# MIREX 2019 Audio Fingerprint System

Xiao Lu
QQ Music BU, Tencent Music Entertainment
Shenzhen, China
shawlu@tencent.com

Lester Kong
QQ Music BU, Tencent Music Entertainment
Shenzhen, China
lesterkong@tencent.com

Ethan Zhao
QQ Music BU, Tencent Music Entertainment
Shenzhen, China
ethanzhao@tencent.com

Simon Lui
QQ Music BU, Tencent Music Entertainment
Shenzhen, China
nomislui@tencent.com

***Abstract*—This short paper illustrates our audio fingerprinting system submitted to MIREX 2019 for the audio fingerprinting task. In the proposed system, local maxima, also called landmarks, of the spectrogram are detected, paired and hashed. To build the database for fast matching, a hash code is built as the index of its corresponding file id and time. Our submissions attempt to achieve high recognition rate while saving memory occupation and approaching fast retrieving.**

## I. Introduction

The audio fingerprinting problem can be described as finding the correct audio that an audio query is corresponded to. It has been applied in a variety of applications and scenarios, such as music recognition services, audio copyright detection and broadcast monitoring.

The query can usually be noisy, involving various acoustic noise and distortion. The noisy versions of the GTZAN dataset [1] have been used as the MIREX challenge for years. Therefore, a reliable audio fingerprinting system should be robust to these variations of the source sound. Additionally, there are two challenges when constructing the database and the retrieving system – a) an audio fingerprint should be compact but representative so that it saves storage space, memory consumption and data transmission time; b) the matching strategy and its implementation should be efficient.

To address the issues stated above, our system follows the initial work of [2], where the local maxima of the spectrogram as the feature are extracted, and then hashing and reverse indexing are employed to minimize data size and increase the efficiency of searching and matching. A preliminary experiment on our system has been conducted and the results are competitive.

The remainder of this paper discusses the general system architecture and methods that we used, followed by a brief description of the submitted package.

## II. Fingerprint Database Building

The audio fingerprinting system consists of two main parts, the building part and the matching part. As illustrated in *Figure 1*, each part involves a landmark extraction stage and a hashing approach. A reference audio is first downmixed to mono, downsampled to 8k Hz and analyzed using STFT with a window size of 1024 and 50% overlapping. By finding the local maxima (peak values) of a spectrogram within designated regions and pairing them into $\{t1, f1, t2, f2\}$ pairs, we use a hash function to encode $\{f1, f2-f1, t2-t1\}$ to a hashcode and keep the time information $t$ for building the database. To build the database, a reverse index is constructed where a hashcode is indexed to its corresponding track ID and time.

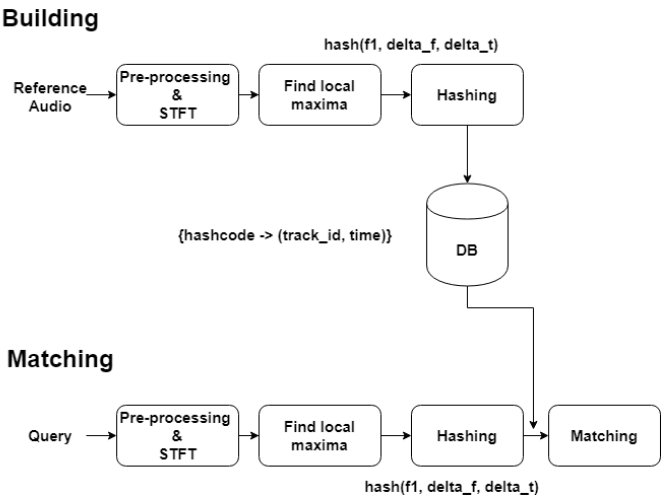


***Figure 1.*** *General audio fingerprinting system.*

## III. Fingerprint Matching

During the matching process, with the same peak-picking and hashing techniques, the extracted fingerprints are used to look for matched track IDs in the reference database.

The system counts the number of matched hashes and checks if the occurred time are aligned to time in the matched reference audio and it declares the result after sorting and thresholding. For better results, we also attempt to recognize different portions of the query audio and make a final decision by a voting algorithm.

To speed up the process, some information in the hash index might be discarded as they are less important to the recognition result but time-consuming. For instance, we can set a limit of how many hashes at the same time occurred to load to the hash table. In addition, parallel processing is used to improve the performance.

## IV. System Implementation

The system is implemented in C++, designed with scalability to insert or remove new fingerprints from the hash

table online or offline. It also allows convenient deployment for large song databases and large number of servers.

The submission includes a simplified version of the retrieval services that are already compiled to binaries. The building part and the matching part are written as python scripts used for running different binary programs. More details of the submission are documented in the README file.

## REFERENCES.


[1] G. Tzanetakis, "Gtzan genre collection," web resource, 2001. [Online]. Available: http://marsyas.info/ download/data sets

[2] A. Wang: "An industrial strength audio search algorithm," Proceedings of the International Symposium on Music Information Retrieval, 2003.