VNU Jounal of science, Mathermatics - Physics 27 (2011) 181-188

An Efficient Music Identification System Based on PostgreSQL User-Defined Functions

Pham Cam Ngoc*, Nguyen Hai Chau

Faculty of Information Technology, VNU University of Engineering and Technology, 144 Xuan Thuy, Cau Giay, Hanoi, Vietnam

Received 28 September 2011, received in revised from 31 October 2011

Abstract. In this paper, we present a novel approach for music identification task aimed at proving the ability to identify a song by recorded song snippets. By combining Y. Ke's feature extracting method [1, 2] with PostgreSQL user-defined functions [3, 4, 5]], our system proves as an effective search strategy for the field. We construct training data sets in a noisy environment and compare the search speed and the search accuracy of the system with Y. Ke's system. Experiment results show that our system is more powerful with the accurate retrieval ability of 98% on a database of 600 songs and the search speed is 3.6 times faster than Y. Ke's system.

Keywords: Audio snippet, music identification, user-defined functions.

1. Introduction

The goal of music identification is to match short recorded audio snippets to an appropriate song [6, 7, 8, 9]. Because of the distortions of the snippets caused by low quality recording devices or natural ambient noises, simple matching is insufficient for this task.

Recently, numerous approaches, which utilize efficient audio feature calculating mechanisms, have been proposed to deal with this task. Haitsma and Kalker [10, 11] propose the technique based on overlay windows of audio signal to preserve the best energy properties called sub-fingerprints. Y. Ke et al [1, 2] present a method of combining machine learning technique in selecting features, which is also called rectangular features [12], to enhance their persistence. Baluja and Covell [13, 14, 15] improve the technique by using data stream processing method instead of machine learning method to allow querying in large databases. In this paper, we use the approach based on machine learning of Y. Ke when extracting time-frequency features of audio snippets to build our music identification system. Furthermore, we execute a preprocess on recorded audio snippets by increasing these signal amplitude and construct two new data training sets aimed at better search results. Based on PostgreSQL, we construct a song/fingerprint data base enabling effective index and user-defined functions, both of which help increasing average search speed of the system.

* Corresponding author. Email: phamcamngoc@gmail.com

2. The music identification system based on postgresql user-defined functions

2.1 The structure of the system

In Y. Ke's research, the music identification system in Fig. 1 consists of two parts: the client and the server. The client's tasks are to receive audio signals, record these signals, perform useful calculation to extract audio features (energy features that are selected by machine learning technique) and then send those features to the server. The server will search in meta-data/features database an appropriate song that best matchs with those features and after that return the result for the client.

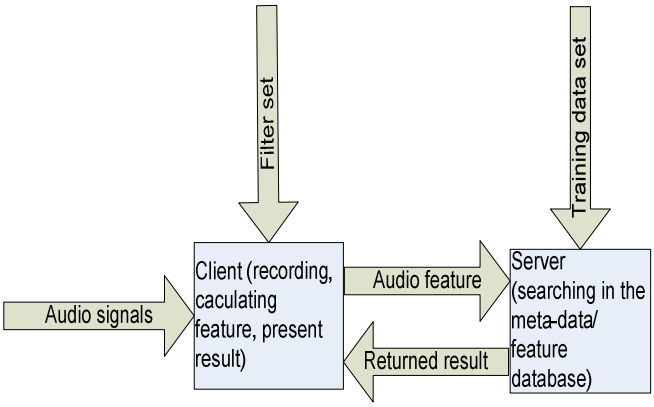

Figure 1. The structure of Y. Ke's system.

Our system (Fig. 2) is constructed similarly to Yan Ke's system; however, there are two essentially points distinguishing our system from his. Firstly, after being sent to the client, audio signals are preprocessed by increasing the amplitude signals, and then client will calculate audio features of those. Secondly, when receiving these audio features, the server would utilize user-defined functions and a newly built training data set to find out the appropriate song that matches with these features in a large database created in PostgreSQL.

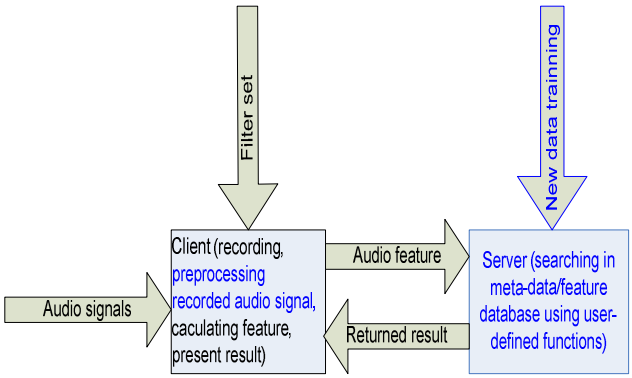

Figure 2. The structure of our system.

2.2 Constructing test data set and training data set

When carrying out building these data sets, we use Unix/Linux OS environment to compile and run applications. We have useful functions, which help automatically play song, record the songs and after that split the songs into song snippets with the length of 30 seconds. Those functions are created using open source libraries such as fftw3 [16], ffmpeg [17] and mpg123 [18]. Throughout the process, we attempt to create noises to diminish the quality of recorded songs in order to increase the difficulty of test data set and the effectiveness of the training data set. The noises include natural noises, for example the sound of electronic fan, the sound of TV, the sound of engine of cars or motorbikes. In order to recognize precisely the recorded song snippets in the noisy environment, we propose two solutions. Firstly, the recorded song snippets will be increased amplitude by increasing volume and thus we can reduce effects of noises. Indeed, recorded song snippets due to particular reasons will be distorted, which result in differences between features of recorded song snippets and those of correnspponding original song snippets. When incresing volume of recorded song snippets, we also restore the signal spectrum of recorded song snippets and thus, effects caused by noises, quality of recorders and others will be reduced. Secondly, in the research of Y. Ke [1, 2], the training data set is built relatively simple when it only contains some pairs of recorded and corresponding original audio snippets. Therefore, when constructing training data sets, we attempt to select an appropriate number of song snippets from various genres and use the above functions to build training data sets. Thanks to a considerable number of songs and varied genres of songs used for training, we can create sufficient and good training data sets that bring us better identification results. The training data sets in our system are built based on EM algorithm like in [1, 2]. For instance, each original song and corresponding recorded song are split exactly into 30 seconds intervals (audio snippets) and then aligned together to estimate 66 Bernoulli parameters [1, 2], which are used by the system to compute the likelihood between a query song snippet and original song snippets in the data base. We note that the large number of songs used to train will help estimate those parameters more precisely, but not reduce the search speed of the system because the estimating parameters and searching of the system are two independent works.

In our experimental setup, the test data sets are created by selecting randomly a specific number of 30 seconds recorded song snippets, which have been recorded and split, according to different music genres.

2.3 Designing meta-data/feature database in PostgreSQL user-defined functions

The task of searching in a meta-data/feature is not simple. Assuming that we have a medium database with 10.000 songs and the average lengths of songs is 5 minutes; we will have approximately 250 million features, which is a very waste of time even with the best search algorithms. In the research of Haitsma [10, 11], the fingerprint database is organized as a lookup table-LUT with all possible 32 bit sub-fingerprints as an entry. Each entry will point to a list of positions of sub-fingerprint, which have the same value with the entry. When receiving a fingerprint block, the system will use the lookup table to search in the fingerprint database songs, which contain at least one sub-

fingerprint that also belong to the fingerprint block and then compare the similarity between the candidate songs with the fingerprint block by calculating the Hamming distances between these. In practice system, because of the limited ability of the memory, a lookup table including 232 entries is impractical and thus, a hash table is used instead of a lookup table.

In our system, we organize the database as follows. The database consists of two relations: the song relation contains attributes: song identification, song name, song directory, features (is an array of 32 bit integer numbers), the length of the array and the key relation contains attributes: feature identification, song identification that the feature belong to, the position of the feature in the respective song and the value of the feature (a 32 bit integer number). After designing the database, we construct functions to allow automatically receiving a list of songs, calculate audio features and insert the features into the database. In order to receive high search speed we utilize index technique of PostgreSQL in feature field of the key relation, which proves to be very effective in practice. PostgreSQL supports the following index methods: B-tree, hash, GiST and GIN and users can also define their own index methods. In our system, because the feature field is a 32 bit integer number, we use the B-tree index method in the field. When the query features are sent to the server, the server first will search in the database songs that their features contain at least one feature which also belong to the query features. The work is performed by comparing 32 bit integer numbers of the query features to 32 bit integer numbers in the database and therefore, the B-tree index method is the best choice in the situation.

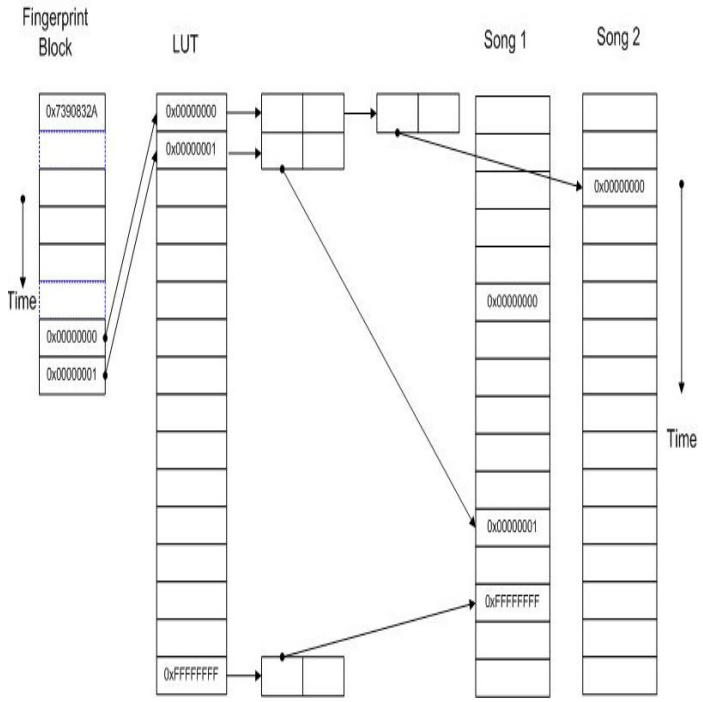

Figure 3. The fingerprint database is organized as a lookup table [6].

Based on user-defined function that PostgreSQL provides, we construct two functions that meet the requirements of search speed.

The *Find_near_neighbour* function will receive a query as a set of features, the function will look through these features, searches in database, and then returns songs called candidate songs that contain one of these features. The function also tells the system exactly what the position of those features in the candidate songs are.

The *Compute_simalarity* function uses the RANSAC algorithm [12] to calculate the similarity between the query audio features and each of candidate songs. The function will return the song that is most likely to be the original song of the query audio features.

Thanks to the index method, these functions can be performed effectively in a large database.

3. Accuracy and cumputation

We do experimental work with three training data sets: Y. Ke's training data set called YK and two training data sets that we created called HL1 and HL2 on two test data sets T1 and T2. For instance, T1 contains approximately 600 song snippet recorded by low quality recording devices and in a natural noise environment while T2 contains all song snippets in T1 after increasing amplitude. When testing with T1, we recognized that the true positive rate (the number of positive results in comparison with the number of recorded song snippets used to experiment) gained by YK, T1 and T2 are 86.7%, 89.5% and 90% respectively. Now, the accuracy is much better with preprocessed data set T2, as the results gained are 94.3%, 98% and 98%, respectively. Therefore, both HL1 and HL2 bring us better experimental result than YK when being tested with T1 and T2. There are two reasons for our better accuracy. First, in our system, we have a preprocessing step by increasing amplitude of recorded song snippets before sending to the server to compute features and recognize the original song of these snippets. Therefore, computed features are more reliable than those which are not preprocessed. Second, our training data set is built in real environment with natural noises and each song snippet pairs in training data set are aligned exactly using shell functions. This will result in a better training data set compared to Y. Ke's training data set and thus bring us better precision.

Moreover, we compare the performance of Y. Ke's system and our system. We create a test data set containing 100 recorded song snippets with the length of 30 seconds and then use these as queries in order to measure query timing of our system and Y. Ke's system. On average, our system query timing is 2.38 seconds, which is 3.6 times faster than Y. Ke's system.

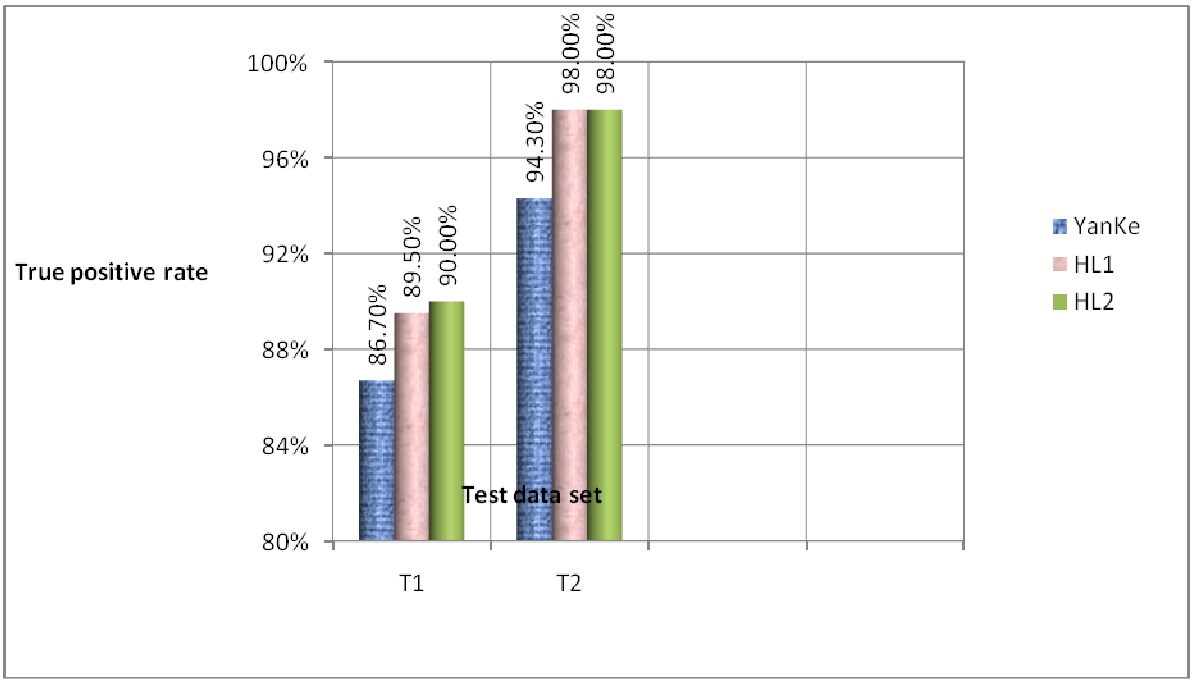

Figure 4. The identification accuracy gained by YK, HL1and HL2.

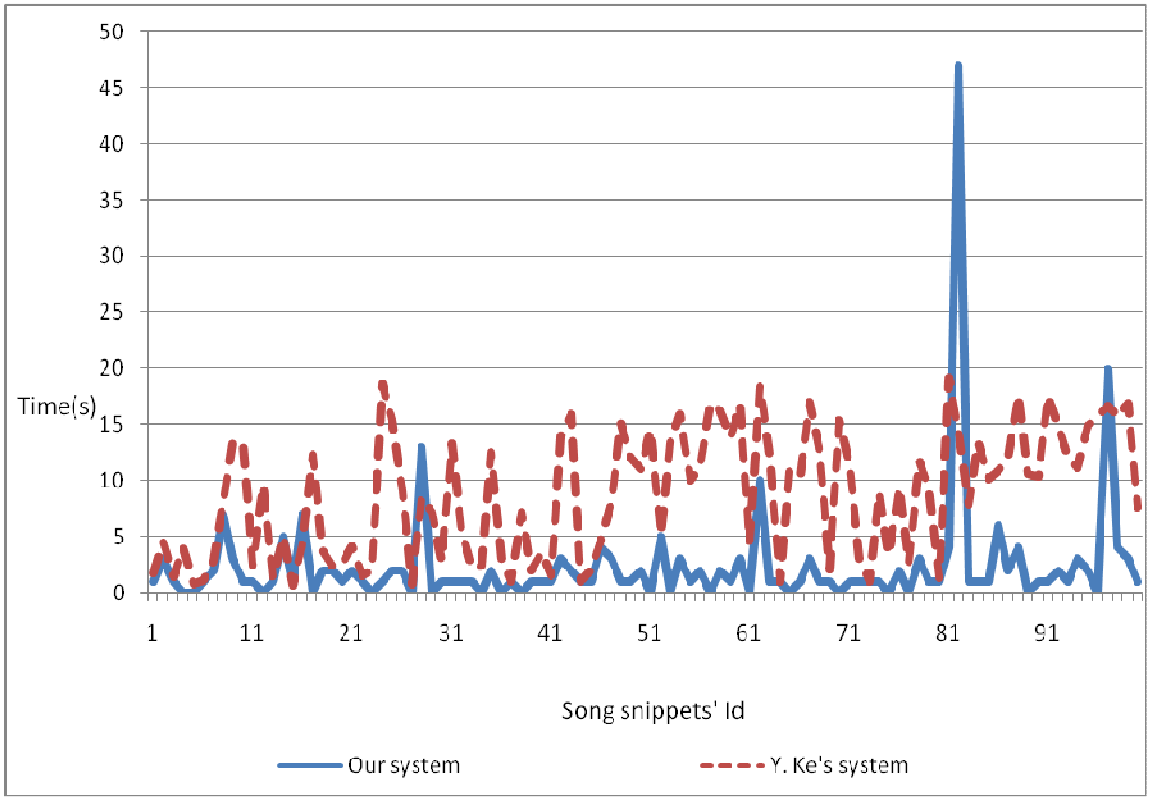

Figure 5. Comparing the search speed of two systems.

4. Conclusions

This paper has presented that user-defined functions supported by PostgreSQL can be applied to construct an efficient music identification system. Our research contributions can be listed as follows. First, we describe how to build a good training data set to enhance query results of the system. Second, we propose the way to design and organize a meta-data/fingerprint database beside an indexing method. Third, we suggest using user-defined functions of PostgreSQL to erect an impact strategy. Finally, we demonstrate by experiment results that our system is more effective than Y. Ke's system in terms of the accuracy of search result and the search speed.

In the near future, we would continue to improve our system by constructing a larger song data base which combines better indexing scheme. We also examine other feature extracting algorithms to increase the distinguishing feature of each sub-fingerprint. Experiment results have motivated our belief that PostgreSQL is a good way enabling us to build a music identification application that can be effectively applied in practice.

Acknowledgments. The work is supported by the research projects No. QC.08.01 and QG.09.27 granted by Vietnam National University, Hanoi.

References

[1] Y. Ke, D. Hoiem, R. Sukthankar, Computer Vision for Music Identification, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005.

[2] Y.Ke et al., Computer vision for music identification: server code, <http://www.cs.cmu.edu/~yke/musicretrieval/musicretr-1.0.tar.gz, 2005>.

[3] Nei Matthew and Richard Stones, Beginning Databases with PostgreSQL: From Novice to Professional, Second Edition, 2005.

[4] Korry Douglas, Susan Douglas, The comprehensive guide to building, programming, and administering PostgreSQL databases, Second Edition, 2005.

[5] http://www.postgresql.org/docs/8.0/static

[6] http://www.shazam.com

[7] http://www.relatable.com

[8] http://www.musipedia.org

[9] http://www.napster.com

[10] J. Haitsma, T. Kalker, A Highly Robust Audio Fingerprinting System, *Proceedings of the International Conference for Music Information Retrieval*, 2002.

[11] J. Haitsma, T. Kalker, J. Oostveen, Robust Audio Hashing for Content Identification, *Content Based Multimedia Indexing* 2001, Brescia, Italy, 2001.

[12] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography In *Communications of the ACM*, 24(6), 1981.

[13] S. Baluja, M. Covell, Content fingerprinting using wavelets, Proceedings of the 3rd European Conference on Visual Media Production (CVMP), 2006.

[14] S. Baluja and M. Covell, Audio Fingerprinting: Combining Computer Vision & Data Stream Processing, *Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[15] M. Covell, S. Baluja, Known-Audio Detection Using Waveprint: Spectrogram Fingerprinting By Wavelet Hashing, *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2007.

[16] http://www.fftw.org

[17] http://www.ffmpeg.org

[18] http://www.mpg123.de