Volume 4, Issue 2, February 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Study on music Recognition Technology-Shazam

Vijayalakshmi A*, Aastha Hissaria, Abhishek Jaisinghani
Computer Science Department
Christ University, India

Abstract*— Shazam is a mobile phone based music recognition technology that takes a sample of music recorded using a microphone and finds the matching track. The service provided by this app will take a short sample of the music and identifies the song. The search for the music that is to be recognized is performed by comparing key points in the signal's spectrogram. This article explains how the application works, what mathematical support the application runs on and as well as the areas in which the same algorithm can be used. To give a very casual explanation about Shazam we could say that this application chiefly through the inbuilt microphone of the device, records 10 seconds of the song and eliminates the disturbances, giving the end result of the name of the song, a link to a website that offers to purchase the song and also saves the song details in your tags for future reference.

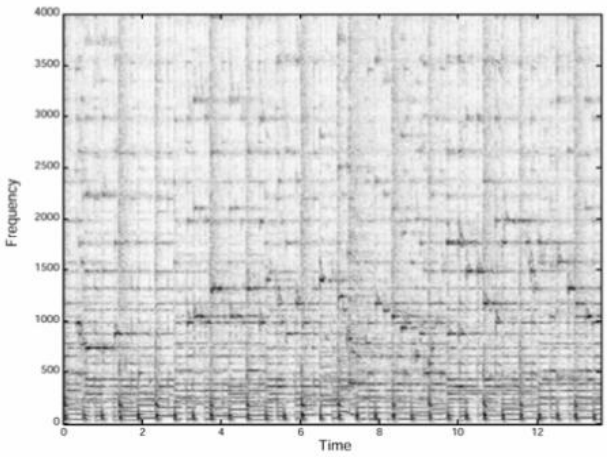
***Keywords*— shazam, music recognition**

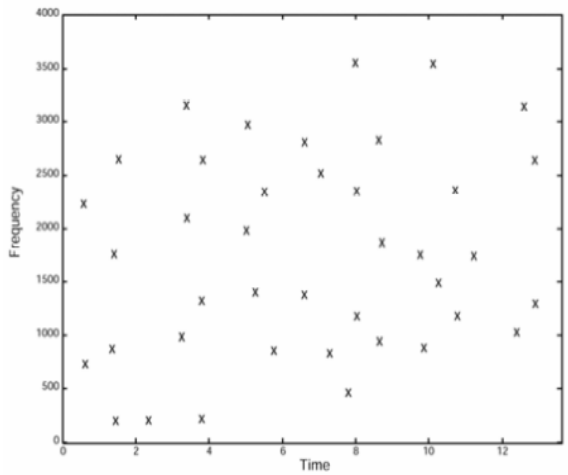
I. INTRODUCTION

Shazam relies on fingerprinting music constructed with the help of spectrographs. The application fingerprints a catalog of music and saves these fingerprints in their database. Then the user who is interacting with this application records 10 seconds of the song though the microphone on the device and the 10 second sample is fingerprinted, this fingerprint is then uploaded onto Shazam's service and is then checked for a matching fingerprint in their central database.If a match is found the details and a link to purchase the same is returned to the user otherwise an error message is thrown to the user[1].This article explains the terms fingerprinting and how a spectrograph is plotted. The main aim of the work is to explain how shazam makes use of the algorithm and mathematical calculations to create a disguise of a comprehensive looking application

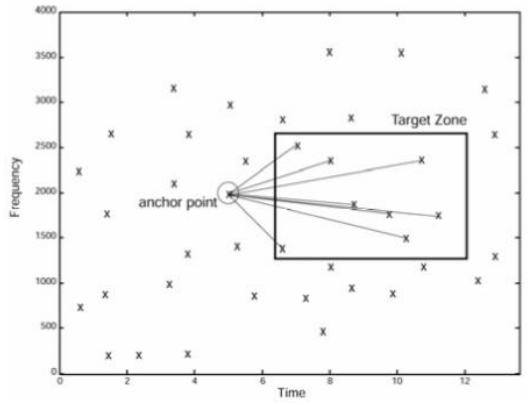
II. WORKING OF SHAZAM

To understand the working of Shazam and how the hash table is computed we first need to understand what a spectrogram is and how the frequencies are plotted on the spectrograph. Each audio file can be represented as a time-frequency graph called a spectrogram. The spectrogram is a three dimensional graph: one axis representing the concentration of a given frequency at an explicit point of time, the x-axis being represented as time and the y-axis as frequency. A horizontal line would represent a continuous pure tone and a vertical line would represent an instantaneous burst of white noise. Each dot created where the horizontal and vertical lines meet represent the volume of a certain sound at a certain time in the song. A very significant dot means the specific sound or frequency is louder than the ones that are less significant. Shazam doesn't store the entire song in their database instead they store only concentrated sounds in the song, the time of their occurrence and at which frequency point.





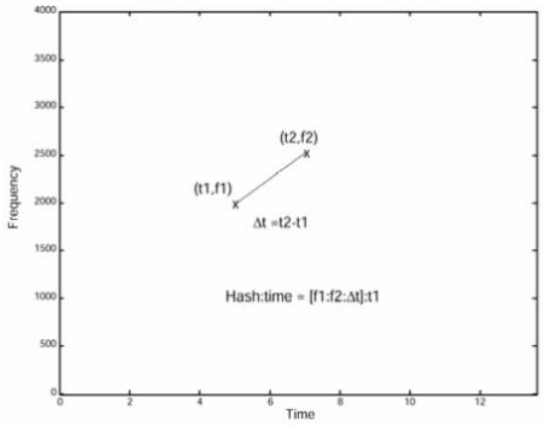
In the database to store the files in a resourceful manner to make it easy to index, the application choose some of the dots from the simplified spectrograph called anchor points and zones in the surrounding area of these anchor points called the target zone



Now for each anchor point that lies inside the target zone an hash is created that is the combination of the frequency at which the anchor point is located (f1) + the frequency at which the point in the target zone is positioned (f2) + the time difference between the time when the point in the target zone is situated in the song (t2) and the time when the anchor point is situated in the song (t1) + t1.

To simplify the computation equation for hash tag table the equation looks something like this:

((f1+f2) + (t2-t1)) + t1.



After this, they will store each hash produced like this in the database in a hash tag table.

So, this is basically how the hash is computed and it serves as the core of their program as their entire algorithm and search is based on the computation of a hash table, it is created for each song in their database as well as the recorded sample in a matter of seconds.

To explain how they match the recorded sample with their entire database to match the song they firstly fingerprint even the recorded sample by computing the hash and each hash that is generated will be examined for a match with the hash of each song in their database.

If a match is found you will have the time of the hash from the sample (*th1*), the time of the hash from the song in the database (*th2*) and implicitly the ID of the song for which the hash matched. Basically, *th1* is the time since the beginning of the sample until the time of the sample hash and *th2* is the time of the song hash.

Now, they will draw a new graph called scatter graph. In the graph, the horizontal x-axis represents the time of the song in the database and the vertical y-axis represents the time of the recorded sample. On the X axis we will mark *th2* and on the Y axis we will mark *th1*. The point of intersection of *th1* and *th2* will be marked with a small circle. The magic happens now: if the graph will contain a lot of duos of *th1*'s and *th2*'s from the same song, a diagonal line will form. The basic indication behind the diagonal line is that the rate at which the peaks (the small crosses from the simplified spectrogram) in the database song appear will be the same rate in which the peaks appear in the recorded sample, so if you pair these times, the coordinates on the scatter graph will grow constantly (to the right-top of the graph) as the time passes on both axes.

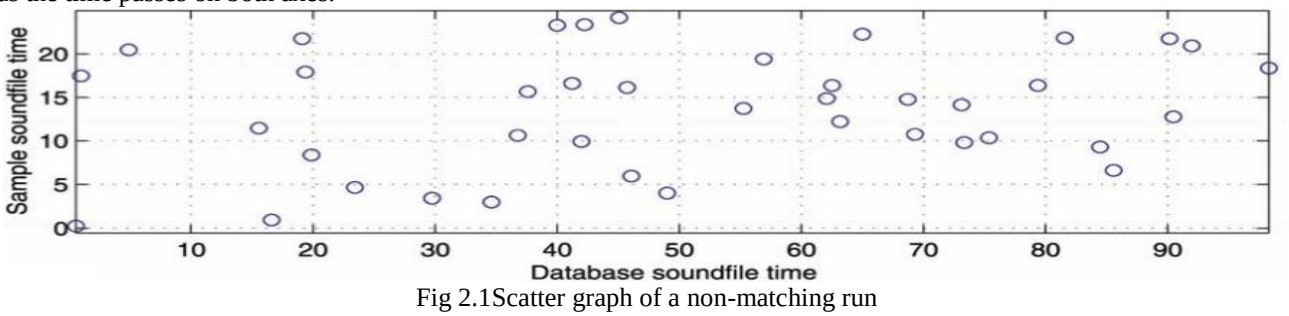


Fig 2.1Scatter graph of a non-matching run

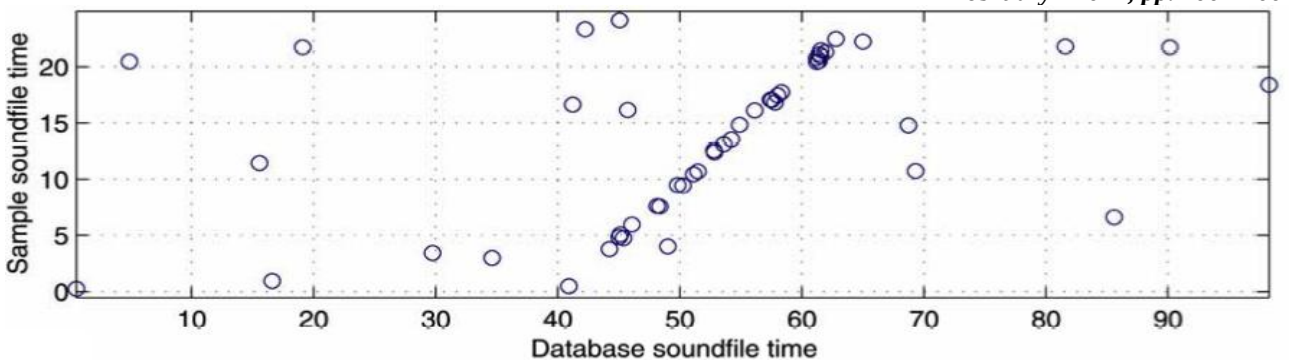


Fig 2.2 Scatter graph of a matching run

The final step is that, they will calculate a difference between *th2* and *th1* (*dth*) and they will plot it in a histogram. If there is a match in the graph plotted, then there will be a lot of *dths* with the same value, because, basically, subtracting the *th2* from *th1* will give the difference between a point in the original song and the same point in the recorded sample. This will result in a peak within the histogram, which will confirm a match.

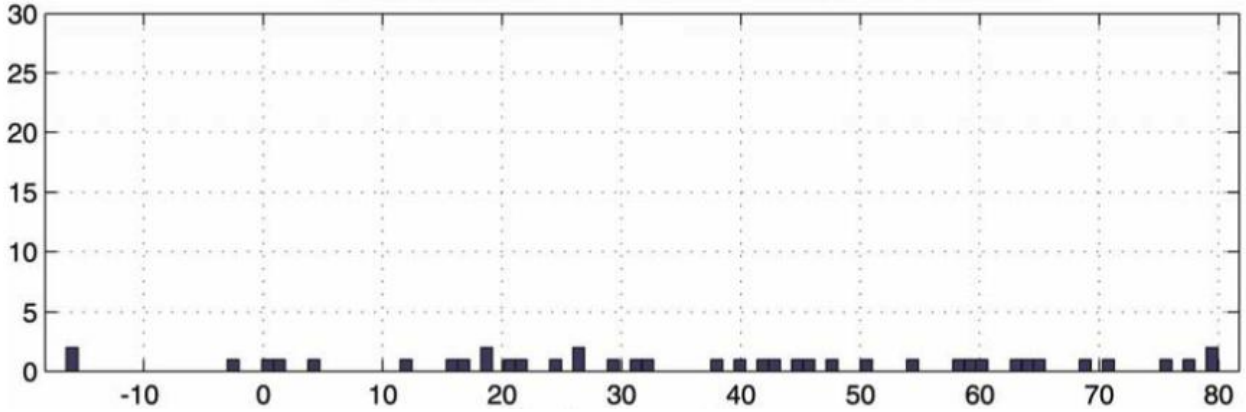


Fig 2.3Histogram of a non-matching run

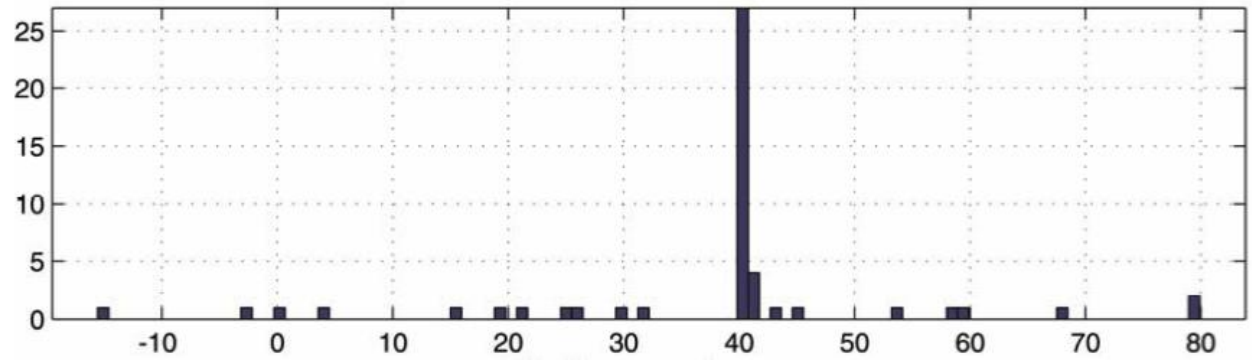


Fig 2.4Histogram of a matching run

This is basically how the spectrogram works and how the hash table is computed. Shazam uses this method to make the unique acoustic fingerprint for each music file they have in their data base and make it possible to search for the details and ID of any song by simply recording 10 seconds of any song you listen to even in the most crowded places. All these computations and graph making is done within 5 seconds after recording your song so one can imagine the speed at which all the above mentioned computations take place.

III. CONCLUSIONS

This application uses simple calculations that require a simple logic and statistical knowledge to understand makes it possible for a million of its users of to recognize the song they hear no matter where they are and what kind of an environment it is. The Shazam algorithm can be used in many applications besides just music recognition over a mobile phone. Due to the ability of the application to dig deep into noise we can identify music hidden behind a loud narration, such as in a radio announcement. On the other hand, the algorithm is also very fast and can be used for copyright monitoring at a search speed of over 1000 times real-time. The algorithm is also suitable for content-based cueing and indexing for library and archival uses.

REFERENCES

[1] http://laplacian.wordpress.com/2009/01/10/how-shazam-works/.
[2] Avery Li-Chun Wang and Julius O. Smith, III., WIPO publication WO 02/11123A2, 7 February 2002, (Priority 31 July 2000).
[3] Jaap Haitsma, Antonius Kalker, "A Highly Robust Audio Fingerprinting System", International Symposium on Music Information Retrieval (ISMIR) 2002, pp. 107-115.
[4] Wang, Avery Li Chun., "An Industrial Strength Audio Search Algorithm." ISMIR, London : Shazam Entertainment, Ltd., 2003.
[5] Hatch, Wes., "A Quick Review of Audio Fingerprinting." March 2003.
[6] Doets, P.J.O, Gisbert, M. Menor and Lagendijk, R.L., "On the comparison of audio fingerprints for extracting quality parameters of compressed audio." [ed.] Edward J. Delp III and Ping Wah Wong. Delft : Security, Steganography, and Watermarking of Multimedia Contents VIII, 2006, Vol. 6072

