Multimedia Security
Audio Fingerprinting Practical Work

2024-2025

Multimedia fingerprinting (also known as perceptual hashing) are a class of techniques used to produce short summaries of multimedia content. These summaries can then be used to search for these pieces of content in databases in an efficient way. Typical uses of perceptual hashing are *reverse search*, by which a piece of multimedia itself (instead of the file metadata, such as the name of the piece or its author's name) can be used to query a database; and *copy detection* to identify unauthorized copies of copyright-protected work or their distribution. An example of the first use would be the Shazam service[1], which can be used to identify a song in a database from a short, often noisy, recording from a smartphone's microphone. An example of the second use would be Toutube's ContentID[2]. More details on media fingerprinting can be found in the slides for Unit 5 *T5 - Copy Detection*.

1 Task description

This practical work consists on implementing an audio fingerprinting system for reverse search database search, similar to the *Shazam* service, that is capable of identifying songs from short noisy samples. In the **Practical Exercices** section in the course moodle page, you will find a music library named *Music Library part 1* and *Music Library part 2*, which contain 40 songs in total. The provided songs are wave files (.wav) with 2 audio channels, sample rate of 44100 samples per second and 16 bits per sample, making them $16 \times 2 \times 44100 \times t$ in size each, being t the song duration in seconds. Also, in the same moodle page section, you can find a selection of test samples, extracted from the songs in the Music Library, that you can use for testing. All samples have a random duration of 8 to 15 seconds and are divided in 4 categories:

- **Clean samples** are extracted direclty from the songs in the Music Library and have no aditional processing.
- **Filtered samples** have been filtered with a bandpass filter. They should emulate quality of samples taken with a (low quality) microphone or

[1] Shazam, `https://www.shazam.com/`
[2] ContentID, `https://support.google.com/youtube/answer/2797370?hl=en`

through a landline phone.

- **Noisy samples** have been mixed with the noise samples also available in the moodle course page. The volume of the song with respect to the background noise is random for every sample (some of the samples are barely audible).
- **Noisy filtered samples** combine the two previous processing actions, they should emulate noisy samples recorded with a bad quality microphone.

Your work is to **implement two utilities**, one that **builds a fingerprint database** from the songs directory and a second that **identifyies a song from the database using one of the samples**. You can implement the system in any programming language of your choice. The application design and architecture are up to you. **Note:** The paper describing Shazam is in the moodle course page in the complementary reading section of Unit 5. Use that paper as your guide. However, the commands to build the fingerprints database and to identify a sample must be of the following form:

- Building the fingerprint database: `builddb -i songs-folder -o database-file`
- Identifying a sample: `identify -d database-file -i sample`

The first of the utilities does not need to return anything in particular (except for building the database). The second one should return the **song name** in the console. This does not mean that the sofware should be compiled. If you develop the programs in an interpeted language such as python or deliver a Java .jar file indicate so in your documentation. This means that the following examples are also valid:

- `python builddb.py -i songs-folder -o database-file`
- `java -jar builddb.jar -i songs-folder -o database-file`
- `builddb.sh -i songs-folder -o database-file`

But be completeley clear in how to execute the commands in the provided documentation.

2 Evaluation

You have to deliver three items:

- The `builddb` utility.
- The `identify` utility.
- Documentation.

The documentation file must include at least the following items:

- Design of the utilities. This includes:
  - Programming language of choice.
  - Any and all choices you make for parameters. For example, window sizes of the Fourier transforms when building the songs' spectrograms, density of points, minimum amplitude of points, etc. Details on these parameters are in the Shazam paper.
  - Database schemas.
  - Any other design choices of relevance.
- Detection metrics: precision, recall, etc.
- User manual, including System requirements and dependencies.

The evaluation of the practical work will be based on the quality of the delivered items, both software and documentation. Please make your code readable and include any necessary comments, make sure the documentation is clear and complete.