



Shazam is a mobile application that was launched in 2008 to identify songs based on a short sample of sound. The speed and effectiveness by which Shazam was able to identify songs made it the leader in the market and led to it being acquired by Apple in 2017 for a reported $400 million.

The key issue for Shazam in the early days was that, unless the sample was being played from a vinyl disk, most of the tracks were coming from a digital file (mp3, audio CD etc) while Artists were using analog recordings, which were not represented by bits. The recordings were then digitized, stored and played by electronics devices (IPods, computers, etc.). For the purposes of this report, we will omit the discussion on the process of transforming analog sound to digital. The rest of the report is therefore written on the assumption that digital sound has already been generated.

Before explaining the algorithm itself, some concepts need to be understood beforehand. Starting with the basics: a musical note can be represented as a combination of frequencies (as shown in figure 1), so a song can be represented as sheet music or in a spectrogram (as shown in figure 2).

|  |  |  |
| --- | --- | --- |
| Resultado de imagen de musical note frequencies | Imagen relacionada | |
| figure 1. | | figure 2. |

To transform a function of time into a function of frequencies, a mathematical function called the Discrete Fourier transformation (DFT) is used (notation shown in figure 3).

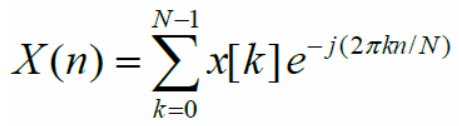


figure 3.

In other words, if the Fourier transformation is applied to a piece of sound, it will return the frequencies (and intensities) for this sound, which can then be represented in a spectrogram.

Expanding the formula above:

1. “N” is the size of the **window: the number of samples** that the signal is composed of:

The standard unit of time in digital music is 44.100-**sample windows** per second. The number comes from a theorem first proposed by Nyquist and Shannon, which states that to digitize a signal from 0 Hz to 20 kHz (since humans can only hear sounds from 20Hz to 20kHz) one needs to take at least 40.000 samples per second.

1. X(n)” represents the ‘n’ **bin of frequencies:**

The DFT gives a discrete spectrum, and the bin of frequencies are intervals between samples i.e continuous frequencies can become N discrete bins. For example, if the sample rate is 100 Hz and the DFT size is 100, then there are 100 points between 0 Hz and 100 Hz. Therefore, the entire 100 Hz range is divided into 100 intervals, such as 0-1 Hz, 1-2 Hz, and so on. Each small interval, say 0-1 Hz, is a frequency bin. The example can be formulated as follows:



-’x[k]’ is ‘k’ sample of the audio signal

A final example might help clarify the functions: the DFT formula must be applied 4096 times on a sound with 4096 samples. Applying the nth bin formula, the sample will have a standard audio sampling rate at 44.1kHz, as the frequency resolution is 10.77 Hz (except bin 0, which is special). So, the first iterations the example would be as follows:

- A single time for n = 0 to compute the 0th bin frequency. In this case, the 0 bin represents the frequencies between 0Hz to 5.38Hz

- A single time for n = 1 to compute the 1st bin frequency. In this case, the 1 bin represents the frequencies between 5.38Hz to 16.15Hz

To get the N bins, 2 \* N^2 operations are required. Is there a faster way to do this transformation?

Yes, with the Fast Fourier Transformation (FFT), using the Cooley-Tukey algorithm. The Cooley–Tukey algorithm computes the DFT using a divide and conquer approach. The idea is that, after shuffling our elements, we divide the N-sample window into 2 N/2-sample windows, then compute the FTT (recursively) for the 2 N/2-sample windows. Finally, the algorithm computes FFT for the N-sample windows from the 2 previous FFTs (as shown in figure 4)

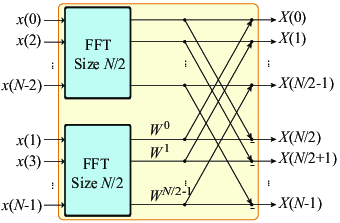


figure 4.

In the “FFT Size N/2” image above, the pairs of elements (for example, x(0) with x(2)) are combined, after which the pairs of pairs are combined (as shown in figure 5 below):

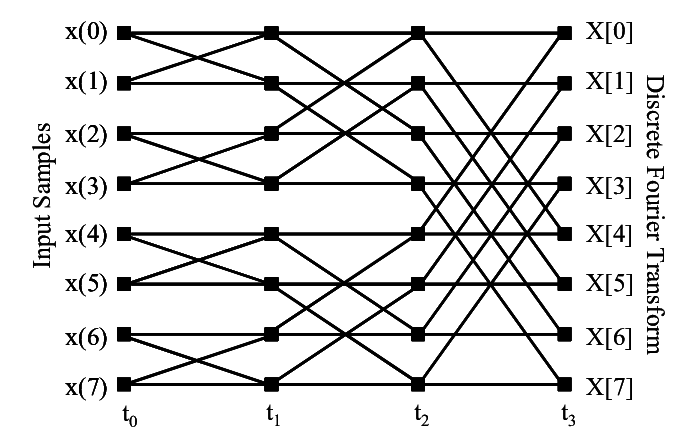


figure 5.

Each combination is called a “butterfly”, and the graphing for all the pairs will is known as a “butterfly diagram”. Each butterfly is a small DFT that, by using this algorithm, avoids the need for complex matrix operations. Coming back to the DFT , we see that it requires the sum of the entire space:

A picture containing object, watch

Description automatically generated

With the Cooley algorithm, we “split” the space into butterflies. As can be understood from figure 5, only 7 “weaker” butterflies can be used instead of 8 DFT operations, which explains why FFT Cooley-Tukey algorithm is faster than classic DFT.

Moving onto the cost of the FFT, the input is divided into two parts of N/2, and the division and result takes O(N) time, so: T(N) (total time) =2 \* T(N/2) (time for N/2) + O(N). The total running time is therefore O(N\*log(N)). In that sense we required 340 times less operations than with DFT, making the transformations feasible (and important) if we are working with mobile’s processors, which is the case with Shazam (at least in its early years).

The Fourier Transformation does have a peculiarity however. It must be applied on only one channel, which means that if sound sample is stereo, it needs to be first transformed into mono. Without delving too much into the terminology, Shazam does this transformation with a very intuitive idea: by taking the average of the right and the left speakers.

This leads back to the spectrogram, which is a visual representation of the spectrum of frequencies of a signal that varies with time. The vertical axis goes from the deepest sound to highest while the horizontal axis is time. Although the representation can seem to be 2 dimensional, it is actually 3 dimensional: the higher the noise, the redder the spectrogram (as can be seen in figure 6).

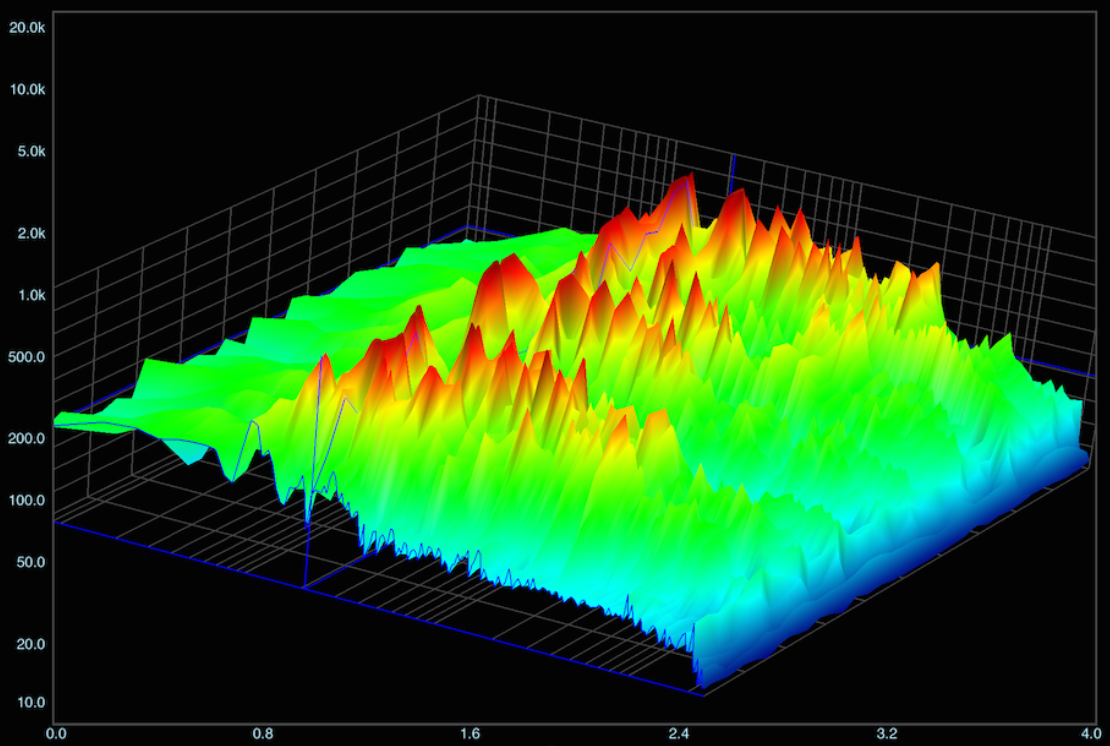


figure 6.

The logic behind using spectrograms is simple: if two songs share the same spectrogram, they are in fact the same song. The problem with this approach is that it is not very fast, so Shazam could not sell their services as a mobile application. Avery Wang, Shazam’s co-founder, came up with the idea of selecting only key points of a spectrogram and comparing them with key points of spectrograms in a large database in order to identify the song that is being played in the environment. Naturally, if there is less to compare, the operations are faster. Avery wanted to follow the same method that the police uses to identify people using their fingerprints, and that is why the method was named “Audio Fingerprinting”.

How are these key points selected? Since each key point is a local optima, in a 3D representation (figure 6), the local optima will be red peaks. To aid our understanding however, let's use a 2D example that Wang used in a paper (figure 7):

|  |  |
| --- | --- |
|  |  |
| figure 7. | |

To summarize, since Shazam needs to be noise tolerant (because the recorded sample can have quality issues depending on the recording environment), only the loudest notes are kept. Keeping only the most powerful frequencies isn’t a very good strategy either as that leads to “ drum sheet music” (figure 8) i.e the matches found are not the same song but instead have the same drum partition.

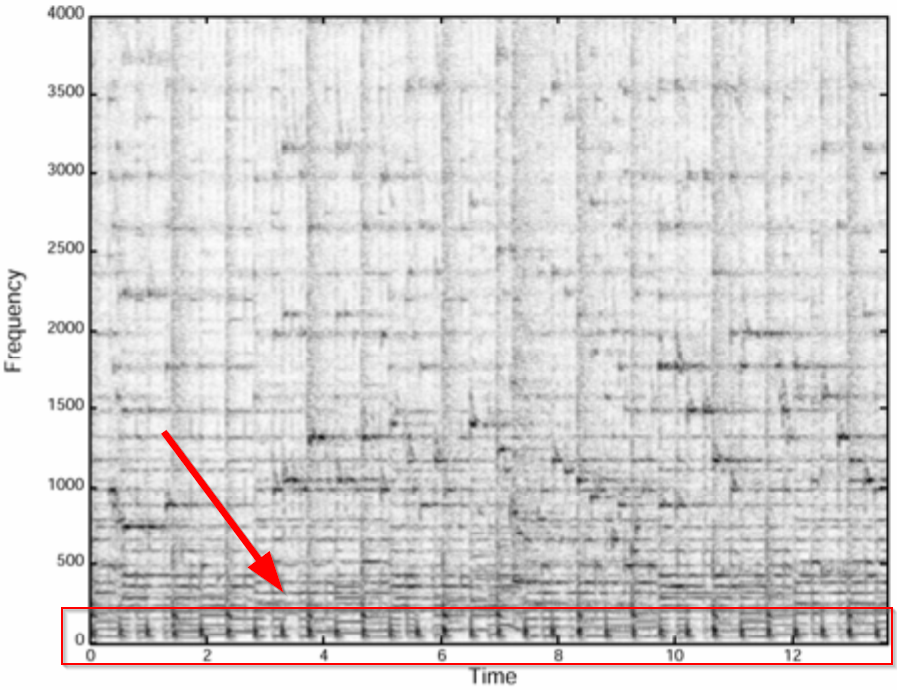


figure 8.

It should be that the number of peaks selected needs to be balanced so as to not end up with too much or too little peaks.

Once the key points were selected, Shazam thought that if they got tuples of frequency and time and compared the tuples to audio recorded by the client, they had the perfect algorithm to match songs. In reality however, the process was not as simple as that.

If we look at the spectrogram in figure 7, we see there are tuples such as 240 Hz/3.9 seconds, 1300 Hz/3.9 seconds, and 2600 Hz/3.9 seconds (figure 9).

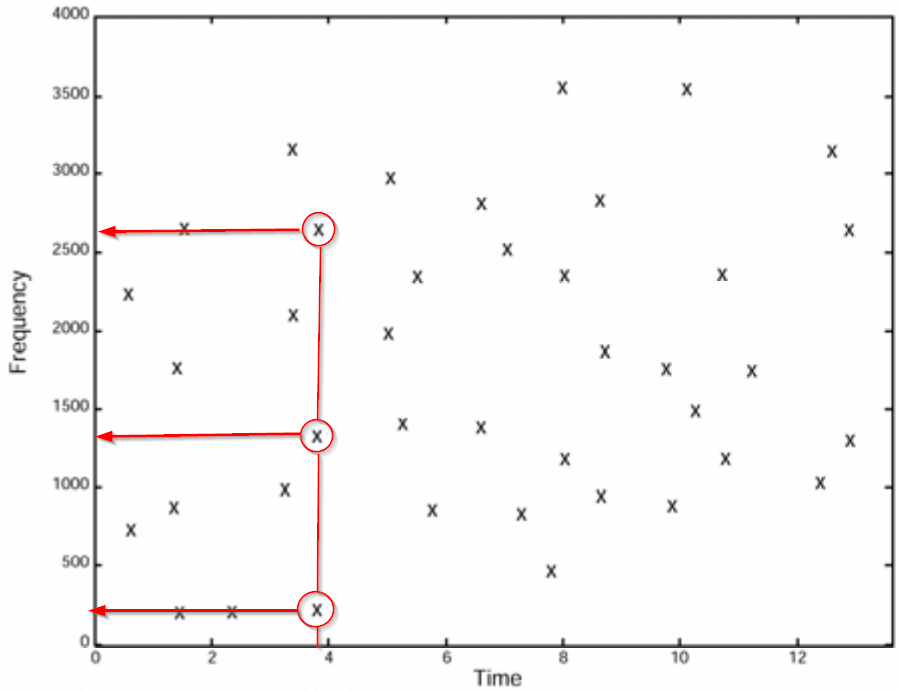


figure 9.

If these tuples were thought to correspond exactly to the recorded audio, the first problem would appear when the first tuples were compared: to have a match, the frequencies 240 Hz, 1300 Hz and 2600 Hz must be found, but they also need to be within 3.9 seconds. Since the person using the app can start recording somewhere in between, say 30 seconds after the song started, the tuples of songs in Shazam’s database would never match since they start in the first seconds of the song of the song.

The approach is not wrong in and by itself, but the amount of money and time required to find the client’s song with the approach of frequency/time would be extremely high, and unfeasible from a business perspective, so this reasoning was discarded.

Wang and his team almost gave up in finding the optimum algorithm, but they soon discovered the solution. Instead of having a tuple of frequency/time, the team figured out that they should create tuples with three values: the first frequency, another frequency that is close to the first (so the first two values will be the frequencies of two key points) and the third the difference in seconds that separate them. Following this approach, in figure 7, the tuples would be : 1650 Hz/2600 Hz/0.1 second, 200 Hz/1000 Hz/1.35 second, 1000 Hz/1300 Hz/1 second, and 1950 Hz/2300 Hz/1 second, etc. (figure 10).

Although the triple value approach would mean that Shazam had to store 10 times more information, it was worth the effort because they finally had an algorithm that worked.

This tuple approach is actually called a Hash function.

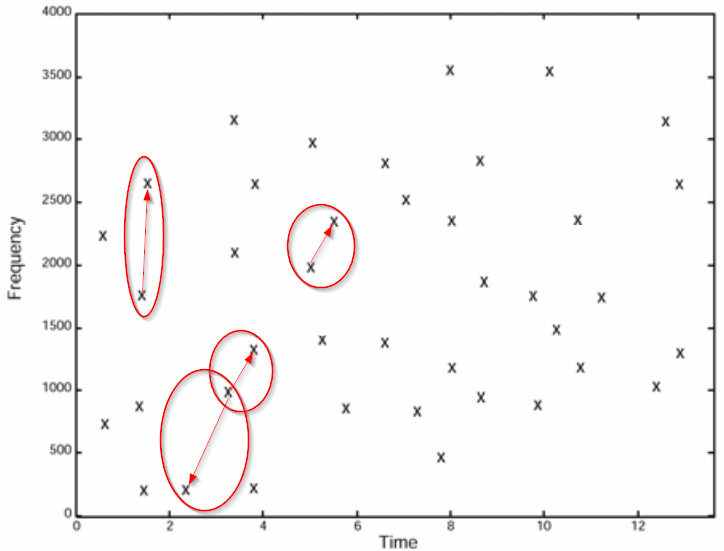


figure 10.

In the next section, we are going to explain the Hash function more in detail.

**Hash function:**

Hashing is the process of transforming an input into a hash value, usually in the form of a fixed size string of text, using a mathematical algorithm called a hash function. For Shazam, hashing acts as an indexing and retrieval framework that accelerates the speed and efficiency of its software. For a hash to be useful, the same inputs should always produce the same hash value, while even a slightly different input will result in a completely different hash value output. This property gives hash values a very important characteristic, which is the ability to easily distinguish between extremely similar inputs. In addition to the distinguishable aspect of the hash values, its compressed nature allows ease of storage, since it only takes a minimal amount of space. This helps in creating an enormous database of hashes, representing distinct inputs with minimal storage size.

**So how is hashing useful for Shazam?**

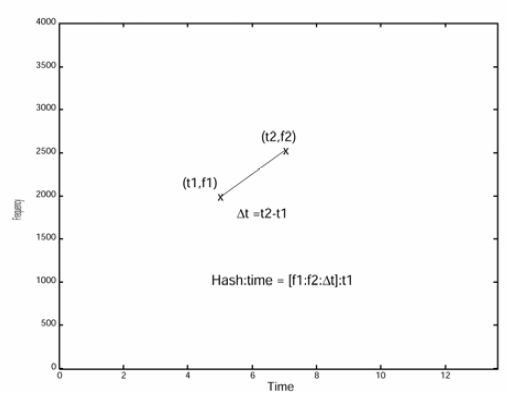
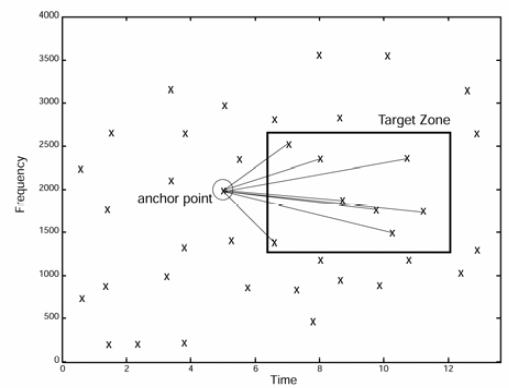
Shazam’s algorithm should be able to take an input from a remote microphone and run this input on its database to find a match. This match will return the song’s name and the metadata gathered about it, such as the artists name, the song’s name, the album and even the lyrics with the links where to listen to this song, all this in a couple of seconds. How can Shazam compare this 4-5 second audio sample with a database consisting of hundreds of millions of songs and return all within the frame of 5 seconds? To be able to run the app effectively, storing the actual songs and the metadata of every song should be compressed in a way to make this more efficient. This is where hashing comes into play.

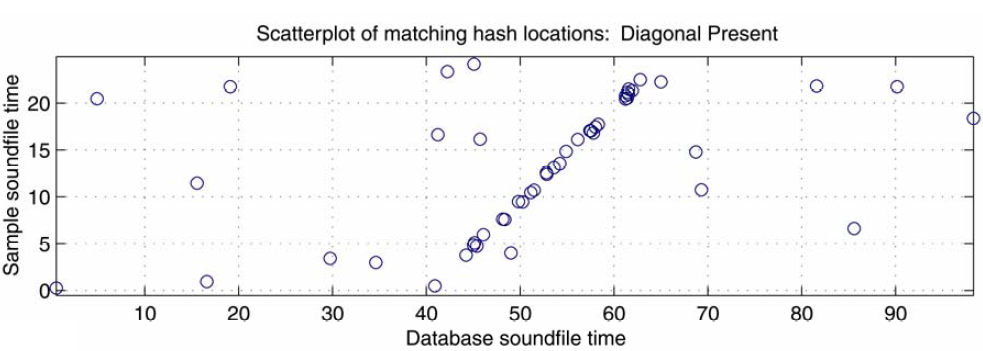
Shazam uses the original’s songs audio fingerprint as inputs that, within that input, contains all the required metadata. Knowing that each song has a unique audio fingerprint as previously explained, this fingerprint is now used as an input for the hash function to produce and store the unique hash value output. “The fingerprints from the unknown sample are matched against a large set of fingerprints derived from the music database. The candidate matches are subsequently evaluated for correctness of match.” (Wang)

However, the question remains how a short-recorded sample have the same hash value as the entire song? For that to happen, shouldn’t the sample recording start from the exact same moment of where the audio fingerprint of the original song is stored?

According to Wang, “The temporal locality guideline suggests that each fingerprint hash is calculated using audio samples near a corresponding point in time, so that distant events do not affect the hash. The translation invariant aspect means that fingerprint hashes derived from corresponding matching content are reproducible independent of position within an audio file, as long as the temporal locality containing the data from which the hash is computed is contained within the file. This makes sense, as an unknown sample could come from any portion of the original audio track. Robustness means that hashes generated from the original clean database track should be reproducible from a degraded copy of the audio.”

To face the problem of degradation, Shazam uses a hashing technique called “fast combinatorial hashing”. Fingerprint hashing is produced from the simplified spectrogram (the constellation map). Points chosen from the map will be assigned to act as anchors within certain time frames from the original song’s spectrogram. These anchor points correspond certain target zones on the map, and the anchor points will be paired with each point within its specific target zone. These pairs will consist of the two different frequency values associated with each point and the time difference between them. Moreover, each hash input is also containing the time from the beginning of the recorded sample and the anchor point of that targeted zone. The illustration of the anchor points and its target zone, along with the distance measurement between frequency points can be seen in the following charts.

These operations are carried out for each track, producing a list of hashes that will be stored in the database. The matching of constellation pairs between the anchor points and the points in the target zone of the constellation map accelerates the speed of matching hashes. “By forming pairs instead of searching for matches against individual constellation points we gain a tremendous acceleration in the search process.” (Wang).



Shazam’s algorithm measures anchor points and their distance from the beginning of the songs with matching hashes. Therefore, for shazam to give out a match, as discussed before, if the matching hashes have the same distance from the beginning of the song then they know for sure it corresponds to that particular song. In the scatterplot above, the y-axis is a representation of the moment where the hash occurs in the user’s recording. The x-axis illustrates the time at which the hash occurs from Shazam’s database. Hence, when a diagonal line is formed, this indicates matching of hashes form the user’s file and Shazam’s database.

Theoretically, there is a lot of potential for Shazam’s algorithms and it has served the company well. The next section discusses the industrial applications for Shazam’s algorithms and processes outside of its main domain.

**Value Creation**

Shazam is one of those pioneering pieces of innovative technology that has revolutionised the music industry. It has created value for different stakeholders, not just the users who are able to benefit from quickly identifying a song. As of data from December 2014, in terms of searches attributed to Shazam, there are over 500 million users of their service who on average search 20 million times about songs, artists and the information behind them as a direct result of Shazam (Thompson, 2014).

Shazam has also contributed directly to digital music providers by enabling access to their services by including links to those providers through song searches. Users can then purchase the songs. Around 5-10% of the searches result in a direct purchase so Shazam can be attributed over $300 million in annual digital sales for providers such as Apple music (Dignan, 2014) (Bostic, 2013). These referrals from Shazam drive a significant amount of revenue for such music providers.

However, the real value is the data generated and captured by Shazam. After 17 million searches, Shazam’s systems are now more aware of which songs, artists and music genres are trending. Artists or recording agencies can plan their artist album launches and song releases accordingly. The search data from Shazam is more valuable than googling a song because if someone is trying to identify a song based off just hearing it vaguely a couple of times, it demonstrates genuine interest which is a more valuable metric than a simple search. As a result, most music labels use this data to present and pitch their songs to certain radio stations and channels to target their advertising in a more targeted manner (Thompson, 2014).

Most of the industry application scenarios are usually particular to the identification aspect of the algorithm, where the fingerprint of the audio can be used to retrieve not only the original version but similar audio recordings as well.

**Barriers to Entry**

The market potential for music fingerprinting is large but before entering such a market, careful considerations must be made as there are significant barriers to entry that exist (Benschop, n.d). These include:

● The majority of music fingerprinting algorithms are patented and optimisations of such algorithms in existing industry applications are protected by trade secrets. Therefore, a licence must be negotiated for all the relevant patents and significant time and effort must be put into developing the software or program to produce similar matching results. While difficult, it is possible to write an algorithm that does not use any of the patents.

● Use-cases of audio/music fingerprinting require a huge database of music recordings (such as in the case of Shazam). While fingerprints can be taken or made without obtaining a whole copy of the original music, if in the future additional fingerprints are required, this may not be possible without possessing or accessing the entire original musical recording – for which, permission and usage rights must be obtained before making copies or accessing the music.

● Any optimisation or improvement to the fingerprinting method would require the re-recording of the entire catalogue of audio fingerprints.

● While it is tempting to consider purchasing existing audio fingerprints to apply to a use-case, this may not be possible as there isn’t a publicly available or standardised industry format for these fingerprints.

● The copyright status of music fingerprints themselves is unknown hence there are risks involved however there are ways to mitigate this.

**Proof of Ownership**

One of the more obvious use-cases for such algorithms is in identifying the owners of audio recordings. Content distributors or broadcasting networks need to know whether they have the rights to broadcast the audio to their consumers. For example, for any radio station broadcast, a complete list of audio recordings can be generated, and this can ensure that the appropriate royalty payments are made to the artists who own the rights to the music .

The best example of this are for YouTube and even Instagram, which applies this process to scan music in the background of its videos and then ascertain whether the user who uploaded the content owns it or not. When the content is not owned by such users, the system notifies them of the issue asking it to be taken down due to copyright infringement. This proves to be invaluable in managing the problem of music piracy (DataArt, 2017).

A further use-case is when advertisers can use this algorithm to monitor radio and TV broadcasts to ensure that their commercials are being broadcast without fail and as agreed.

**Meta-data Management**

A digitized piece of music or audio recording usually contains several pieces of content information or metadata. This includes describing the audio in terms of its harmonic, rhythmic and melodic qualities. In addition to this, metadata such as the recording and composition process such as the year of publishing, performance date and whether the music was live or a studio recording is kept track of. Services like Spotify go further and provide information such as the artist biography, song lyrics and concert information (Cano, Batlle, Gomez, Gomes & Bonnet, 2005)

Other than simply identifying the audio, there are other value-added services that can be employed such as storing the metadata of a song or piece of music in a database after which through audio fingerprinting, this information or metadata can be retrieved for a previously unknown recording. The best open source example of such a service is MusicBrainz (<https://musicbrainz.org/doc/About>).

Such services could be very useful for people with different priorities such as:

● Musicians who might want to identify the different instruments used in a recording

● Common users who would information such as the genre, title, album and year

● Sound engineers who might want information about the recording process of a song

**Forensic Audio and Law Enforcement**

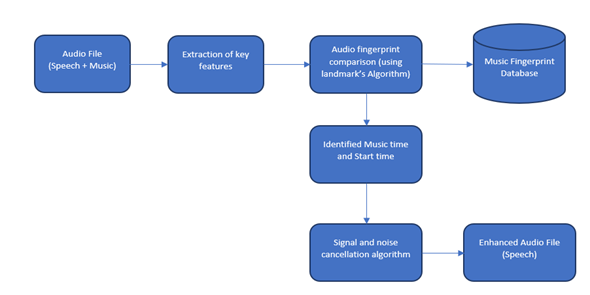
Another highly interesting use-case where audio fingerprinting can play an important role is the proposed application in forensic audio enhancement. In a lot of law enforcement audio recordings, especially when the audio has been obtained covertly, there are usually cases of loud background music or videos being played while the person is speaking. This is especially the case when the person knows they are being monitored, hence they attempt to drown out their voice using music or television audio to make it impossible to decipher (Alexander and Forth, 2005).

The key task for forensic audio enhancement is to reduce or eliminate the background music or sound and bring the desired level of attention to the voice of the speaker. This is a difficult problem which requires the following key measures:

● The exact moment in the recording that the song or video begins playing

● Identification of the original song or music being played

Once these have been obtained a noise and distortion-robust algorithm can be used to remove the background music while leaving the voice of the speaker intact. The schema below by Alexander and Forth, 2005) provides an idea of how this proposed application would work.



**Copyright and Audio Fingerprinting Patents**

To protect itself, Shazam has had numerous patents assigned to its name. The site Justia has a list of some of these patents which can be accessed here:<https://patents.justia.com/assignee/shazam-entertainment-ltd>.

However, unlike commercial solutions such as Shazam, there are several open-source solutions such as MusicURI, The Echo Nest and AcoustID that are also available.

The key differences lie in the methods and processes used by the algorithm. There are 3 main different audio fingerprinting methods that currently exist (Nieuwenhuizen, Venter & Grobler, n.d):

● Methods that use multiple sub-bands such as the Philips Robust Hash (PRH) algorithm, which is more robust against distortions in sound.

● Techniques that use features based on a single band such as the spectral domain such as Shazam’s AudioID algorithms as previously described

● A combination of sub bands or frames which are then optimised through training such as Microsoft’s Robust Audio Recognition Engine (RARE) that employs Hidden Markov Models (HMM)

Shazam’s method employs the single band using the spectrogram approach as previously mentioned, this spectrogram is then divided into smaller sizes called windows or frames. The algorithms that employ a similar method typically involve how much the frames overlap, how the fingerprint is defined in the frame as well the storing, searching and matching processes of these fingerprints (Nieuwenhuizen, Venter & Grobler, n.d).

There are several different ways to do each of these processes and while it is difficult to achieve the same level of accuracy and speed as Shazam’s algorithm, there are different methods and processes one could use to achieve similar results by also employing any of the 3 main types of audio fingerprinting methods.

**Sources:**

Li-Chun Wang, A. (2003). An Industrial-Strength Audio Search Algorithm.

Retrieved from: <https://www.ee.columbia.edu/~dpwe/papers/Wang03-shazam.pdf>

Bekele, A. (2016). Cooley-Tukey FFT Algorithms.

Retrieved from: <http://people.scs.carleton.ca/~maheshwa/courses/5703COMP/16Fall/FFT_Report.pdf>

Burrus, C. (2012). The Cooley-Tukey Fast Fourier Transform Algorithm.

Retrieved from: <https://pdfs.semanticscholar.org/c423/b4635f3e166d9d25dd3a25a36500436f12d4.pdf>

Altozano, J. (2019). ¿Cómo sabe Shazam qué canción está sonando? [Video].

Retrieved from: <https://www.youtube.com/watch?v=OE4gcdjFbmc>

How does Shazam work | Coding Geek. (2019). Retrieved 1 October 2019, from http://coding-geek.com/how-shazam-works/

Retrieved from:<http://coding-geek.com/how-shazam-works/>

Cano, Batlle, Gomez, Gomes & Bonnet (2005). Chapter 17: Audio Fingerprinting: Concepts And Applications. Retrieved from:<http://www.mtg.upf.es/files/publications/0e9cd9-Springer05-pcano.pdf>

Nieuwenhuizen, Venter & Grobler, (n.d). The Study and Implementation of Shazam’s Audio Fingerprinting Algorithm for Advertisement Identification. Retrieved from:<https://pdfs.semanticscholar.org/7e6d/dfc8dac9fea7d98745e8dd486ab5eb39c359.pdf>

Alexander and Forth (2005). 'No, thank you, for the music': An application of audio fingerprinting and automatic music signal cancellation for forensic audio enhancement. Retrieved from:<https://www.kfs.oeaw.ac.at/publications/iafpa_abstracts/nr11_alexanderforth_revised.pdf>

# DataArt (2017). Global Technology Consultancy DataArt Shares Insights Impact of Audio Fingerprinting on the Music Industry. Retrieved from:<https://www.dataart.com/pressroom/press-release/global-technology-consultancy-dataart-shares-insights-impact-of-audio-fingerprinting-on-the-music-industry>

Benschop (n.d). Could music fingerprinting be copyright infringement? Retrieved from:<https://lennartb.home.xs4all.nl/fingerprint.html>

Thompson (2014). The Shazam Effect. The Atlantic. Retrieved from:<https://www.theatlantic.com/magazine/archive/2014/12/the-shazam-effect/382237/>

Bostic (2013). Shazam on 20 percent of US iPhones, generates $300M in annual digital sales. Retrieved from:<https://appleinsider.com/articles/13/03/01/shazam-on-20-percent-of-us-iphones-generates-300m-in-annual-digital-sales>

Dignan (2014). The business of Shazam: Why an Apple iOS partnership would be a boon. Retrieved from:<https://www.zdnet.com/article/the-business-of-shazam-why-an-apple-ios-partnership-would-be-a-boon/>