A Fast Retrieval Algorithm Based on Fibonacci Hashing for Audio Fingerprinting Systems

Mei Chen[1], Qingmei Xiao[1], Kazuyuki Matsumoto[1], Minoru Yoshida[1], Xin Luo[2], Kenji Kita[1]

[1]Faculty and School of Engineering, the University of Tokushima, Tokushima, Japan
[2]School of Computer Science and Technology, Donghua University, Shanghai, China
{chen-mei, xiaoqingmei}@iss.tokushima-u.ac.jp, {matumoto, mino, kita}@is.tokushima-u.ac.jp, xin@dhu.edu.cn

Abstract - In audio fingerprinting system, the database consists of hundreds of millions of sub-fingerprints. How to find out the most similar audio in the shortest time and use less memory in huge repository of sub-fingerprints is a hot topic of research. Recently, Philips introduced an effective search method based on hash table. In this paper, a fast retrieval algorithm based on the Fibonacci Hashing is proposed as an extension of Philips's method. The algorithm uses Fibonacci Hashing function which can adjust the size of hash table according to capacity of memory, provide a good distribution of hash value, and save memory. The performance of presented algorithm has been evaluated by experiments.

Index Terms - audio retrieval, Fibonacci Hashing, audio fingerprint, hash table.

1. Introduction

An audio fingerprint is a compact representation of an original signal and considered as a short summary of an audio object. Therefore the fingerprint is considered as identification in the sense that it almost uniquely represents the signal. The audio fingerprinting can achieve the monitor of audio content without metadata, which helps to identify an unknown audio clip from database via the Internet, PC, microphone, mobile phone, etc. Furthermore, people also adopt audio fingerprinting technologies to protect copyright of music, and prohibit copyright infringement of songs, and so on.

The rapid growth of music data available on the Internet has resulted in the need for efficient search algorithms. The crucial issues of memory usage and query response time should be taken into consideration in designing fingerprinting system. In [1], Haitsma and Kalker exploited the hash table as index to improve the search speed. Miller et al. [2] use a tree representation to achieve query. In [3], binary search is employed based on array representation.

In this study, we propose a fast retrieval algorithm based on Fibonacci Hashing for the extension of Philips's method, which can save memory and improve the query speed. The remainder of the paper is organized as follows: Hash table is briefly reviewed in section 2. Then, we present the proposed method in section 3. In section 4, we evaluate its performance by experiments. Finally, we summarize the paper in the last section.

2. Fibonacci Hashing

Hash table is an effective data structure exploited to implement an associative array which can map keys to values. Given a key, hash table employs hash function to compute hash value as entry to find the correct value.

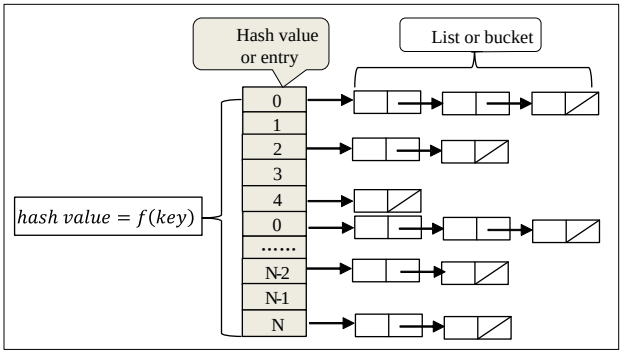

Fig. 1 Structure of hash table

The design objective of hash table is as follows [4]:

A) Each key corresponds to a unique hash value.

B) Hash values can evenly spread across hash table.

In fact, these ideal situations are hardly achievable. Hash collisions [5] that the hash function maps two different keys to the same entry are practically unavoidable. Therefore, hash table implementations must have some collision resolution strategy to deal with such events. A good hash function is also indispensable for improving the performance of hash table. A basic requirement is that the function should provide a uniform distribution of hash values [4] which can also help to reduce the probability of collisions. Fibonacci Hashing [6-8] is believed to provide one of the best hash functions. Its main idea is to find a very special value *C* and perform operations by key to generate the hash values. Fibonacci Hashing function is defined as follows:

$$f(key) = (key * C) >> N \quad (1)$$

The hash values obtained in this way provide a good distribution of hash value in hash table. Here, the value of *C* we opt relates to the golden ratio [9-10].

If the ratio of two positive numbers *a* and *b* is the same as the ratio of their sum to the larger of the two, then the ratio is called the golden ratio. Let $\lambda = \frac{a}{b}$ denote the golden ratio,

and easily get: $\lambda = \frac{1+\sqrt{5}}{2}$. Then the value of *C* is the closest integer with $W\lambda^{-1}$ [11-13].

$$C \approx W\lambda^{-1}, \tag{2}$$

where *W* denotes word sizes and λ^{-1} be the reciprocal of λ. It easily gets the values of *C* for various word sizes, and the results are as Table I.

TABLE I The value of C for various word sizes

W	*C*
2^{16}	40503
2^{32}	2654435769
2^{64}	11400714819323198485

In fact, we exploit $W\lambda^{-1}$ as the value of *C*, and there are the regularities as the following: Each newly added hash value will fall into one of the largest remaining intervals and divide the interval according to the golden ratio.

3. Fast Retrieval Algorithm Based on Fibonacci Hashing

3.1. Hash Function

In Philips's method, a hash table containing all possible 32-bit sub-fingerprints is exploited as the index. Firstly, due to the limited memory, a size of 2^32 of hash table will be usually sparsely placed into some computers. Secondly, the distribution of hash value is non-uniform in hash table and a lot of memory space is out of use, which leads to a lot of waste. For example, the average usage rate of a hash table containing approximately 250 million sub-fingerprints is only 0.058 in Philips's method. In fact, the rate is probably much lower in the practical application. On the basis of these considerations, the paper uses Fibonacci Hashing function.

Here, using (1), *key* denotes a 32-bit sub-fingerprint. Because *key* is 4 bytes, *key* is rightly equal to 2654435769 according to Table I. *N* ($1 \le N \le 31$) is the right shift bit number. The value of *C* mainly depends on capacity of available memory and the amount of songs. The equation (1) performs right shift operation which moves *key* * *C* *N*-bit to right. As a result, different sub-fingerprints of the same top (32-*N*)-bit will be assigned to the same bucket or list. Thus size of hash table becomes 2^(32-*N*).

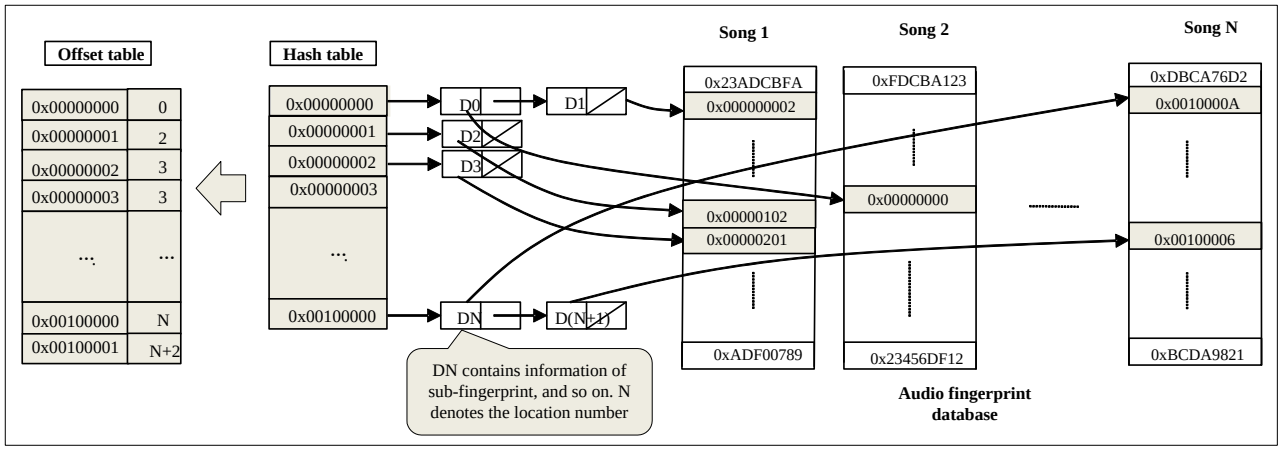

Fig. 2 The model of improved hash table

3.2. Searching

In practical application, the amount of songs may achieve tens of thousands. If all sub-fingerprints are put into memory, it would be infeasible. Therefore, two hash tables are designed in this paper, respectively called hash table (HT) and offset table (OT). OT contains offset information of HT. And HT includes important position information with pointers to a real fingerprint location in audio fingerprint database. Each list node of HT points to a unique real sub-fingerprint location. The structure of improved hash table is shown in Fig. 2.

Their operations are as follows. First, we create HT by using (1). Second, OT is built according to HT. The execution firstly numbers by traversing all list nodes of HT from top to bottom and from left to right. The position number of the first node of each bucket will be selected as offset and recoded to the corresponding entry of OT. Finally, the data domain of hash table is written into file called HTF by the location number order, and hash table is destroyed. When hash table is need, it will read from HTF. So that can save a lot of memory.

The search layout is shown in Fig. 3. The retrieval operations are as follows.

Step1: Extract the sub-fingerprints sequence from query audio clip as candidate sub-fingerprints.

Step2: Exploit (1) to generate hash value.

Step3: Locate corresponding entry and read offset in OT according to the hash value.

Step4: Use offset to read the data of specified rang from HTF and compare with candidate sub-fingerprint.

Step5: Continue comparing their corresponding sub-fingerprint blocks by calculating theirs distance. If the distance

is less than the threshold, it successfully matches. Otherwise, it goes on comparing the next candidate sub-fingerprint from last compared position until the end.

Finally, output the top *n* songs of minimum distance as query results.

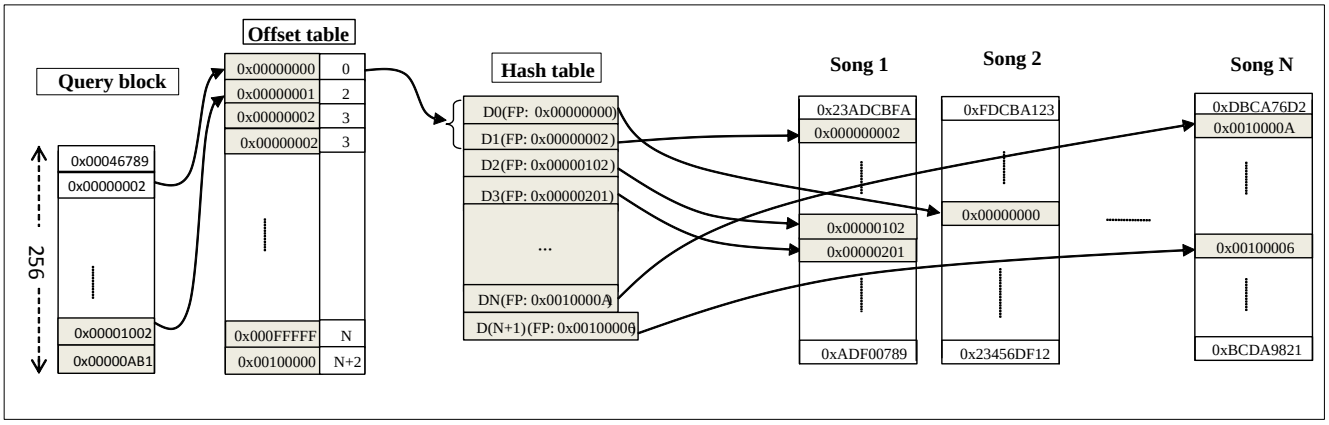

Fig. 3 Search layout

4. Experiments

This section will evaluate the performance of the proposed algorithm. Experiments will be performed from 3 aspects: search accuracy, search time and memory usage.

The database contains 7605 songs in mp3 and wav format from the Internet. These songs we choose are of low distortion. Genres of songs contain pop, classical, folk music, etc. In the experiment, memory is 4G, with a free memory of 1.6G.

4.1. Search Accuracy

In the experiments, fingerprint extraction is based on the Philips's method. First, the input audio signal is segmented into overlapping frames with an overlap factor of 31/32 by a Hanning window. Next, it selects 33 non-overlapping frequency bands from 300Hz to 2000Hz and extracts a 32-bit sub-fingerprint. Furthermore, a fingerprint block is composed of 256 subsequent sub-fingerprints.

We extract about 61million sub-fingerprints from 7605 songs. In the experiments, we randomly select 100 audio clips from 7605 songs in 10 times. The search accuracy can achieve 100%.

4.2. Search Speed

In order to verify the performance of search speed, we perform the experiments of average search time for a song. The results are shown in Fig. 4.

From Fig. 4, average search time stabilizes at about 0.97s with the change of value of *N*. In order to improve the query speed, the paper performs the following improvements. Firstly, it uses hash table as index. For hash table having the property of quick positioning, and its time complexity is *O*(1). And the offset table use offset as hash values which achieve fast positioning. Secondly, the size of hash table can be adjusted according to the capacity of memory by right shift operation. Therefore the whole hash table can be loaded into memory, and search speed becomes much faster.

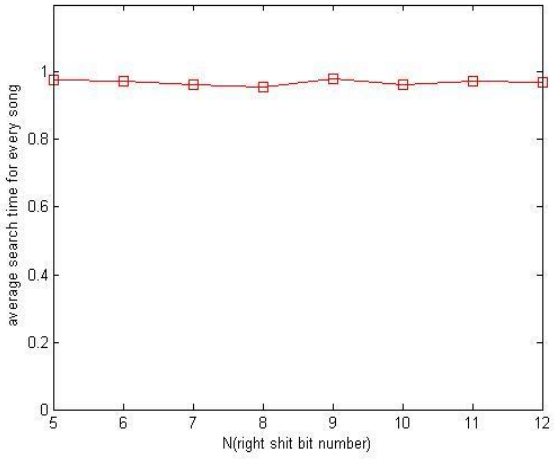

Fig. 4 Average search time for a song

4.3. Memory Usage

We will evaluate the performance of memory usage in this part by exploiting usage rate (*UR*) and average usage rate (*AUR*). Let *UR* represent percentage of used entries in the total entries; *UN* denotes the number of used entries in hash table; *L* stands for the length of hash table. Calculation of *UR* is as follows:

$$UR = \frac{UN}{L}. \tag{3}$$

Let UR_N denote the proposed method, and UR_P stand for Philips's method. The experiments results are shown in Fig. 5.

In Fig. 5, we can find that UR_N is remarkably higher than UR_P, which shows that hash values evenly distribute across hash table. The Fibonacci Hashing optimized utilization of memory.

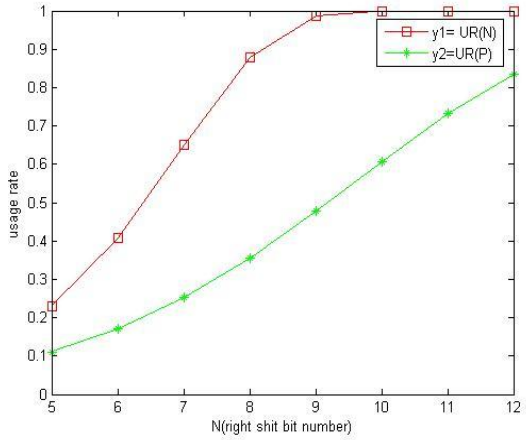

Fig. 5 Usage rate

AUR denotes the percentage of the number of fingerprints and size of hash table. As mentioned above, another factor is the amount of songs in the selection of *N*. If songs are quite many and size of hash table is small, every entry will contain more information. As a result, frequency of comparison will increase. If songs are very few and size of hash table is large, memory will be wasted a lot. Here we use *AUR* to reflect this case. Let *AUR* represent average usage rate; *T* stands for the total number of sub-fingerprints from database; *L* for the length of hash table. Calculation of *AUR* is as follows:

$$AUR = \frac{L}{T}. \tag{4}$$

The experiments results are shown in Fig. 6. From Fig. 6, for 61 million sub-fingerprints, the recommended values of *N* are from 6 to 9 from the standpoint of *AUR*.

As shown in Fig. 5, optimal value of *N* is 8. In terms of 61 million sub-fingerprints and 1.5 G free memory, the most appropriate value of *N* is 8 in the experiments.

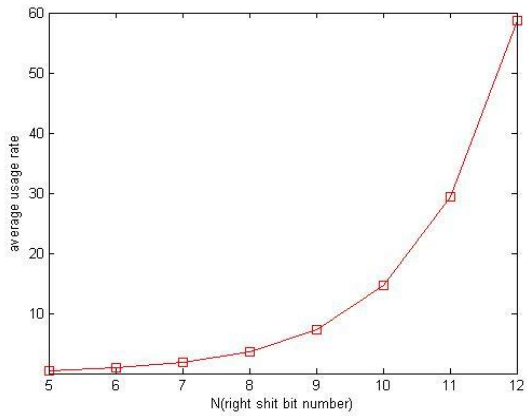

Fig. 6 Average usage rate

5. Conclusion

In this paper, we proposed a fast retrieval algorithm based on Fibonacci Hashing. Firstly, we use Fibonacci Hashing function. This function implements better distribution of hash value and thus remarkably save memory. Moreover, the size of hash table can be changed by bit right shift operation according to the capacity of memory. Therefore, the hash table can be completely put into memory and achieve the high-speed retrieval. Secondly, fingerprint information data is stored in file and read as it is needed. Furthermore, we use position offset as hash value, which achieve fast positioning. In the future, the research will focus on pre-treating audio data before extracting fingerprint and exploring regularity of fingerprinting data to enhance the robustness under of distortion and improve the retrieval speed.

References

[1] J. Haitsma and T. Kalker, "Highly Robust Audio Fingerprinting System," *3rd International Conference on Music Information Retrieval (ISMIR 2002)*, pp. 107-115, 2002.

[2] M. L. Miller, M. A. Rodriguez, and I. J. Cox, "Audio Fingerprinting: Nearest Neighbor Search in High Dimensional Binary Spaces," *Journal of VLSI Signal Processing*, vol. 41, no. 3, pp. 285-291, 2005.

[3] Q. Xiao, M. Suzuki, and K. Kita, "Fast Hamming Space Search for Audio Fingerprinting Systems," *ISMIR 2011*, pp. 133-138, 2011.

[4] I. Damgard, "A Design Principle for Hash Functions," *Advances in Cryptology" – CRYPTO'89*, *Lecture Notes in Computer Science*, vol. 435, G. Brassard ed., Springer-Verlag, 1989.

[5] M. Bellare and P. Rogaway, "Collision-Resistant Hashing: Towards Making UOWHFs Practical," *Advances in Cryptology – CRYPTO '97*, *Lecture Notes in Computer Science*. vol. 1294, B. Kaliski ed., Springer-Verlag, 1997.

[6] F. Aydin and G.Dogan, "Development of a new integer hash function with variable length using prime number set," *Balkan Journal of Electrical & Computer engineering*, vol.1, no. 1, pp. 10-14.

[7] R. I. Duncan, "Application of Uniform Distribution to the Fibonacci Numbers," *The Fibonacci Quarterly*, vol. 5, no. 2, pp. 137-140, 1967.

[8] L. Kuipers, "Remark on a Paper by R. L. Duncan Concerning the Uniform Distribution Mod 1 of the Sequence of the Logarithms of the Fibonacci Numbers," *The Fibonacci Quarterly*, vol. 7, no. 5, pp. 465-466, 1967.

[9] G. Markowsky, "Misconceptions about the golden ratio," *The College Mathematics Journal*, vol. 23, no. 1, pp. 2-19, 1992.

[10] G. Markowsky, "Misconceptions about the Golden Ration," vol. 23, no 1, pp. 1-18, 1992.

[11] J. H. E. Conn, "On square Fibonacci numbers, Etc.," *The Fibonacci Quarterly*, vol. 2, no. 2, pp. 537-540, 1964.

[12] H.L. Montgomery I.Niven, and H.S. Zuckerman, "The theory of Numbers," *John Wiley & Sons*, 1991.

[13] D. E. Knuth, *The Art of Computer Programming*, Fundamental Algorithms, Addison-Wesley, Reading, MA, USA, 1973.