IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 12, December 2016

Fast Audio Fingerprinting System using GPU and a Clustering-Based Technique

Shweta Patil[1], N.R. Wankhade[2], J.V. Shinde[3]

Student, Comp Department, Late G.N. Sapkal College of Engineering, Nashik, India [1]

Asst. Professor, Comp Department, Late G.N. Sapkal College of Engineering, Nashik, India[2, 3]

Abstract: We present our audio fingerprinting system that detects a transformed copy of an audio from a large collection of audios in a database. The audio fingerprints in this system encode the positions of salient regions of binary images derived from a spectrogram matrix. The similarity between two fingerprints is defined as the intersection of their elements (i.e. positions of the salient regions). The search algorithm labels each reference fingerprint in the database with the closest query frame and then counts the number of matching frames when the query is overlaid over the reference. The best match is based on this count. The salient regions fingerprints together with this nearest neighbor search give excellent copy detection results. However, for a large database, this search is time consuming. To reduce the search time, we accelerate this similarity search by using a graphics processing unit (GPU). To speed this search even further, we use a two-step search based on a clustering technique and a lookup table that reduces the number of comparisons between the query and the reference fingerprints. We also explore the trade off between the speed of search and the copy detection performance. The resulting system achieves excellent results on TRECVID 2009 and 2010 datasets and outperforms several state-of-the-art audio copy detection systems in detection performance, localization accuracy and run time.

Keywords: Audio fingerprint, content-based copy detection, fast search, parallel processing, GPU.

I. INTRODUCTION

Audio fingerprint is extracted from spectrograph by a process called feature extraction. Track identification using audio fingerprinting is considered to be one of the fastest and accurate forms of music recognition. Audio fingerprints are compact content based signatures of audio recordings. These audio fingerprints capture highly specific characteristics of a short audio fragment. This feature makes it possible to accurately identify the fingerprint and distinguish itself from millions of songs. The key specialty of audio fingerprinting is to link unknown audio to corresponding metadata, irrespective of its audio format. An ideal audio fingerprinting system should be able to recognize an unknown audio fragment from the database regardless of the noise in the environment. The audio fingerprint system should be efficient. This is attained by making the fingerprints compact and implementing a complex and smarter search algorithm. It also requires the fingerprint extraction process to be powerful. Fingerprints and matching algorithms should result in the same content taken from the distorted audio recording. Fingerprints extract the characteristics of the audio recording in robust and concise format

II. OBJECTIVE

1. To propose a fast and robust audio fingerprinting system using parallel computing.
2. To optimized algorithm searching time using gpu shared memory.
3. To propose search algorithm based on a clustering technique that reduces the number of fingerprint comparisons significantly.

III. LITERATURE SURVEY

Hendrik Schreiber and Meinard Muller presented various improvements for the audio ID system originally proposed by Haitsma and Kalker. Our main observation was that the probability of finding a matching reference sub-print is elevated in the case that multiple consecutive query sub-prints are identical. Which supports for mildly distorted audio files.

J. Haitsma and T. Kalker presented a new approach to audio Fingerprinting. The Fingerprint extraction is based on extracting a 32 bit sub Fingerprint every 11.8 milliseconds. The sub-Fingerprintare generated by looking at energy differences along the frequency and the time axes. A Fingerprint block, comprising 256 subsequent sub Fingerprints, is the basic unit that is used to identify a song. The Fingerprint database contains a two-phase search algorithm that is based on only performing full Fingerprint comparisons at candidate positions pre-selected by a sub-Fingerprint search with reference to the parameters like Robustness, Reliability, Fingerprint size, Granularity, Search speed and scalability.
A.Wang processes the Fingerprint from the unknown sample and matched with a large set of Fingerprint derived from the music database. The candidate matches are

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
Vol. 5, Issue 12, December 2016

subsequently evaluated for correctness of match. Some guiding principles for the attributes to use as Fingerprint are that they should be temporally localized, translation invariant, robust, and sufficiently entropic. 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.

IV. RELATED WOK

A review of audio copy detection systems shows three main approaches to accelerate fingerprint search: binary search, hashing-based search and approximate search. The energy difference fingerprint, where a binary fingerprint encodes the energy differences along the frequency and the time axes, figures among the fastest CBCD systems. The binary representation of the fingerprints makes the search very fast. However, this fast search results in a modest performance compared to other methods. Regions around selected points from the maxima in the Mel-filtered spectrum are encoded to generate binary fingerprints. Compared to, this approach improved significantly the detection accuracy while maintaining a fast search. Local regions of the spectrogram image are transformed into a set of 32 bit vectors, and a classical hash table is used to perform the search. In the Shazam system, several time-frequency points are chosen from the spectrogram. Compact signatures representing peak pairs are then generated to form fingerprint hashes that allow very fast search. Approximate searching techniques such as Locality Sensitive Hashing (LSH) are used in several works to accelerate the search. Wavelets with the largest magnitude are selected from spectrogram, and LSH is used to accelerate fingerprint similarity search. Although more robust than and this system is computationally very expensive. A comparative study in terms of detection accuracy and computation time can be found. The Weighted Audio Spectrum Flatness (WASF) is used as audio features, and LSH is adapted to compute the dissimilarity between two WASF features. LSH is used in many works to accelerate the search, but it is slower than the hashing-based search.
Recently, Graphics Processing Units (GPUs) have been used to accelerate scientific computations. Using GPU to accelerate large-scale applications became easier with NVIDIA's CUDA platform. Several GPU implementations of widely used algorithms such as k-nearest neighbor and LSH are available and can be used for audio copy detection. A GPU implementation of the Metric Permutation Table algorithm speeds up the search of digital images. A GPU implementation of the nearest neighbor search between the reference and query fingerprint is then described in, where the copy detection algorithm has been modified to perform advertisement detection. Compared to its CPU implementation, this GPU implementation improves speed by a factor of 70. Similar GPU based nearest neighbor search is used in search through millions of fingerprints. However, these papers do not include a description of the GPU implementations. A parallel implementation is introduced and tested over a large database of more than 11,600 hours of audio. A GPU is used to parallelize two parts of the system leading to an overall speedup of a factor of 5. The authors also explored the use of three GPUs instead of only one allowing them to further improve performance by a factor of 3 on some parts of the system.
In another work, the computation of the cross-correlation between two audio windows is accelerated using a GPU . The database used to test this algorithm is very small (1 hour), and the GPU lead to a moderate improvement by a factor of 2 (compared to the CPU implementation). In Another technique that reduces search time and complexity is starting from the fact that search time is related to the size of the database, the authors partitioned the database of 100,000 songs into 10 sub-databases. The search then executes 10 independent processes on different machines. It divides the fingerprints database into several parts, and the search algorithm is executed in parallel based on the Message Passing Interface (MPI) standard. Clustering techniques have been used in several works to avoiding exhaustive search. In binary fingerprints of the reference videos are grouped into k different clusters, and only fingerprints that belong to the cluster closest to the query fingerprint are searched to find a match. The algorithm continues to examine other clusters if a match is not found. The problem in using this strategy is the possibility of visiting all the k clusters before a match is found resulting in an exhaustive search.

V. PROPOSED SYSTEM

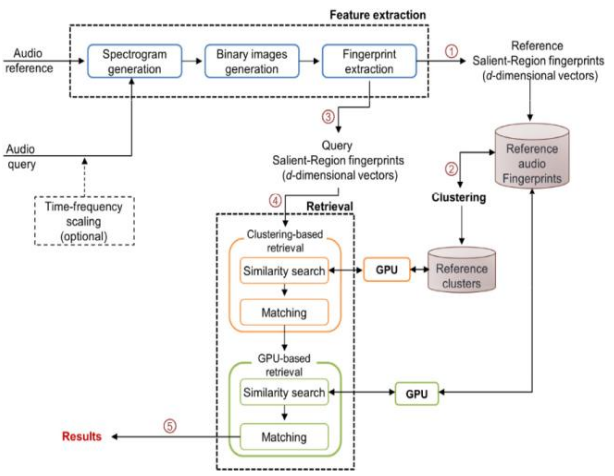

Fig. 1 System overview.

In a typical CBCD task, we search for a query audio in a database of reference audio files (containing copyrighted or original audio content) to see if the query is a copy of one of these reference audio files. For this task, we first extract robust audio fingerprints from these reference audio files. To extract these fingerprints, we first transform the audio signal into a spectrogram. We convert the resulting spectrogram into a set of 2-D binary images. Finally, we extract the top-d salient regions from each binary image.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 12, December 2016

Each Salient-Regions fingerprint is represented by a d-dimensional vector, which contains the positions of the selected salient regions only, and stored into a reference fingerprint database. Once all reference fingerprints have been generated, a k-means like clustering algorithm groups the reference fingerprints into different clusters. These clusters are then used to accelerate the fingerprint search. Query fingerprints are extracted in the same way with an optional time-frequency scaling step (resampling) that reduces the speed difference between the query and the reference. Fingerprint retrieval is a two-step search: clustering-based retrieval followed by a GPU-based retrieval. This two step retrieval reduces the search time by several orders of magnitude. In this two-step search algorithm, a similarity search is performed using either the reference clusters (in case of clustering-based retrieval) or the original references (in case of GPU-based retrieval) to find the closest query fingerprint for each reference fingerprint. This similarity search associates with each reference fingerprint a query frame number corresponding to the best matching query fingerprint. The similarity search is followed by a matching step that associates a matching score between each reference file and the query. This matching step produces the final results and determines if the query is a copy of a reference audio file.

A. Feature Extractions

The feature extraction step is shown in Fig. 2. The spectrogram matrix in this figure represents the intensity of the signal at any given time and frequency. Finally, each binary image is divided into small square tiles of size 11 × 11 resulting in 744 tiles2. The sum of all the 121 elements within each tile is computed to obtain a quantized image. Then, a few salient tiles that have the highest sums are selected. The fingerprint encodes the positions of these salient tiles and eliminates their sums. The example in Fig. 2 shows a binary image divided into 16 tiles (D = 16) and 6 salient tiles (d = 6) are selected.

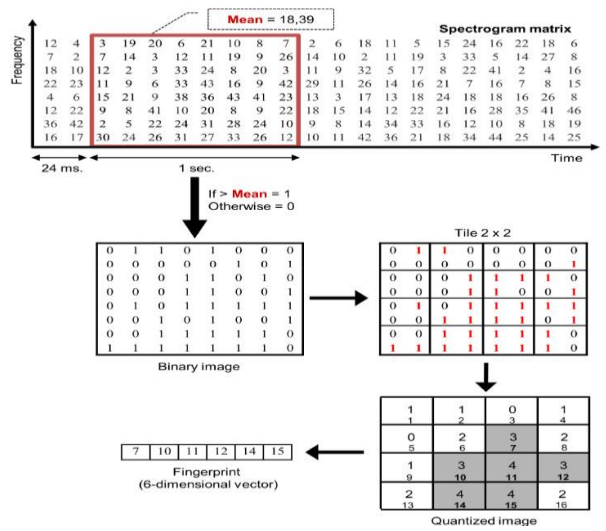

Fig. 2 Feature extraction with 16 tiles (D=16) and 6 salient tiles (d = 6) from quanized binary image derived from the spectrogram matrix.

The positions of these 6 salient tiles represent the compact fingerprint. Adding more information to the spectrogram here does not increase the complexity of our system, since this spectrogram will be converted into compact signatures. These parameters are not critical to our system, and changing their values to a certain degree will not affect the system performance. Each spectrogram matrix is composed of 333 consecutive spectrograms derived from 96 ms windows with a frame advance of 3 ms. Second, the large overlap (1-sec frame length with 24 ms frame advance) is chosen to overcome the lack of synchronization between the query and the reference. In fact, a large frame advance can prevent matching of query and reference frames when the start of the query is not synchronized with the start of the reference.

B. Retrival

Once all the reference and query fingerprints have been created, a search is performed to see if the query is a copy of an original audio in the reference database. Two principal steps for the retrieval process are:

Similarity search: Each reference frame is labeled with the frame number of its closest query fingerprint. The query consists of a number of frames, say n. For each query frame we compute its fingerprint as shown in Fig. 2 Similarly, each reference consists of a number of frames m, with a fingerprint extracted for each frame as shown in Fig. 2. For the fingerprint corresponding to each reference frame, we find the closest query fingerprint. The query frame number of this closest query fingerprint is then associated with that reference frame. In other words, in similarity search, we are associating a query frame number to each reference frame through this search. In this similarity search, the similarity measure is equal to the number of salient positions that coincide (i.e. intersection between reference and query fingerprint elements). Each reference frame is then labeled with the frame number of the closest query fingerprint. The total number of frame matches in the similarity search between a query (containing n frames) and a reference (containing m frames) is n × m (the total computing is proportional to n × m × d).

Matching: After the closest query frame has been found for each reference frame, the total number of reference frames that match the query frame-synchronously is computed: We move the query over the reference, and we count the number of reference frames that match exactly the query frame number for each alignment. This count represents the confidence in the match between the query and the reference.

VI. ALGORITHM

The similarity between two fingerprints is defined as the intersection between the elements of these two fingerprints. For example, if F1 = {1, 3, 4, 6, 8} and F2 ={1, 2, 4, 7, 9}, then similarity (F1, F2) = count {1, 4} =

2. Formally, for two fingerprints F1 and F2, the similarity between them is:

Algorithm 1 Hashing-based similarity computation
Input: Two vectors v1 and v2 of size d
Output: The similarity between v1 and v2

```
1: sim = 0
2: create a vector hash of size D
3: for i = 1 to D do
4: hash[i] = 0
5: end for
6: for i = 1 to d do
7: hash[v1[i]] = 1
8: end for
9: for i = 1 to d, do
10: sim = sim + hash[v2[i]]
11: end for
12: return sim
```

Algorithm 1a GPU kernel of hashing-based similarity search
Input: reference fingerprints, query fingerprints
Output: the closest query frame to each reference frame

```
1: create a vector hash of size D in shared memory
2: create a vector ref of size d in registers/local memory
3: for each thread do
4: load in ref one d-dimensional reference frame from global memory
5: max = 0
6: sim = 0
7: pos = –1
8: for each query frame n in global memory do
9: for i = 1 to D do
10: hash[i] = 0
11: end for
12: for i = 1 to d do
13: hash[queryframe[n][i]] = 1
14: end for
15: synchronize threads
16: for i = 1 to d do
17: sim = sim + hash[ref [i]]
18: end for
19: if (sim > max) then
20: max = sim
21: pos = n
22: end if
23: end for
24: end for
25: results [threadId] = pos
26: return results
```

Algorithm 2 Similarity computation without hashing
Input: Two vectors v1 and v2 of size d
Output: The similarity between v1 and v2

```
1: sim = 0
2: for i = 1 to d do
3: for j = 1 to d do
4: if (v1(i) = = v2( j)) then
5: sim = sim + 1
6: end if
7: end for
8: end for
9: return sim
```

VII. ALGORITHM WORK

The hashing-based algorithm (Algorithm 1) converts one d-dimensional fingerprint (vector v1) into a vector of D dimensions (d << D), and then looks for matching entries of the second d-dimensional fingerprint in this D-dimensional vector (See Fig. 2 for definition of d and D). This algorithm has a linear time complexity. However, memory is a critical commodity on a GPU, and a D-dimensional vector, for every thread, may drain GPU's resources (in our experiments D = 744 compared to a maximum d = 44). The first step in any GPU implementation is to transfer data from the host (CPU memory) to the device (GPU memories). Since the GPU has limited memory space, we process the reference set by portion. We transfer a reference portion to the GPU's global memory, and we perform the similarity search on this portion, then we process the next portion until all reference fingerprints have been processed. The number of query fingerprints is small and all the query fingerprints are transferred to the global memory. Once query and reference fingerprints are transferred into global memory, the GPU launches a kernel that performs the parallel portion of the application. In this kernel, hundreds of threads are executed in parallel, where each of them finds the frame number of the closest query fingerprint for one reference fingerprint. Algorithm 1a presents the proposed kernel, where the input are the reference and query fingerprints (loaded into global memory) and the output is a vector containing the frame number of the closest query fingerprint for each reference fingerprint. This similarity search uses the hashing-based algorithm. Each thread in this kernel starts by loading one d-dimensional reference fingerprint into registers or local memory (depending on dimension d: if there are not enough registers, then local memory is used to store one reference fingerprint for each thread). A D-dimensional vector (named hash) is created in shared memory to hash the elements of the query. It is also possible to switch the places for storing these variables by using the hash vector to hash the reference instead of the query.

VIII. MATHEMATICAL MODEL

Let system s can be defined as s= { A, As, B, Fd, C, R, Gm, O }
A = Audio refrence file.
A = { a1, a2, a3,............. an}
As = Audio sectogram generation.
As = { as1, as2, as3,............. asn}
B= Set of binary images generated from audio.
B = { b1, b2, b3,............. bn}
Fd= Set of fingerprint for each audio file.
Fd = { fd1, fd2, fd3,............. fdn}
C= Set of refrence cluster of audio fingerprint.

C = { c1, c2, c3,............. cn}
k= numbers of cluster.
R= Refrence query/search query audio file.
Gm= Set of GPU match result.
Gm = { gm1, gm2, gm3,............. gmn}
O= Set of output match audio files.
O = { o1, o2, o3,............. on}

IX. CONCLUSION

In this work, we propose speed up of a state-of-the-art audio fingerprinting system appropriate for detecting audio copies subjected to complicated transformations. To generate audio fingerprints, this system converts the audio signal into 2-D binary images derived from the spectrogram. Each fingerprint encodes the positions of salient regions selected from this binary image. The similarity between two fingerprints is defined as the intersection between their elements.

Because of the high dimensionality of the fingerprints and the large volume of data, searching over millions of fingerprints is computationally challenging. We investigate the use of a GPU to reduce the computing: we describe an efficient way of utilizing the GPU memories to search for similar fingerprints in parallel on a database of 60 million fingerprints. Experimental results demonstrate that the GPU implementation can accelerate the search by up to 150 times compared to the CPU implementation.

To reduce the run time even further, we propose a two step clustering based search. In the first step, we cluster the reference fingerprints into several thousand clusters, reducing the nearest-neighbor search time significantly. In the second step, we rescore the top N results obtained in the first step to produce more accurate copy detection.

Our system also outperformed MASK, WASF feature, Coherency vocabulary method, the energy difference fingerprints, and the robust NN-based system. Future work will be mainly devoted to exploring new strategies to make the Salient-Regions audio fingerprint invariant to time-frequency scale modifications. A possible way to achieve this goal is to encode the positions of the selected salient regions relative to each other, instead of their positions within the window.

A time-frequency modification applied to an audio signal leads to a proportional change in the time and frequency axes. This proposed strategy will ensure that the temporal and the spatial information will not be included in the fingerprint. As another part of our future work, we will investigate other promising clustering algorithms for robustness and for computing reduction. Finally, we intend to adapt the proposed audio fingerprint extraction technique to the video copy detection problem. In this case, salient regions will be selected from video images instead of audio spectrograms.

REFERENCES

1. Youtube. (Mar. 20, 2015). YouTube Statistics [Online]. Available: https:// www.youtube.com/yt/press/statistics.html.
2. W. Lezi et al., “Contented-based large scale web audio copy detection,” in Proc. IEEE Int. Conf. Multimedia Expo. (ICME’12), 2012, pp. 961–966.
3. A. Saracoglu et al., “Content based copy detection with coarse audio-visual fingerprints,” in Proc. 7th Int. Workshop Content-Based Multimedia Index. (CBMI’09), 2009, pp. 213–18.
4. X. Anguera, A. Garzon, and T. Adamek, “MASK: Robust local features for audio fingerprintingin Proc. 13th IEEE Int. Conf. Multimedia Expo. (ICME’12), Melbourne, VIC, Australia, Jul. 9–Jul. 13, 2012,pp. 455–460.
5. X. Zhang et al., “SIFT-based local spectrogram image descriptor: A novel feature for robust music identification,” EURASIP J. Audio Speech Music Process., vol. 2015, no. 1, pp. 1–15, 2015.
6. H. Jegou et al., “BABAZ: A large scale audio search system for video copy detection,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP ‘12), 2012, pp. 2369–72.
7. M. Malekesmaeili and R. K. Ward, “A local fingerprinting approach for audio copy detection,” Signal Process., vol. 98, pp. 308–321, 2014.
8. M. Ramona and G. Peeters, “AudioPrint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme,” in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), 2013, pp. 818–822.