Volume 4, Issue 1, January 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

An Efficient Indexing and Searching Approach for Minimizing Searching Time in Audio Fingerprinting

Mr.Simarjeet Singh Bhatia*, Mrs. Rupali Bhartiya

CSE Dept, RGPV Bhopal

India

Abstract— An audio fingerprint summarizes an audio recording or a song. A song can be recognized or identified by matching its fingerprint to the fingerprints present in a database. This whole process is bifurcated into two parts viz. fingerprint extraction and fingerprint matching. In this paper we will be concentrating more on the second part i.e. fingerprint matching. We propose a new method of indexing and search algorithm. This indexing method for our search algorithm will make it more hybrid and efficient. Theoretically our algorithm provides encouraging results.

Keywords— fingerprint; audio fingerprinting; indexing; hashing; bunching.

I. INTRODUCTION

Today, digital music libraries are becoming popular day by day and are growing rapidly. With their more and more demand it has become quite important to analyse them. The efficiency to find the exact song from a large audio database is particularly a very important task for a number of applications. The audio fingerprinting technique has many applications in broadcast monitoring, automatic generation of playlist of radio, TV, or web broadcasts etc..Another application can be described as searching (retrieving) the name of the artist and title of the song given a short clip of an audio, filtering technology for file sharing and so on. Since an audio contains many features, extracting a representative audio fingerprint is the most important task that describes each song. There are many techniques given by many authors to extract audio fingerprint namely Haitsma and Kalker, M.L Miller, Avery Wang, collectively by Yu Liu, Kiho Cho, Hwan Sik Yun, Jong Won Shin and Nam Soo Kim, Guang Ho-Cha. Haitsma and Kalker introduced the concept of look up table to store and searched the audio fingerprint whereas M.L Miller proposed the concept of 256-ary tree, Avery Wang presented landmark hashing technique, Yu Liu emphasized on discrete cosine transform to improve accuracy, and Guang Ho Cha proposed the concept of inverted file in order to improve the recognition rate. In our method we presume a representation of song by Philips Robust Hash [1] method in which a 3 second interval is represented by 256 sub-fingerprints of 32 bits each and provided this fingerprint our main concentration would be on indexing as well as searching a song from the database. This problem can be characterized by the nearest neighbor search problem of a very high dimensional (i.e. 256*32=8192) space.

II. LITERATURE REVIEW

Haitsma and Kalker introduced an indexing scheme which works on the principle of look up table. A lookup table consisted of all the combinations of fingerprints. The method lets the entries in the lookup table point to the song or songs where the respective sub-fingerprint value occurs. A sub-fingerprint value can occur at multiple points in a song or multiple songs, the song pointers are stored in a linked list. Therefore a single sub-fingerprint can point to multiple pointers at the same time and thus all the 256 blocks of fingerprint block are compared with the entries in database [2].

The first problem in the haitsma and kalker method is that there are 2^{32} entries in lookup table which is practically too large. This problem is solved to some extent in the next work.

The second problem in this method is that haitsma and kalker make assumption that if a signal undergoes a “mild” degradation then also at least one sub fingerprint is error free which is quite unavoidable to consider.

M.L Miller, M.C. Rodriguez, and I.J. Cox [3] proposed 256-ary tree technique for fingerprint search. Each 8192(256*32) bit fingerprint is represented as 1024 8 bit bytes. This technique adopts top down approach. The value of each successive byte determines the next step of where to find the fingerprint of 256 child to follow. The path of the root node to a leaf defines a fingerprint. The searching is done by comparing the first byte of query with the children of the root node by calculating the cumulative error rate with each child node. If the error rate seen so far is greater than threshold then the child is searched. This process is done continuously until the leaf node is reached. After this process the error rate is compared to the best error rate seen so far, if the error rate is less than it then it is considered as best candidate in the nearest neighbour. This technique’s main loophole is same as that of haitsma and kalker method i.e. the size of the 256-ary tree is too large and the depth of the tree is also too big for disk based database search. The second problem [4] in Miller’s method is that it uses probabilistic method based on binomial distribution to estimate the bit error rate in each tree node which leads the correct error up to 85% since it is difficult to find bit error rate in advance. This drawback is removed to some extent in the next work.

Avery Wang introduced the scheme of landmark hashing [5]. The basic operation of this scheme is that each audio track is analysed to find prominent onsets concentrated in frequency, since these onsets are most likely to be preserved in noise and distortion. These onsets are formed into pairs, parameterized by the frequencies of the peaks and the time in between them. These values are quantized to give a relatively large number of distinct landmark hashes (about 1 million in number). Parameters are tuned to give around 20-50 landmarks per second. Each reference track is described by the (many hundreds) landmarks it contains, and the times at which they occur is stored in an inverted index. Similarly to identify a query it is converted to landmarks. This method again suffers from same drawback i.e. it can be seen that it contains about 1 million landmark hashes but again the problem is that to search a audio file it will go to the same position where the audio file is kept and therefore each time when a song is requested it will go through the same process. It can be seen that the problem of accuracy is solved to a good extent but still one million landmarks is very high and thus extends the search domain still to a large area.

Yu Liu, Kiho Cho, Hwan Sik Yun, Jong Won Shin and Nam Soo Kim proposed a concept of DCT [6] i.e. discrete cosine transform based multiple hashing technique which was quite robust in nature and had a good accuracy as well. It worked by including discrete cosine transform phase into its fingerprint extraction technique and making k hash tables in order to store them. As for their algorithm they used 4 hash tables in order to retrieve the song, thereby giving a good accuracy. Although the accuracy is quite good but still there is a major setback in this algorithm i.e. the number of hash tables which are 4. These hash tables at first look to seem small in number but when the number of songs increase then there length increases and so on thereby increasing the complexity of the algorithm therefore it then becomes quite complex to find a song which is present in 4^{th} hash table at a far position. Also this method concentrates more on accuracy rather than speed of search. However, the constraint of accuracy is removed in next work to a large extent.

Haitsma and Kalker method doesn't provide an efficient indexing method also the searching is quite ineffective because of large database, therefore in order to improve efficiency and increase searching speed we need to revise the Haitsma and Kalker algorithm. To overcome the above two problems we reuse the concept of Bunching with the help of Hash Count.

III. PROPOSED SCHEME

The above mentioned problems of stated algorithms can be solved by introducing Hash Count and Bunching into the proposed algorithm. Our main emphasis is on how to increase the speed of searching as well as decrease the search time by a considerable amount.

The proposed algorithm can be explained as follows:

Suppose a song named 's' has to be searched in the database, according to above mentioned algorithms a search is applied to it by first extracting its fingerprint, then those songs are retrieved which are likely to match with the query finger print. After this phase the fingerprint of query audio is matched with the fingerprint of retrieved audio from the database. Now in the best case scenario there can be only one song in the database which matches the first time and so the query finger print matches the very first time or there could be 'n' number of songs which would have the same fingerprint that of the query audio [7]. After the exact song is matched it is fetched from the database as the result. The main problem as we stated is that the retrieved song is coming from that place where it is present in the database and next time when it is searched again it will come from that place only which will take the same time. In order to remove this flaw in Haitsma and Kalker method, in the proposed scheme we will give a Hash Count to the searched song. Each time when the song is searched and retrieved its Hash Count will be increased by a number and the song with the maximum number of Hash Count will be kept at initial or top position in the database, thus when the fine search is carried out for that song it can be retrieved as early as possible because it will be found at the top position and not on the same position in which it was kept at the first time. Thus with this process the overload can be reduced to some extent for random access of disk.

There can be another scenario with the database containing songs. What happens when the database gets updated with a new song? Since there is a new song then its hash count will be zero, also it could be possible that query fingerprint would exactly or near to exactly match this song. In this situation the algorithm will again have to scan till the end of the database for that song which is now updated. To solve this problem we introduce a further scheme of "Bunching". According to this scheme we will make a bunch of those songs which match with each other near to exact by making a bunch or group of them [8]. Due to this process the song database will contain groups of songs which are similar with respect to the fingerprints, also a song can be grouped according to their genre, pitch, level of noise content, zero crossing rate etc..

An audio fingerprinting algorithm is divided into two phases:

1.) The fingerprint extraction phase.
2.) The search algorithm.

The audio fingerprint is extracted with the help of Philips Robust Hash algorithm. The stages of fingerprint extraction phase are as follows:

a.) **Framing**: - In the first stage of this phase the query audio is framed.

b.) **Fast Fourier Transform**:- In the second stage Fourier transform is taken so as to change the signal from time domain to frequency domain.

c.) **Energy band extraction**:- Out of the result of second stage 33 energy bands are selected between the range of 300hz - 2000hz because this is the range for human auditory system.

d.) Up to this stage **feature extraction** takes place.

e.) Prior to final stage **filtering** is done.

f.) **Threshold**:- At the last stage we get a 32 bit sub fingerprint for every 11.6 millisecond therefore 256 sub fingerprints for every 3 seconds after performing threshold process. Figure 1 below shows various stages of sub-fingerprint extraction

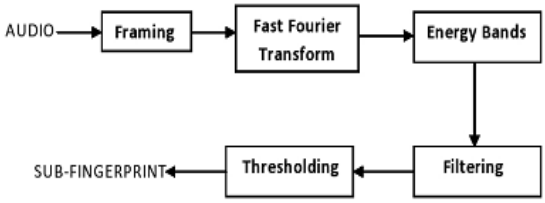

Figure 1: Sub-Fingerprint Extraction Stages

The proposed search algorithm contains two phases as described below:

a.) In the **first phase** all those songs are searched and retrieved which could possibly contain the candidate's fingerprint.

b.) In the **second phase** each matched song's full fingerprint is fetched from the database and is compared with the query fingerprint [9], if the song matches with the query fingerprint then it is retrieved and the song's hash count is now increased by one unit i.e. if the songs hash count is 0 it is now 1 after the search and each time it is increased by 1 unit when it is retrieved after searching. After updating the hash count of the searched song, its position is updated in the database according to its hash count [10, 11], the more the hash count the higher will be its position i.e. the top most position, so whenever this song's query comes next time then it will be retrieved from the updated position and not from the old position [13], which will decrease the time of retrieval thereby increasing the speed of searching. Figure 2 shows the flow chart structure of the proposed algorithm.

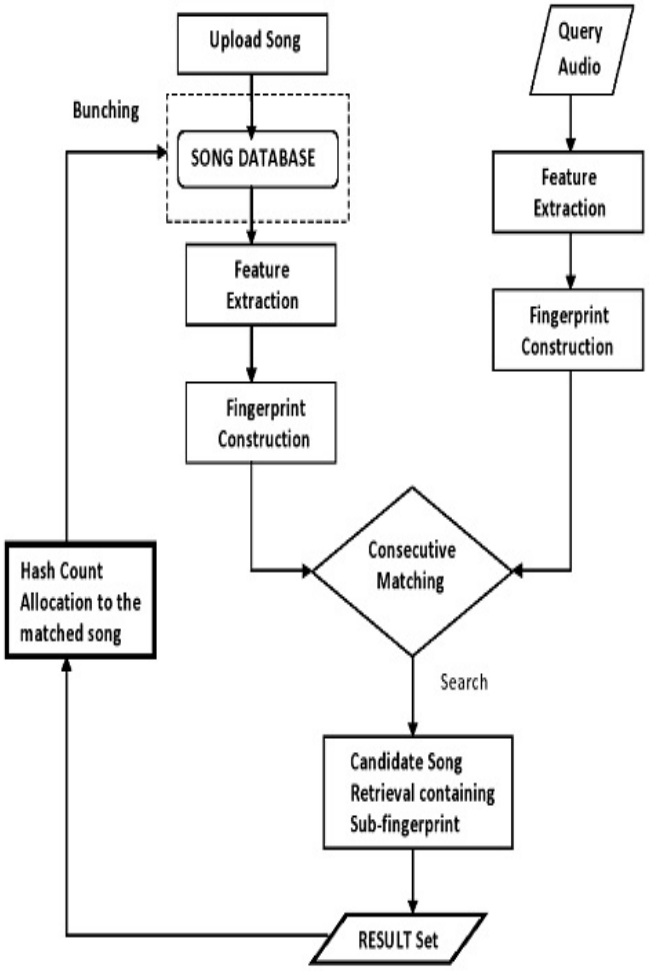

Figure 2. Proposed Hybrid Algorithm Flowchart

Figure 3 shows the song retrieval architecture for the above algorithm. This is the second phase of the proposed algorithm. It can be seen from the figure that a query fingerprint block contains 256 blocks of sub-fingerprints each of 32 bits. A fingerprint might be present at many positions in many songs. This fingerprint is projected in database which then points to the pointers which further point to the position of a sub-fingerprint in a song.

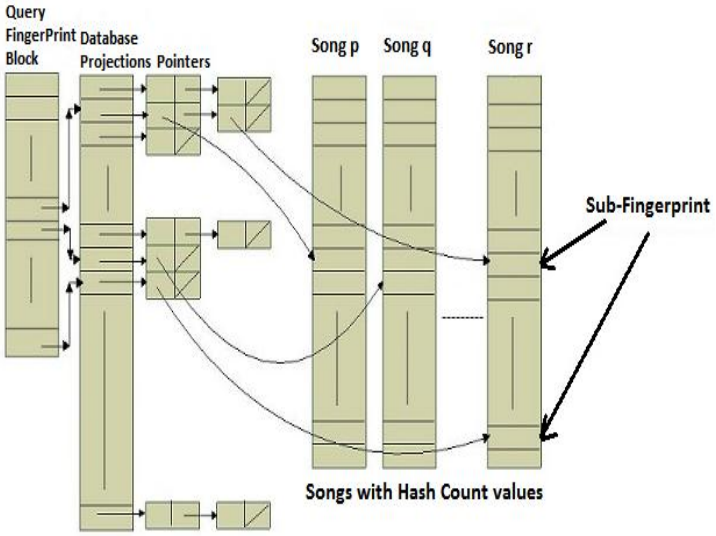

Figure 3.Proposed Retrieval Architecture

IV. COMPARISON OF EXISTING ALGORITHM WITH PROPOSED ALGORITHM

In comparison of Haitsma and Kalker algorithm our algorithm throws light towards new indexing scheme i.e. Hash Count and Bunching. In ideal conditions i.e. when no song is searched the performance of our algorithm will be same as that of Haitsma and Kalker algorithm because to search a song whose hash count is 0 it has to go to its initial position, similarly if the song is searched for the first time then also the algorithm has to go to its initial position. This algorithm is much more advantageous for those songs which would be having there hash counts more than any other song, this can be said so because a song which is searched more number of times will be on higher position in database and would be easily and rapidly retrieved, which is an advantage of this algorithm that if a song has been searched more number of times it would be retrieved in less time.

V. CONCLUSION

In this paper we propose a new indexing scheme for large audio fingerprint databases. This indexing scheme theoretically provides good results for large databases. This scheme is a little more complex than Haitsma and Kalker algorithm as it introduces an additional concept of hashing and bunching but it sheds out advantages with respect to speed and accuracy as well which makes it a useful method for audio fingerprinting.

REFERENCES

[1] J. Haitsma and T. Kalker, "*A Highly Robust Audio Fingerprinting System*," Proc. Int. Symp. on Music Information Retrieval, pp. 107–115,2002.

[2] J. Haitsma and T. Kalker, "*A Highly Robust Audio Fingerprinting System With an Efficient Search Strategy*," J. New Music Research, vol. 32, no. 2, pp. 211–221, 2003.

[3] M.L. Miller, M.C. Rodriguez, and I.J. Cox, "*Audio Fingerprinting: Nearest Neighbor Search in High Dimensional Binary Spaces*," Proc. IEEE Multimedia Signal Processing Workshop, pp. 182–185, 2002.

[4] M.L. Miller, M.C. Rodriguez, and I.J. Cox, "*Audio Fingerprinting: Nearest Neighbor Search in High Dimensional Binary Spaces*," J. VLSI Signal Processing, vol. 41, pp. 285–291, 2005.

[5] Avery Wang "*An Industrial-Strength Audio Search Algorithm*", Proc. 2003 ISMIR International Symposium on Music Information Retrieval, Baltimore, MD, Oct. 2003.

[6] Yu Liu, Kiho Cho, Hwan Sik Yun, Jong Won Shin, and Nam Soo Kim "*DCT Based Multiple Hashing Technique for Robust Audio Fingerprinting*" –IEEE trans. ICASSP, pp.61-64, 2009.

[7] "*Structural fingerprint based hierarchical filtering in song identification*" Qiang Wang, Gang Liu, Zhiyuan Guo, Jun Guo, Xiaoyu Chen-IEEE-2011.

[8] F. Balado, N.J. Hurley, E.P. McCarthy, and G.C.M. Silvestre, "*Performance analysis of robust audio hashing,*"IEEE trans. Information forensics and security, vol. 2,no. 2, pp. 254–266, Jun. 2007.

[9] Chih-Chin Liu, Po-Jun Tsai, "*Content-Based Retrieval of MP3 Music Objects*"- CIKM'01, ACM, pp.-506-511, November 5-10, 2001.

[10] Anonymous ICME submission, "*Audio-To-Tag Mapping: A Novel Approach for Music Similarity Computation*", IEEE trans.2011, pp.1-4.

[11] Arijit Ghosal, Bibhas Chandra Dhara, Sanjoy Kumar Saha, "*Speech/Music Classification Using Empirical Mode Decomposition*"- IEEE, Second International Conference on Emerging Applications of Information Technology, pp.49-52, 2011.

[12] C. Panagiotakis and G. Tziritas,"*A speech/music discriminator based on rms and zero-crossings*," IEEE Transactions on Multimedia, vol. 7, pp. 155–166, 2005.

