unlabeled audio to its coterminous metadata (song title, genre, artist's name etc.). Using matching algorithms and fingerprints gnarled interpretation of recording can be identified as the same audio content and later stored in the database. Fingerprint is shorter numeric sequence procreated from an audio signal which is used to instantly locate or identify similar items in database. Whenever a new song is released its fingerprint is calculated and stored in the database. While trying to match the fingerprint if the song has not been broadcasted it should not be identified as a match.

Features of fingerprinting mechanism:

1. Efficiency plays a crucial role in calculating fingerprint of an audio and searching for the best match among huge collection of fingerprints. It should accurately identify an item delinquent of distortions, compression in a transmission channel.
2. It should be able to identify the song within few seconds.
3. Performance also depends on number of correct or wrong identifications.
4. The mechanism should be robust and should defend the vulnerable conditions like tampering.
5. It should be expandable enough to maintain huge databases of fingerprints.

Creating Database:

The acquisition of recordings which are to be recognized are passed to system and their fingerprints are calculated and are linked with their metadata and stored.

Identifying the unlabeled audio:

Later the fingerprint of unlabeled audio is extracted and is compared with the repository of fingerprints from the database. If match is found the respective tag or metadata of that particular audio is obtained from database.

Framework Of Audio Fingerprint Extraction

The framework as shown in fig 1 consists of several stages which are Preprocessing, Framing and overlapping, Transform, Feature Extraction and Post processing.

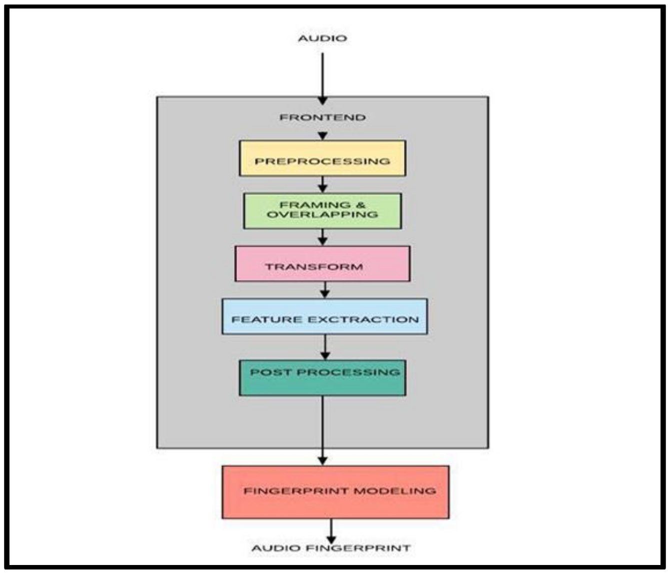


Fig 1:-Framework

Frontend

The frontend converts the audio signal into desired features which are useful in calculating fingerprints. Few factors are to be considered while designing front end. They are dimensionality reduction, robustness and so on. Dimensionality reduction is the process of converting a high dimension data set into a data set having lesser dimensions so the same information can be represented more efficiently in reduced space. This can be done by using linear transformation into components by separating noise and representing the signals which are important according to original audio.

Preprocessing

The stages in pre-processing are

1. Digitization- Sampling, Quantization
2. Conversion into mono PCM
3. Normalization

Digitization

It is the procedure of transforming an analog signal to digital. Since the song played in electronic devices like mobile, MP3 players etc. is in the form of the digital signal so, digitization needs to be done. It is performed in two stages Sampling and Quantization.

Sampling

In digital signal audio is in the form of bits whereas analog signals are continuous so it is not possible to store such huge information in bits. Therefore, we record the audio at distinct intervals of time (samples)without losing its quality and information. This is called Sampling. For example, let us take the analog signal as shown in fig1.1 lasting for 10 seconds. We need to perform sampling on this signal by recording the audio for every 1/10 th second. Then the continuous analog signal can be represented using a small sample. We need to consider the minimum possible division of the signal. 44,100 samples per second can be recorded generally. In order to transform the continuous values into discrete values there must be fixed aperture between the samples. The fixed aperture is Sampling period Ts. Reciprocal of Sampling period is Sampling rate. An analog signal (sine wave) with frequency f is a sine wave which requires minimum of 2 samples per cycles is needed in order to identify the wave form. This can be obtained when the frequency of the sampled signal as shown in fig 1.2 is double the original signal's frequency otherwise leads to under-sampling. According to Shannon and Nyquist 40,000 units(samples) is minimum needed in order to digitize 0hz-20khz (human audible range) signal.

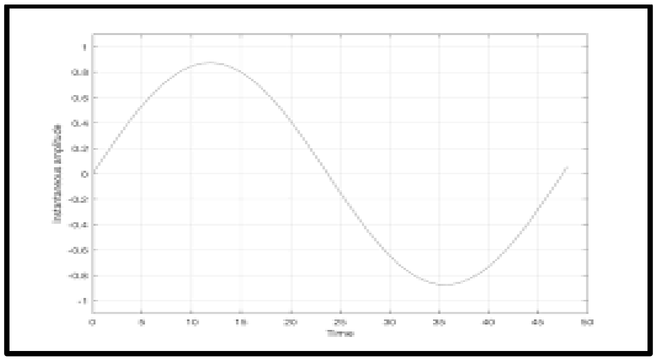


Fig 1.1:-Analog Signal

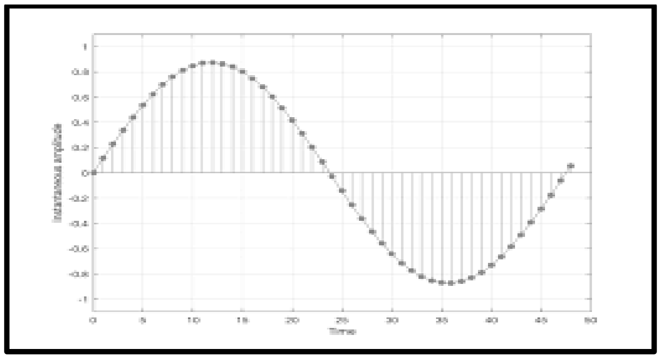


Fig 1.2:-Sampled signal

Quantization
Quantization is a technique of converting continuous values of the signal into set of discrete values. The level of quantization is represented using bits like 8-bit (256 levels) ,16 -bit (65,536 levels) quantization (standard quantization) etc. The quality of the signal is preserved based on the number of quantization levels. The distance between any two adjacent levels of quantization is called a quantum. Quantization is classified into two types- uniform quantization in which the levels of quantization are regularly distributed over the strength of the signal. If the quantization is done based on non-linear function it indicates non-uniform quantization. Based on the probability distribution of the audio signal the choice of a specific method of quantization is made. For the music, songs etc. we generally use uniform quantization because these signals have uniform probability density. In a signal, noise also exists. SNR (measured in Decibels) is the ratio of signal and noise which is used to measure the quality of signal. With the increase in SNR the noise decreases.
SNR=20log (Vs/Vn) where Vs=signal voltage
Vn=noise voltage

The difference between original value of sampled signal and its nearest quantized interval values is called Quantization error shown in fig 1.3. The worst error occurs at half interval.

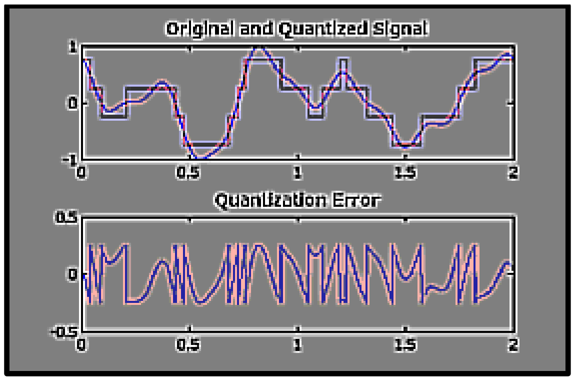


Fig 1.3:-Quantized signal and quantization error

rms power of signal Vs=2^N Δ/2sqrt2 where N=no. of bits per sample

Quantized Noise variance Vn=(Δ)^2/12
SQNR=Vs/Vn=sqrt (1.5) *2^N in decibels

Vs is divided among 2^N steps each step with a quantum Q
SNR=20 log(Vs/Vn)
=20 log((2^NQ/2√2)/Q/√12)
=20 log(2^N√12/2√2)
=20 log(2N) +20log (√6/2)
=6.02N+1.76(dB).

Thereby each bit gives 6dB resolution. Hence 16-bit quantization shown in fig 1.4 results in maximum SQNR of 96dB.

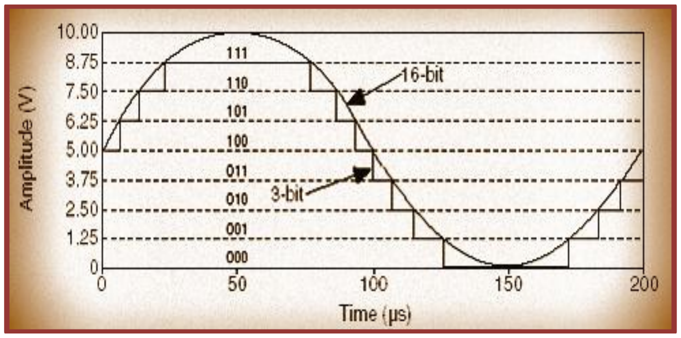


Fig 1.4:-16-bit quantization

Conversion Into Mono Pcm
Pulse Coded Modulation shown in fig 1.5 is a technique used to convert the analog signals into sequence of bits 0's and 1's. Pulse Coded Modulation is done in order to transmit analog signal over high speed digital channel. While performing sampling we may encounter with redundancy. In order to reduce the redundant information and have an efficient output we use Differential Pulse Coded Modulation (DPCM). DPCM is based on prediction where current sample is predicted from its past sample which is not accurate enough. During DPCM as the redundancy decreases bits needed per sample also decreases. Differential PCM has a transmitter (sampled input, Quantizer, prediction filter, encoder) and a receiver (Repeater, reconstruction filter, Decoder). First, the analog signal is passed through low pass filter prior to sampling, the high frequencies in the audio signal are eliminated to evade aliasing. In the fig 1.5 comparator is represented by circle which is used to find the signal error(e(nTs)) which is the deviation between the predicted signal(X^nTs) and sampled signal(X(nTs)). Prediction filter produces the predicted value(X^nTs) which refines the signal from undesirable segments. The previous predicted value and the output(eq(nTs)) from the quantizer is combined and given to prediction filter (Xq(nTs)). Since the quantizer error(eq(nTs)) is negligible and is concealed with less number of bits are used to represent it. As a result, bitrate

(number of bits per sample) decreases in DPCM.
signal error e(nTs)=(nTs)- X^n(Ts)
output of quantizer
eq(nTs)=q(nTs)+e(nTs).................eq1
where e(nTs)=signal error, q(nTs)=quantization error
input of prediction filter
Xq(nTs)=eq(nTs)+X^n(Ts)..............eq2
substitute value of eq(nTs) in eq2
Xq(nTs)=q(nTs)+e(nTs)+X^n(Ts)
since e(nTs)+X^n(Ts)=X(nTs)
therefore Xq(nTs)=q (nTs)+X(nTs)

1st bit (MSB) represents polarity which is either negative (0) or positive(1). 2nd,3rd,4th bits indicate sampled value location.5th,6th,7th,8th (LSB) bits indicate one among the obtained segments of sampled values where each segment is partitioned into 16 levels. This encoding process depreciates the bandwidth. Then the DPCM signal is passed through the repeater which boosts the signal's strength, reimburses the information lost and recoup the signal. At the Decoder in the receiver's end the modulated output signal is passed as the input. Reconstruction filter regenerates the quantized form of the actual signal differing by quantization error(q(nTs)).

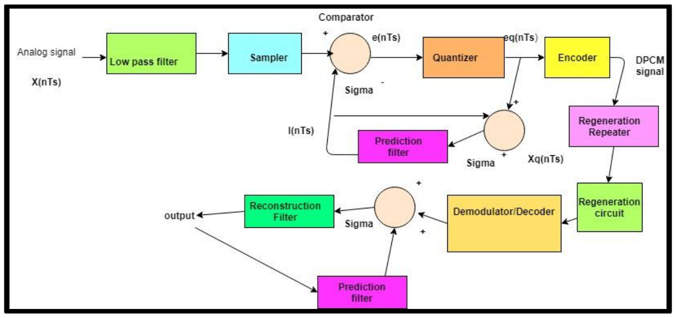


Fig 1.5:-Pulse Coded Modulation

Normalization

If the audio of the song played is very low then without affecting the signal we can increase its loudness. This process is called normalization shown in fig 1.6. This can be done in two methods: detection of peak values and RMS values. In the first method stated based on the height of the peaks they increase the loudness. It cannot be increased beyond 0dBfs.Peak volume detection is not suitable for multi-track recording. When we add more clips, it results in overhead.to the RMS based detection considers both the subtle sounds and larger peaks and increases the loudness by taking the average between two. This method is close to the working of human ear. For human beings the frequencies between 1k hz - 6k hz is anticipated as loud sounds. But this method will not consider it. There is another advanced method called EBU R-128 volume detection in which it cleverly listens to the sound and interprets the way the human does and takes into consideration of the sounds between 1000hz and 6000hz.

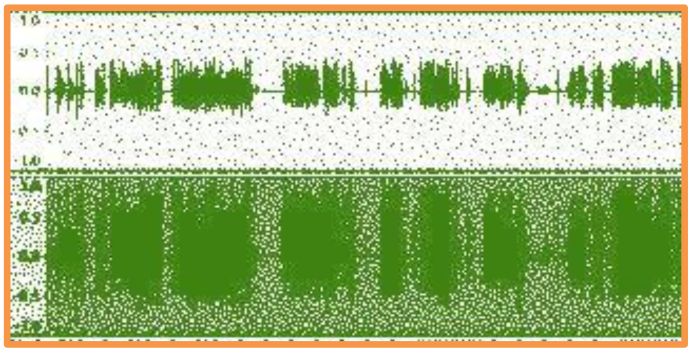


Fig 1.6 :-Normalization

Framing And Overlapping
The signal is partitioned into frames whose stature is proportionate to pace of cardinal acoustic predicaments. Because the signal is considered as static over subordinate time intervals. In order to impose a restriction on the disruption at both ends we need to apply window function. Before applying window function we need to covert the time domain signal into frequency domain signal because Shazam operates with frequencies. To perform that conversion, we have a particular transformation method known as Fourier Transform.

Transform
To get the better results we need to apply Fourier Transform on smaller components of the whole signal then we can obtain the frequencies for each 100-millisecond part of signal. Since we need to perform transformation on digital signal which is finite and has discrete values we apply Discrete Fourier Transform (DFT). While performing transformation we operate on a single channel.

There is mathematical formula for Discrete Fourier Transform

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

where X(n) represents nth frequency bin
x[k] represents kth audio signal sample
N is the window frame size

DFT calculates discontinuous spectrum. Bin size is calculated as spectral/frequency resolution that is same as sampling rate/window size. When 44.1khz is the sampling rate maximum frequency resolution is 10.77hz.DFT cannot separate any two frequencies which are adjacent to 10.77hz. For an audio signal only, half of the bins calculated by Discrete Fourier Transform is needed. Since 10.77hz is the 2047th bit. (Y+2048th) bin is equal to the xth bin. Fast Fourier Transform is the optimized method for Discrete Fourier Transform.
Deconstruction of audio signal with FFT:

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

Using FFT time complexity decreases. When the signal is represented using amplitude-frequency graph as shown in fig 1.7 with amplitude of the original song on Y-axis and frequency on X-axis the highest spike indicates the frequency of the sinusoidal signal with highest voltage. When we look at the signal in frequency domain we get only single spike because we are outputting the signal to only single frequency.

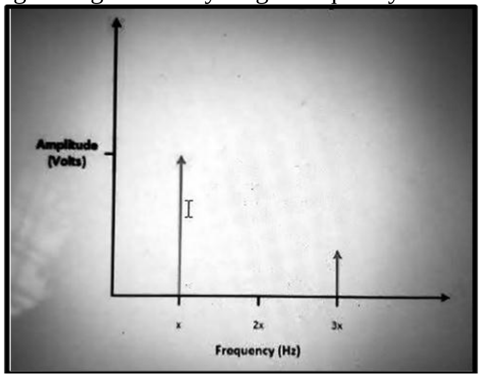


fig 1.7:-amplitude-frequency graph

Window Function
When the ordinal of periods of the sampled signal is not an integer, acute transitions and discontinuity at endpoints is observed. This results in high frequency units (greater than Nyquist frequency) which are not present in the actual

signal and gives us the smeared version of the spectrum which leads to Spectral Leakage as shown in fig1 in which frequency energies are leaked to one other.

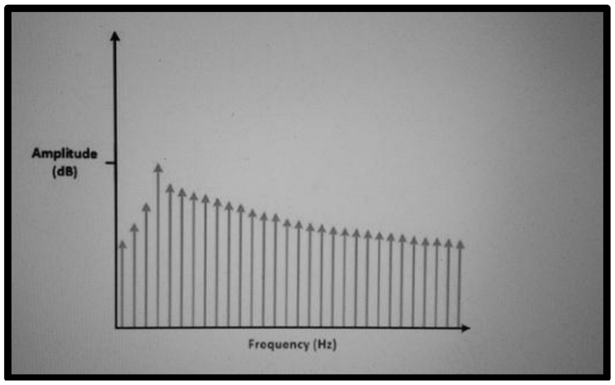


Fig 1.7:-with Spectral Leakage

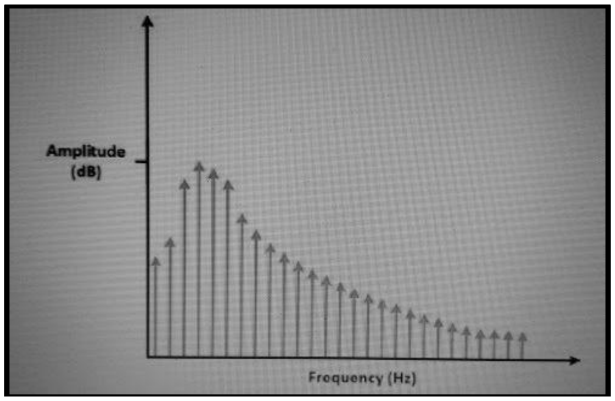


Fig 1.8:-after minimizing spectral leakage

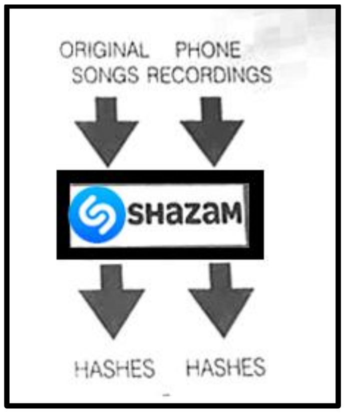
In windowing the discontinuity at end points is minimized by multiplying time with a fixed size window with amplitude varying evenly towards edges and makes the endpoints meet. A window consists of a main and side lobe approaching 0. The lower part of side lobe reduces spectral leakage instead increases the bandwidth of signal.

The choice of window function is based on following frequency estimations of signal:
If the amplitude of the frequency is important than position of component then window with wide main lobe is chosen. If the spectrum of the signal is flat uniform or no window is chosen. If no window is used rectangular window of equal height is chosen. The Fourier Transform is applied for every 0.1 second therefore it forms a rectangular window function. Window function(f) is multiplied with audio signal(s) to get the frequency of 0.1 second audio. Fourier transform(Sample of audio(a)) =Fourier transform(audio(s)*window function(f))

Downsampling
When we increase the window size in order to boost the frequency resolution its time complexity increases. So, there is a method called down-sampling in which we can decrease the window size and maintain the frequency resolution at the same time. Down-sampling is performed by taking the average of the samples and grouping it into a single sample. After resampling the audio consists of only frequencies between 0 and 5khz which is the crucial part of the audio which is considered by Shazam.

feature extraction
Shazam's working algorithm
Fundamentally, it is an algorithm which take sound and gives us the data. We can take original songs as input for this algorithm and get the output data in the form of hashes which is a small part of data which can be used to relate directly back to the original song and also we can take recording from phones as input for the algorithm and get

hashes as output, just like we got with original songs. Core property of this algorithm is ,if original song and phone recordings are same then these two different set of hashes will be mostly equal, which means it takes two different input sounds and gives us hashes that are same if input sounds are same similarly like humans do.



Shazam algorithm looks at features in the file that are unique to that particular audio. Shazam algorithm takes the entire spectrogram data and turns into a blob of points which is called as constellation map. If we take any song and run it through this algorithm entire noise is removed and set of points are generated as constellation map that uniquely represents that particular song and no other song will not have all those points in the same pattern. This is the first step of Shazam's algorithm.

The points which we got, directly relate to that particular song and if we were to combine all these points we would have a unique fingerprint for entire audio file. But, one constraint is that all of these are dependent on time. We only have any meaning from these points if we look at them in relation to the very start of the song. For example in the figure 2.1 the point which is shown exactly occurs at 4 seconds. If we would like to change this recording and start recording at a different point of time then the X-co-ordinates would no longer match up. It would no longer be useful just to compare the points directly to each other and we would not be able to make it match based on the music so this says that fingerprinting whole audio file won't help us match segments of the original audio file. Shazam allows us to hold our phone at any point of time and take a 5-second clip of the song rather than recording the entire song to find out what it is. So, partial/sub fingerprints are made instead of full fingerprints that are invariant to time.

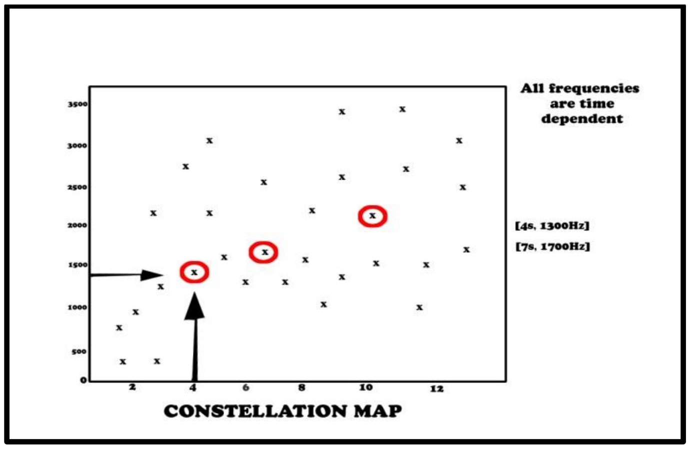


Fig 2.1:-time-frequnecy resulting points

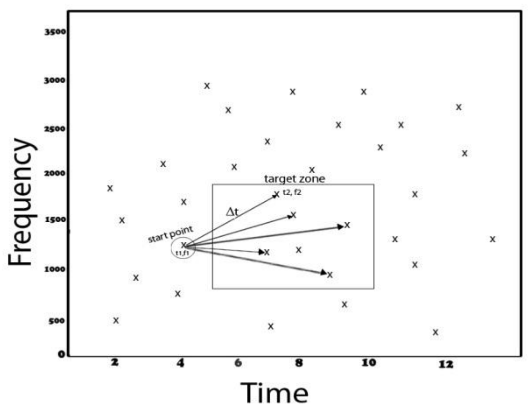


Fig 2.2:-Combinatory hash generation

So, here comes the combinatorial hash generation in simple words it means that pairs of points together in constellation map. Shazam's algorithm goes through every single point and looks at the target zone of points in front of it. So, points that are closely related in time are likely to be captured when we hold up our phone for a 5-second segment we might capture the point which is shown in the figure 2.2 going forward probably all those points in target zone as well. So, small segment of entire audio file it looks like a pair of points between a start point and a target zone and computes hashes (hash tokens). Hashes are simply the line segments between any two notes in the song that are related by time. Now, we can save time invariant hashes specified by exactly 3 numbers i.e., starting(freq1) and ending(freq2) frequency points (note values) and time between those two note values(Δt). We can store these hashes as a very compact data structures.

If we take all of these hashes and store them as tiny bits of data which says that we can take a clip of this song that is a very small segment from the middle and if we analyze the song in the middle we still have the same line segments that we had before and we can now run the algorithm and still we can get that compact data structures. There is still a time based relationship between them so they generate exactly same hashes.

Let us suppose that we have an audio recording "A" and we have a snippet/small excerpt of that recording called A'(A- prime). Then, we can say that A' is a subset or a part of A. If we run a shazam algorithm on A, we get set of hashes relate to that song, and similarly if we run this algorithm on A' which is a subset of "A" the hashes we get are subset of hashes of the "A". This is the primary part of Shazam's algorithm.

Let us consider an entire song as shown in the figure 2.3 which would give me a whole bunch of numbers when they are run by the algorithm. If I take a small clip out of that song only some of those numbers would be in that small clip. But, that set would actually be in this small clip and bigger clip at the same time and it allows us to search the database for those numbers and find the bigger clip from the smaller clip.

Freq1, freq2, Δt

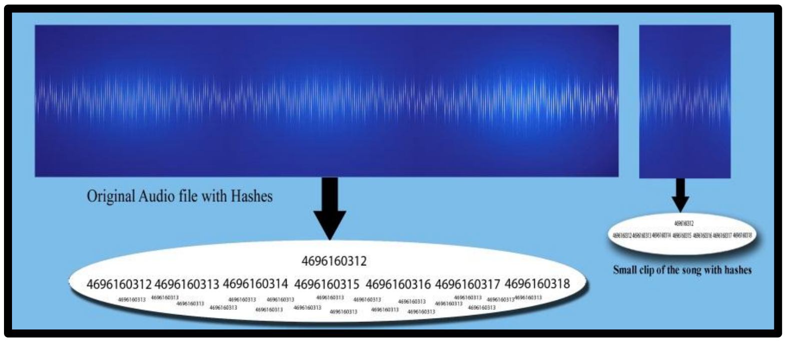


Fig 2.3:-Analog signal of entire audio and clip with generated hashes

We can pack that data down into a 32 bit integer that is 9 bits for frequency of the start point and 9 bits for frequency of the point in target zone and 14 bits for the difference between the frequency points that between the start point and target zone. We can make it really small and compact and save within our database. As, we can see that very little information is encoded in these hashes. This is called the address generation. These addresses are logically associated with time of the start point in the song and Identification number of the song called couple. If this is applied for all the frequency points we will be having the addresses for entire audio file. Now, the couple is coded as 64bit integer that is 32 bit integer for each in the couple. Now, a table(hashing table) is created in the form of array for all these 64 bit integers which consists of index of array as 32 bit integer and catalogue of 64 bits integers address for all the couples .

Create table hashes (hash int (32) unsigned not null, pos int (16) not null, file varchar (255) not null default ‘’ engine=innodb default charset=utf8;

So, a table is made called hashes which consists of columns ‘hash’ which is 32 -bit unsigned integer and ‘pos’ which is the position of the hash in the original audio file and ‘file’ which is the name of the file which we try to get from a search query.

When we want to search for a small segment of the song, all we need is to take the recording of the song in the phone and run the algorithm on it to get hashes that is 32-bit numbers.

```
SELECT *, count (*) FROM hashes WHERE hash in
    (unordered hashes- bag of 32bit numbers)
GROUP BY file ORDER BY count (*) DESC Hash pos file;
```

So, the database consists of all the estimated audio fingerprints and the idea is to find the song through a small clip of that particular song. This can be done by running the algorithm on the small clip and generating a fingerprint for that and comparing this with the original song. This can be done by superposing the small clip fingerprint with that of the original audio fingerprint. This has to be done for each and every song until there is a ideal match of all the songs. There is a possibility that the match could happen at the very end of the song. So, it has to compare that fingerprint with the next songs if doesn’t match. Sometimes, the ideal match is 90% only because the rest of the loss rate is due to the external noise and poor performance of the microphone. So, 90% matching indicate equality rate above threshold. Though this process looks simple, it requires a lot of execution time. Since, Shazam consists of more than 40 million songs it does it in a different and efficient manner that is by introducing the concept of hashes.

Time Coherency

The existent capacity of the Shazam search is that it looks whether delta time between frequency points subsist in the song rather than looking for each and every frequency point.After ,all the filtering of results is done,we have songs that matches the small clip of the song. But, the time consistency is yet to be verified between the frequency points in clip of the song and the filtered songs. If the frequency points in the original song and clip are time consistent then it has to satisfy this relation.

Time of the frequency point in the original song=Time of frequency point in the clip + beginning time of fragment of the song that matches the clip. Out of all the filtered songs, we select the audio file with maximum time consistency. The hashes used in the query are unordered set of data, but we can see that the query does not consider the time about this data shows up in the original audio file. In that SELECT query, some of the hashes are picked from the original song but they could be in completely different order than they were in the original song.

Shazam algorithm actually performs one final step to ensure that all the hashes are not only in both files but all the hashes are in order. We have represented group of hashes using the shapes in figures to track them visually.

If we have a big list of hashes in the sample matches the big list of hashes of what we are searching for, then we can say that sample is a part of that entire audio file. As we can see that query searched for the hashes in an unordered set these hashes could be scattered throughout the song, they could be inexact as shown in the figure 2.4.

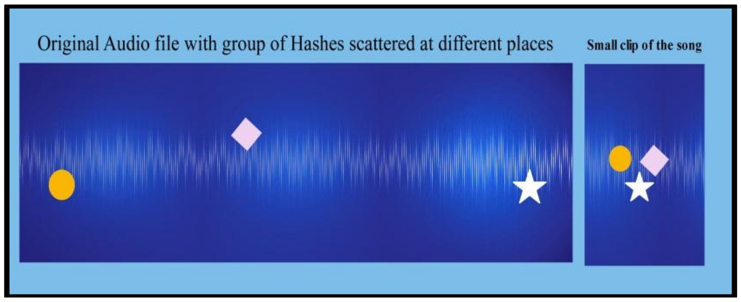

Original Audio file with group of Hashes scattered at different places
Small clip of the song

Fig 2.4:-hashes scattered at different places

As shown in fig 2.5,Let us illustrate this by taking a sample of song which is taken by the phone and put it on y-axis and taking the original song and placing it down on x-axis and making a scatterplot of where the hashes lie between these two songs. Let us consider that the hashes are from the same song as shown in the fig 2.5 , we can look at their positions in the figure ,if all of the hashes shown in the figure happen to be in same order, we would get a distinct pattern (in our example a diagonal line)Iin the fig 2.6 we can see that all hashes line up perfectly along the diagonal. Now, we can say that these are certainly the exact same files since they occur at exact same positions in both of these files in the exact same order.

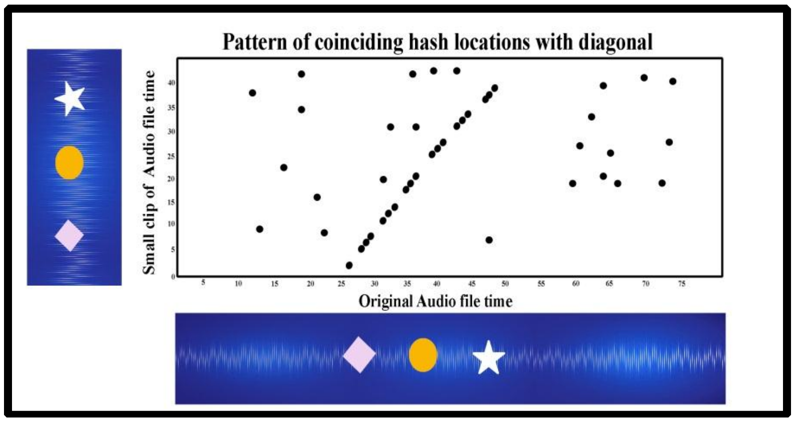
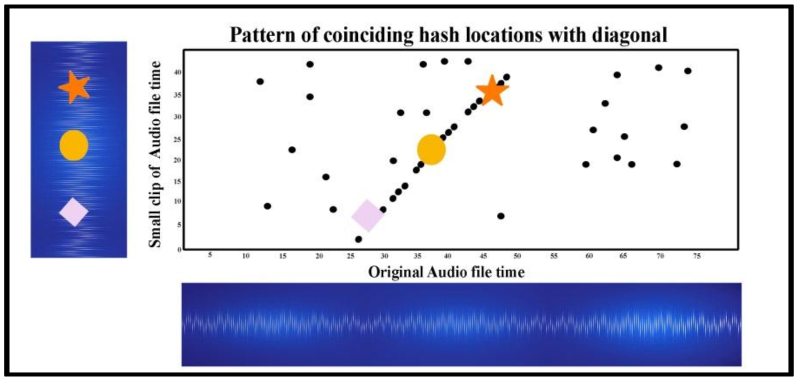


Fig 2.5:-Coinciding hash locations with a diagonal



In the figure 2.7 we can see the differences in the time offsets which can be done by taking the points and subtract each pair of points from each other, ten we would get a spike indicating that there is a match.

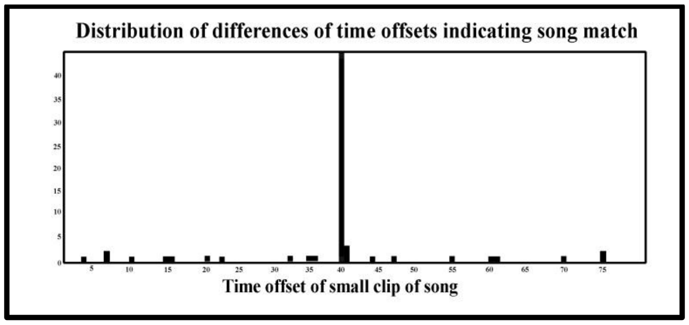


Fig 2.7:-Distribution of differences of time offsets indicating song match

Post Processing
Normalization:
We apply CMN (Cepstrum Mean Normalization) which reduces the distortions and dispersions of energy caused due to loudness variations. Frame energy is taken as input variable and for performing normalization we need to calculate the peak energies of subsequent segments by suing formula
Ee(k)=gamma Ee(k-1) +(1-gamma) E(k)

Where Ee is peak energy at kth frame, Ek is the energy of kth frame.

Conclusion:-

The computational time and external noise is the challenging thing faced by Shazam which has been handled remarkably well and made this application a big success. In the earlier days, it could match only the songs which has been played on a radio or any music player like stereo etc. But the founders say, that the SHAZAM could also identify the songs which are recorded live in any concert. They were unnoticeable earlier due to the pitch alterations and voice modulation by DJs'. Now a days there are many versions for the same song so called remix and cover songs ,which can also be identified through this algorithm. It can tell which version has been played on the phone. The Shazam application also gives you a map showing the recent searches, detailed song description, references to the third-party services. Everyday, a song gets updated in the database of Shazam and it can recognize the song of almost every language and now the developers has managed to improve the application in every criteria like the user interface, machine learning implementation etc.

References:-

1. Christophe Kalenzaga, "How does Shazam work?" May. 23, 2015. [Online]. Available: http://coding-geek.com/how-shazam-works/
2. Tarun Agarwal," Pulse Coded Modulation and Demodulation", Chief Customer Support Officer at Edgefx Technologies Pvt Ltd.[online]. Available: https://www.elprocus.com/pulse-code-modulation-and-demodulation/
3. Pedro Cano and Eloi Battle, Ton Kalker and Jaap Haitsma, Journal of VLSI Signal Processing 41, 271–284,2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands, DOI: 10.1007/s11265-005-4151-3
4. David, Learn, Digital Audio," How to Normalize Audio – Why Do It? Everything You Need to Know". [online], Available: https://www.learndigitalaudio.com/normalize-audio.
5. (Heinrich A. van Nieuwenhuizen, Willie C. Venter, & Leenta M.J. Grobler, 2011)
6. (Cano P., Battle E., Gómez E., de C.T.Gomes L., & Bonnet M., 8 aug 2005)
7. (Grosche, Müller, & Serrà†)
8. (Wei-Ho Tsai, Hung-Ming Yu, & Hsin-Min Wang, 1669–1687, 2008)
9. (Boris D. Kudryashov, Anton V. Porov, & Eunmi L. Oh)
10. (T. Blum, June 1999; T.Blum, D. Keislar, J. Wheaton, & E. Wold, June 1999).