Evaluating the Suitability of SoundFingerprinting for Music Identification in Movies

Tim Huisman
TU Delft

Abstract

Audio fingerprinting has shown to be an effective approach to music identification, having properties robust to noise and signal degradations. A field in which audio fingerprinting has not been evaluated yet is music identification in movies. In movies, music is often accompanied with background noise, sound effects and dialogue, and further processed using mixing and mastering techniques.

This paper evaluates the suitability of the audio fingerprinting framework 'SoundFingerprinting' for the identification of music in movies. The framework is evaluated according to a benchmark established for this field. The framework was tested on actual movie data, noise-layered soundtracks, pitch-shifted soundtracks and tempo-changed soundtracks. The framework was unable to identify the music in actual movie data, thus directing the research to identify problematic areas specific to SoundFingerprinting. In identifying noise-layered soundtracks, the framework showed varying performance dependent on the dominant frequencies present in the noise sample. Furthermore, the framework showed to be robust to tempo-changes, whereas the framework was unable to identify pitch-shifted soundtracks. Based on this performance evaluation, SoundFingerprinting is ill-equipped for the task of music identification in movies.

1 Introduction

For humans, establishing the equality of two songs is a trivial task. While listening to the songs, we can understand lyrics, tone, emotion and structure. For computers, however, this task is much more complex. A computer has no prior knowledge of what music is and can only act based on the ones and zeros that ultimately form the signal. Comparing the complete signal of one song to another is computationally expensive, making identification in this way unfeasible for large music databases. In the early 2000s, a new technique for establishing equality of audio objects emerged, called audio fingerprinting. Instead of comparing the typically large objects directly, audio fingerprinting algorithms reduce these audio objects to many small fingerprints such that these compact representations can be matched efficiently. The concept of an audio fingerprint is analogous to human fingerprints, e.g. a small signature of a human which can uniquely identify a human. Although the fingerprint does not give any information on what the human looked like or what (in the case of audio fingerprinting) the original signal looked like, it can be used to identify the object. Throughout the last two decades, many successful implementations of this technique have surfaced for a variety of applications, such as radio-stream monitoring, integrity verification, and content-based audio retrieval [1]. Research in this field has come to somewhat of a halt, yet new challenges emerge, such as music identification in movies.

In movies, contrary to, for example, radio-stream monitoring, music is often not the main point of focus; it supports in conveying the intended message together with the video. Songs may be played in combination with cinematic shots, dialogue or action scenes. In final mixing for movies, one has to combine several audio sources (dialogue, music, background noise, effects, foley) for the final theatre-mix. Using various mastering techniques such as equalisation, filtering, compression/limiting, and levelling, final mixing engineers are able to balance the audio and highlight the essential parts of the final mix [2].

Audio fingerprinting appears to be a promising approach to this challenge, as it allows for fast querying in large music databases, and there are implementations that have shown to be (to a certain extent) robust to noise and signal degradations [3] [4]. Yet, when audio fingerprinting implementations are applied to the field of music identification in movies, performance drops significantly. Given the complexity of the final theatre-mix and the variety of techniques employed by different audio fingerprinting implementations, there are many factors that could cause this sub-optimal performance. To compare the performance of different implementations in music identification in movies, a benchmark [5](i.e. a global evaluation method) has been established in cooperation with four other researchers. This benchmark decomposes the complex problem of music identification in movies into separate categories to analyse problematic areas for a given implementation. It is highly recommended to read the paper on the benchmark beforehand, as it is referenced throughout the rest of this paper.

This paper evaluates the suitability of an open-source audio fingerprinting implementation, SoundFingerprinting, for music identification in movies. This research answers the following research question is answered:

- How does SoundFingerprinting perform on music identification in movies?

This research question is split up into two sub-questions:

- How does SoundFingerprinting perform according to a benchmark established for evaluating audio fingerprinting frameworks for music identification in movies?
- What configurable parameters influence the performance of the framework for music identification in movies?

This paper continues in the following way. In section 2, the works related to this research are discussed. Next, in section 3, a description of the implementation of SoundFingerprinting is given, along with a description of its configurable parameters and possible strengths and weaknesses. After that, the evaluation methodology is described in section 4. This section contains a summary of the benchmark and a description of the evaluation setup. The results of this evaluation are shown in section 5, followed by a discussion of the limitations of the evaluation in section 6. The paper continues with a section on the reproducibility of this research in section 7, after which it is concluded by section 8, a section containing the conclusion and future work directions.

2 Related works

There is currently no known research on the application of audio fingerprinting for music identification in movies. However, there is research showing promising results in similarly challenging environments.

In one of the papers on which the implementation of SoundFingerprinting is based, Haitsma and Kalker [3] describe an algorithm robust to various kind of signal degradations. It was tested on four short audio excerpts (stereo, 44.1kHz, and 16-bit samples), which underwent signal degradations, including band-pass filtering, amplitude compression, equalisation and tempo-modification. For all above-mentioned degradations, error rates in the order of 15% were achieved. This shows that there are audio fingerprinting implementations robust to the various mixing and mastering techniques applied to movie audio. However, this research did not examine any types of interfering noise.

The most well known audio fingerprinting implementation is Shazam[4]. Shazam is a commercially deployed flexible audio search engine, resistant to noise and distortion. It is able to correctly identify tracks under the presence of voices, traffic, dropout, and even other music. Their algorithm was tested against synthesised data generated by layering a noise sample of a noisy pub over excerpts of 15, 10, and 5 seconds of length taken from tracks in the database. The noise sample was subsequently layered onto the track at specific signal-to-noise ratios. Their algorithm achieved a 50% identification rate for 15, 10, and 5-second samples at approximately -9, -6, and -3 dB SNR, respectively. For SNRs above 0, the identification rate was higher than 80%. The audio sampling and processing was carried out using 8KHz, mono, 16-bit samples. Furthermore, for a database of about 20 thousand tracks implemented on a PC, the average search time was on the order of 5 to 500 milliseconds. While the Shazam algorithm showed high identification rates for short, noise-layered samples, it has only been evaluated on a single sample of a noisy pub. In movies, there is a vast range of types of noise that could pose problems for identification. Each type of noise has a different waveform, indicating the possibility that some types of noise could be more problematic for identification than others.

3 SoundFingerprinting

SoundFingerprinting is an open-source C# audio fingerprinting implementation based on a paper written by Baluja et al. [6]. This paper introduces a novel method for audio identification, which uses computer-vision techniques and large-scale-data-stream processing algorithms to create compact fingerprints of audio data that can be efficiently matched. This section contains a summary of the algorithm, its possible strengths and weaknesses, and its configurable parameters. A complete description of the workings of SoundFingerprinting can be found on CodeProject [7].

In the first step of the algorithm, the input signal is preprocessed. The sampling rate most commonly used in digital audio is 44.1 kHz, which is more than twice the frequency that humans are able to hear (20 Hz - 20Khz). This is necessary, as the Nyquist-Shannon sampling theorem [8] states that perfect reconstruction of a signal is only possible when the sample rate is greater than twice the maximum frequency of the signal being sampled. However, in audio fingerprinting, we are not interested in perfectly recreating the signal but in capturing what is most characterizing. Instead, the input signal is downsampled to 5512Hz, reducing the size of the input significantly while focusing on the frequencies most relevant to the human auditory system [3]. Additionally, before any other processing steps, the audio is converted to mono and Pulse-code modulation (PCM) format.

To further process the audio input, a method is followed that has been found to work well in previous audio fingerprinting studies [3]. The signal is split up into 371 ms long frames, taken every 11.6 ms. The Fast Fourier Transform (FFT) is applied for each frame, transforming the signal from the time domain into the frequency domain. This is the bottleneck of the entire algorithm; the FFT runs in $\mathcal{O}(n \log(n))$. The result, a spectrogram, is then cut such that the 318 Hz-2000 Hz domain is obtained. This frequency range is a configurable parameter; we elaborate more on this in subsection 3.2. To further minimize the output dimensionality, the specified range is encoded in 32 logarithmically spaced bins. The resulting spectrograms are then combined to form a frequency spectrogram of the entire audio input. From this spectrogram, 'spectral images' are extracted, each denoting the distribution of frequencies of 1.48 seconds of the audio input.

These spectral images are then further reduced by applying Standard Haar-wavelet decomposition, a technique that has shown great results in image retrieval [9]. For each of

the spectral images, a wavelet signature is computed. This wavelet signature is a set of wavelets in which each wavelet has a value corresponding to the energy observed in a particular frequency band of the spectral image. Out of this complete set of wavelets, only the ones that most characterize the song, i.e. have the highest magnitude, are kept. By selecting only the wavelets with the highest magnitude, small changes in the sound (i.e. small bit of noise, echo, other degradations) will not affect the wavelet-image. In SoundFingerprinting, the top 200 wavelets are kept, following the value suggested in [6].

The previous stage of processing produces intermediate fingerprints that are 8192 bits long. This length is further reduced by Min-Hashing, a "forgiving hashing" method that negates small differences. The method is based on the algorithms described in [10] and [11]. More information on these methods can be found in the respective papers, as discussing this is beyond the scope of this paper. Finally, using Locality Sensitive Hashing, the Min-Hash signature for each fingerprint is spread over 25 hash tables, which are used for lookup. When a query sub-fingerprint has more than v (a configurable parameter) matches with the Min-Hash signatures of a particular sub-fingerprint in the hash tables, the two sub-fingerprints are compared. Their overall similarity then results in the confidence of the match.

When the algorithm is presented with unknown audio input, the input undergoes the complete fingerprinting process such that its fingerprints can be matched to those stored in the database.

3.1 Possible strengths and weaknesses

An important consequence of the slicing method described in [3] is that the spectrogram of the entire input varies slowly in time. The features of the audio fragment represented in each fingerprint overlap with the features of the previous and the next fragment, providing matching robustness to position uncertainty in time. Next to that, retaining only the most characterizing wavelets causes the framework to ignore small changes in the audio, resulting in robustness to noise and degradations. However, when the interfering noise is louder than the song, the most characterizing wavelets might be populated with dominant frequencies present in the noise sample.

Furthermore, the developer of SoundFingerprinting has specified the following possible causes for matching failures on his blog [12]: aliasing, clipping, tempo changes and specific frequency ranges. Only the last two are described in this section, as aliasing typically only occurs when recording audio with low-quality devices and clipping is not specifically evaluated in this research.

The Tempo of a piece of music determines the speed at which it is played and is measured in beats per minute (BPM). Tempo changes are often used for remixing songs to adhere to a particular style of music. The developer states: "tempo change shortens the distance between frequency peaks. The initial song will not be recognized when querying with a remixed version". This, however, appears to be a bold statement, as the overlap between fingerprints might be able to compensate for this change in distance between frequency peaks.

Finally, as mentioned in the framework description, SoundFingerprinting uses specific frequency ranges for creating audio fingerprints. When songs that are stored in the database contain characterizing frequencies above or below this frequency range, the framework will not capture them in the fingerprint, which can, in turn, decrease the chance of recognition. When dealing with noise, however, this specific frequency range can be both a strength and a weakness of the framework. Noise with dominant frequencies outside of this range will have little impact, whereas noise that falls within this range will have a larger impact.

3.2 Configurable parameters

SoundFingerprinting has three built-in configurations for fingerprinting and querying: LowLatency, Default and HighPrecision. The first configuration, LowLatency, is intended for real-time radio-stream monitoring and thus will not be evaluated for music identification in movies. The other two, however, should be able to handle audio accompanied with noise to a certain degree. The developer recommends the usage of the HighPrecision configuration for queries containing ambient noise, thus making HighPrecision the most appealing configuration for this use case.

The main difference between the two configurations is the frequency range they consider. For the Default configuration, the specified frequency range is 318 Hz-2000 Hz, whereas the HighPrecision configuration only considers the range 1200 Hz-2500 Hz. To examine the effect of frequency range filtering, both configurations of the framework will be evaluated. Another difference is the minimum amount of matches in the hash tables needed for a stored sub-fingerprint to be compared to a query sub-fingerprint. For Default, this is 4, for HighPrecision, this is 3. Due to the time limitations imposed by this project and the time it takes to run a complete evaluation, the two parameters are not evaluated separately. Instead, the two configurations are run 'as a whole'.

The final thing to consider is the match confidence threshold. For each query, the framework returns a list of possible matches together with a confidence value ranging between 0 and 1, denoting the probability of the match being correct. If no confidence threshold is specified, the framework will always try to return a match, even if its confidence is only slightly higher than 0. The developer of SoundFingerprinting has indicated that any potential match with a confidence above 0.15 is most probably a true positive. However, as music identification in movies poses a more complex challenge than for example radio-stream monitoring, it might be the case that the confidence the framework has in exact matches will be lower. To examine the suitability of this threshold for this field, the framework will be evaluated for various confidence thresholds.

4 Methodology

For the evaluation of the framework, a data set of 49 movies, including the original soundtrack, has been made available by Muziekweb[1], a Dutch music library. Next to that, 500 addi-

[1]Muziekweb - The music library of the Netherlands https://www.muziekweb.nl/en/

tional random songs were supplied. The supplier has aimed to diversify this data set to the best of their extent. To get a first indication of the framework's performance on actual movie clips, six movies from different genres were manually split up into query clips. In doing so, we aimed to capture segments of the movie in which some kind of noise accompanies the original soundtrack. The results of this tentative evaluation were far from ideal: SoundFingerprinting was only able to identify 5% of the clips, consisting only of cases where the music was accompanied by very little noise. This called for a different, more controlled approach: synthesising data in a modular fashion in order to isolate problematic areas. This section contains a short description of the benchmark and the method to answer the research questions. A complete description of the benchmark can be found in the paper describing the benchmark [5].

4.1 Benchmark

4.1.1 Evaluation data synthesis

Based on the frequently recurring noise categories of the manually labelled movies, a selection of movie noise categories has been put together. Given the limited time, spatial and computational availability for this project, this selection was limited to 15 categories. The selection of noise categories can be found in Table 2. For each noise category, three different audio files of isolated noise were obtained from Freesound[2] and layered on top of the original soundtrack at multiple signal-to-noise ratios (SNRs). The selection of SNRs was limited to -6, 0 and 6 dB, indicating cases where the noise is twice as loud as the soundtrack, equally loud and half as loud, respectively. Next to categories of noise, the evaluation data set also contains pitch-shifted and tempo-changed versions of tracks. The data synthesis resulted in a total of 13,320 noise-layered soundtracks and 784 structurally altered soundtracks.

4.1.2 Criteria

The criteria by which the noise categories will be evaluated are robustness, reliability, and search speed.

Robustness, the framework's ability to identify tracks even when the signal is degraded, is measured by the metric commonly used in information retrieval: recall.

Reliability, the extent to which an output match can be trusted to be a correct match, is measured by precision, also commonly used in information retrieval.

Search speed is important to evaluate the scalability of the framework. In audio fingerprinting, there is a trade-off between fingerprint size and the search speed of the framework. A larger fingerprint size possibly improves the identification rate but has the drawback of a lower search speed. In a real-life use case, fingerprint databases often contain the fingerprints of tens of thousands of tracks. Search speed and scalability will therefore be measured by average query time with respects to the amount of tracks stored in the database.

[2]Freesound - a collaborative database of Creative Commons Licensed sounds. https://freesound.org/

4.2 Evaluation setup

4.2.1 Database population

Before any querying can be done, the database needs to be populated with fingerprints of tracks. As mentioned in subsubsection 4.1.1, only two random songs per movie (98 in total) were used to generate test data. To most accurately reflect a real-life use case, the complete soundtrack of every movie in the data set together with 500 additional random songs were used to populate the database with, resulting in a total of 1407 stored tracks.

4.2.2 Experiments

Once the database is populated with fingerprints, the framework is ready to be evaluated according to the metrics specified in the benchmark. Each category is evaluated separately to evaluate the performance of the framework in a modular fashion. Throughout the evaluation, only the abbreviations of noise categories are used. Their definitions can be found in Table 2. This evaluation is done for both the Default- and HighPrecision configuration in order to determine the most suitable configuration.

First, as an initial overview of the framework's ability to identify audio accompanied with noise, the framework is evaluated on noise categories without a specified confidence threshold. The results of this are presented in subsection 5.1.

To examine the difference in the performance of both configurations in more detail, a performance analysis based on dominant frequencies in the noise samples is presented in subsection 5.2. Each sample was inspected by plotting its frequency spectrogram using the open-source audio editor Audacity[3]. In this analysis, the goal was to find the frequency range that would influence the characteristic frequencies extracted by the fingerprinting process the most. As SoundFingerprinting essentially looks for local frequency peaks, the frequencies with the highest energy would have the most influence. Therefore, the top 20% of frequency spectrogram was taken as the most dominant frequency range. In the fingerprinting process, frequencies are stored in logarithmically spaced bins. To reflect this behaviour, a small spectrogram window size (2048) was opted for. For categories in which the separate noise samples had different dominant frequency ranges, the average range was taken.

Next, in subsection 5.3, an analysis of the framework's performance for different confidence thresholds is presented. As discussed in subsection 3.2, the confidence threshold specified by the developer of SoundFingerprinting will likely not be suitable for this application. To evaluate the performance of the framework according to different thresholds, each noise category has been evaluated according to the metrics for thresholds between 0.01 and 0.15, in steps of 0.01. Based on this analysis, a new confidence threshold is suggested, which is then used to show the overall robustness and reliability of the framework on all noise categories.

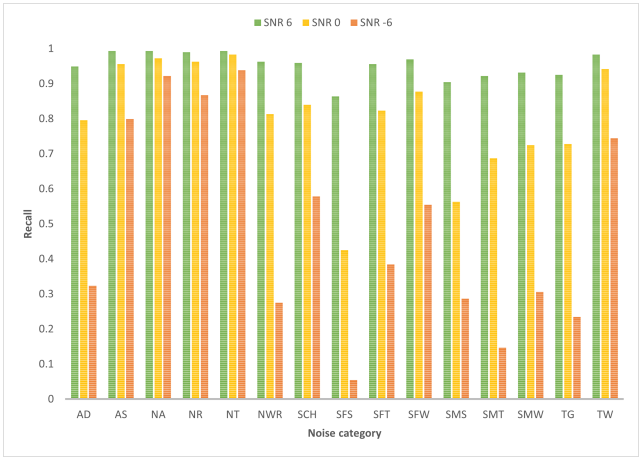
This is followed by an evaluation on tempo-changed and pitch-shifted versions of soundtracks in subsection 5.4. After this, the evaluation is concluded with an analysis

[3]Audacity - a free, open-source audio editor https://www.audacityteam.org/

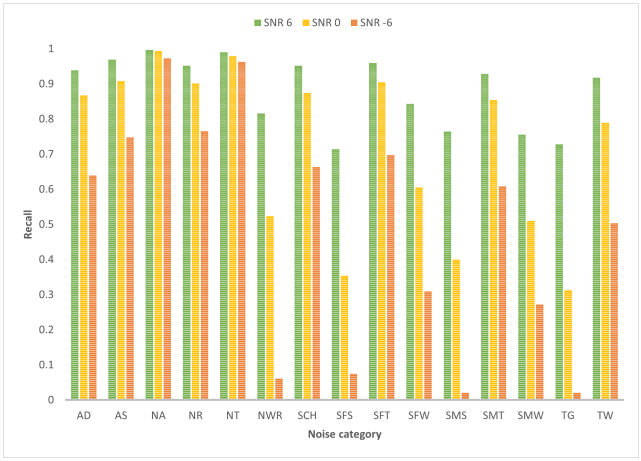
of the search speed and scalability of the framework in subsection 5.5, following the method described in [5].

4.2.3 Hardware specifications

For the experiments presented in this research, a computer was used with the following specifications: a 3.60 GHz 6-Core Processor, with 16GB of DDR4-3200 RAM, run on Windows 10 Pro. The data was stored on an external HDD.



(a) *Default configuration*



(b) *HighPrecision configuration*

Figure 1: *Recall for each combination of noise category and SNR of the benchmark, without a specified match confidence threshold*

5 Results

5.1 Initial performance on noise categories

An initial overview of the framework's ability to identify degraded audio is shown in Figure 1. For these results, no confidence threshold was enforced.

In the case of SNR 6, where the soundtrack is twice as loud as the noise, both configurations perform well, especially the Default configuration, having above 0.9 recall for almost all noise categories. The recall of the HighPrecision configuration for SNR 6 already indicates some problematic areas; categories with a recall of less than 0.85 for SNR 6 also show significant performance drops for SNRs 0 and -6.

The SNR 0 case already proves a more difficult challenge. Whereas the Default configuration mostly stays above 0.65 recall per category, the HighPrecision configuration already shows multiple categories with recall below 0.5. In the final case of SNR -6, where the noise is twice as loud as the soundtrack, both configurations show significant drops in performance, rendering some categories completely unidentifiable.

5.2 Performance analysis based on dominant frequencies

For all but two exceptions, AD and NWR, both configurations handle the Ambient and Nature noise categories well. All other Ambient and Nature noise categories show dominant frequencies in the general range of 1 Hz - 500 Hz, therefore falling mostly outside of the range of the Default configuration and completely outside of HighPrecision's range. However, HighPrecision still shows slightly inferior performance for these categories compared to the Default configuration.

Two cases in which the frequency range of the HighPrecision appears to result in a better performance, are AD and SFT. The dominant frequencies of both categories fall within 1 Hz-750 Hz and 1 Hz-1000 Hz respectively. The results in Figure 1 show that, especially for SNR -6, the Default configuration is less successful in identifying both categories.

However, there are also cases in which this different frequency range has less of an effect. The dominant frequencies of categories SCH (500 Hz-1600 Hz) and SFS (400 Hz-1700 Hz) overlap more with the Default frequency range than the HighPrecision frequency range, yet no notable difference is seen in the performance.

Furthermore, there are several categories which pose an equal challenge to both configurations when considering their dominant frequencies. Categories NWR, SFW, TG and TW all have a very wide range of dominant frequencies, with most noise samples spanning the entire range of both configurations. In these cases, the Default configuration outperforms the HighPrecision configuration.

5.3 Establishing a new confidence threshold

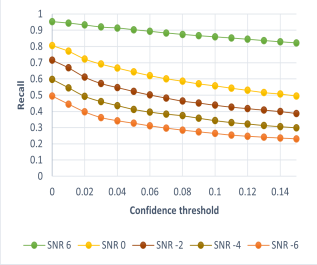
To examine the performance of the framework on negative SNRs in more detail, the experiments from here on will additionally include SNRs -2 and -4.

If we look at the trend of the average recall and precision across all categories for various thresholds in Figure 2 and Figure 3, it becomes evident that the confidence threshold of 0.15 is not optimal for music identification in movies. At thresholds 0.04 and higher, recall only decreases, while precision roughly stays the same.

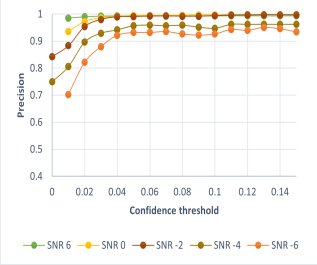
If one were to opt for a maximally robust framework, not using a threshold at all would be the best strategy. On the other hand, if the output of your framework cannot be trusted, i.e. the framework is not reliable, it is debatable whether the output of your framework is actually useful.

We do not know the exact priorities one might have when using this framework for music identification in movies. One might specifically want a recall value of above 80%, whereas others might value the reliability of their framework more,

thus prioritizing precision. Therefore, simply multiplying the two metric values for each threshold and picking the highest value would not be generally applicable. Instead, we aim to suggest a threshold of which we can say that performance only deteriorates at higher thresholds. However, when the confidence threshold increases, recall strictly decreases and precision increases (with the exception of some higher thresholds). Therefore, in picking a threshold, it is assumed that a change in a metric score of less than 0.01 is negligible. For both configurations, according to this assumption, the performance only decreases after threshold 0.05. A final performance overview of both configurations with this threshold is given in Figure 6.

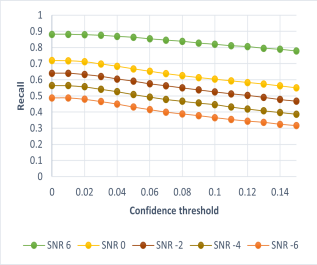


(a) *Recall*

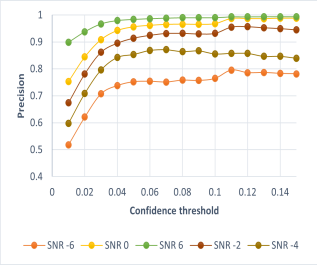


(b) *Precision*

Figure 2: *Trend of recall and precision of the Default configuration across different confidence thresholds*



(a) *Recall*

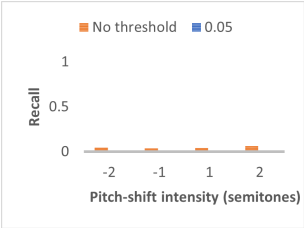


(b) *Precision*

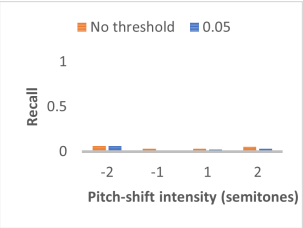
Figure 3: *Trend of recall and precision of the HighPrecision across different confidence thresholds configuration*

5.4 Structural alterations

In Figure 4 and Figure 5, the performance of the framework against pitch-shifted and tempo-changed audio can be found.

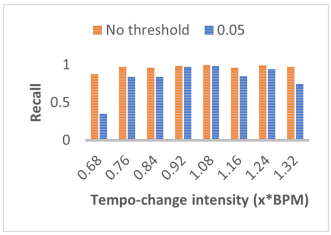


(a) *Default configuration*

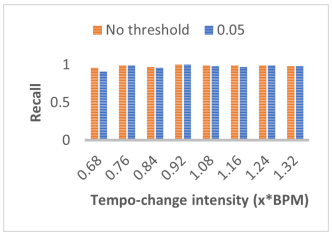


(b) *HighPrecision configuration*

Figure 4: *Recall on pitch-shifted audio*



(a) *Default configuration*



(b) *HighPrecision configuration*

Figure 5: *Recall on tempo-changed audio*

Pitch-shifted audio appears to a weakness of the framework. Both configurations fail to identify almost all queries.

Remarkably, tempo-changed audio, which was one of the weaknesses indicated by the developer, shows a high value for recall. Based on good performance for the tempo-change intensities specified in the benchmark, additional, more severe tempo-changes were examined. There is a notable difference between the two configurations. The considered frequency ranges have no effect on the performance, as the frequencies in tempo-changed audio stay the same. Instead, it appears that the lower minimum of hash table votes of the HighPrecision configuration allows for the identification of severe tempo-changes.

5.5 Search speed and scalability

The results of the search speed analysis of SoundFingerprinting can be found in the table below.

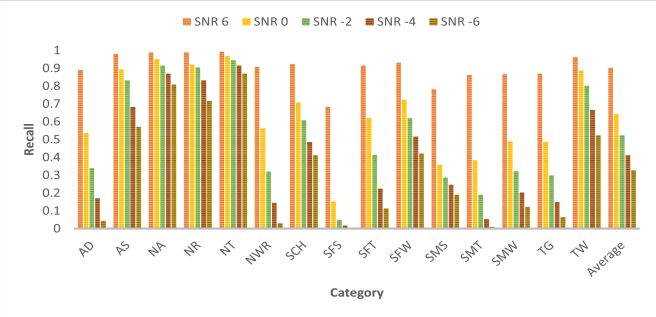
Total tracks	Default	HighPrecision
10	95	117
98	95	133
196	154	169
980	470	420

Table 1: *Average query time (ms) of the Default and HighPrecision configuration for various database sizes*

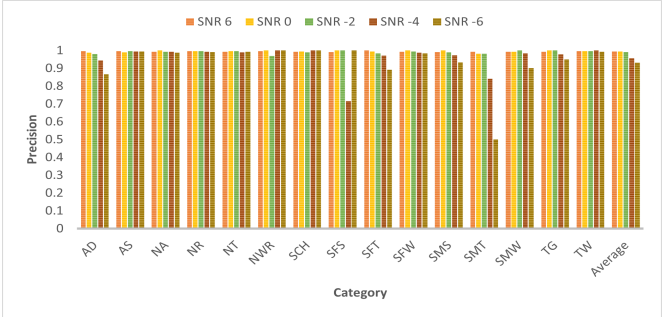
6 Discussion

The graphs displayed in Figure 6 show the performance of both configurations of the framework on identifying noise-layered audio. On average, the Default configuration shows the best robustness to noise for SNRs 0 and 6. On the other hand, HighPrecision is, on average, more robust to noise at negative SNRs, thus making it more suitable for this field. With the newly suggested match confidence threshold of 0.05, the average identification rate of both configurations increased by 0.13 compared to the threshold suggested by the developer of SoundFingerprinting.

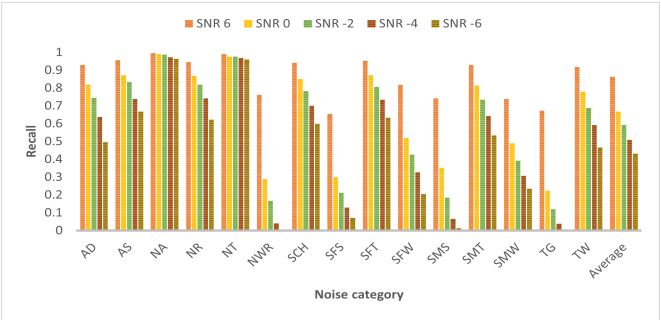
In the performance analysis based on dominant frequencies in the noise samples, there appeared to be a correlation between the performance of a configuration and the dominant frequencies present in the noise sample. This indicates that there is, perhaps, an ideal fingerprinting frequency range for music identification in movies. However, the problem space covered by the categories contained in the benchmark



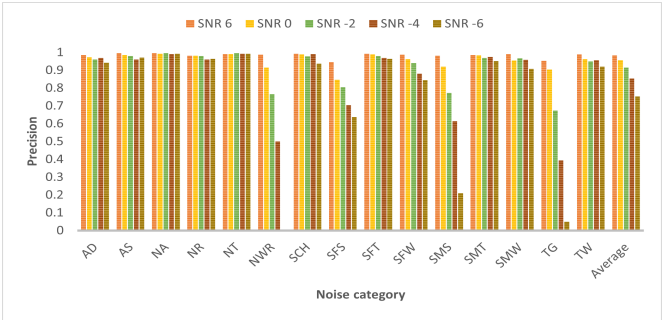
(a) *Recall using the Default configuration*



(b) *Precision using the Default configuration*



(c) *Recall using the HighPrecision configuration*



(d) *Precision using the HighPrecision configuration*

Figure 6: *Recall (left) and precision (right) for all noise categories and SNRs, calculated with a confidence threshold of 0.05.*

was limited. There are many more types of noise present in movies, all possible of hindering identification. Therefore, in order to establish an ideal frequency range, the categories of noise should be expanded.

Furthermore, the framework showed to be robust to tempo-changes. This is likely caused by the windowing method of SoundFingerprinting, following the method described by Haitsma and Kalker [3]. However, as no other windowing settings were evaluated, we cannot say this for sure. Next to that, it appears that the 'minimum hash table votes', as described in subsection 3.2, also play a role in this robustness. This is also not examined further in this paper, thus indicating a direction for future work. The framework was also evaluated on pitch-shifted soundtracks, but the framework showed to be unfit for this task. To further examine the robustness of SoundFingerprinting to signal degradations, its performance should be evaluated on additional mixing and mastering techniques.

Overall, SoundFingerprinting showed unsatisfactory performance in many evaluation categories of the benchmark. Considering that, in movies, the final mix consists of a combination of soundtrack, noise and structural alterations, SoundFingerprinting appears to be unsuitable for music identification in movies. However, one could still further evaluate this suitability by incorporating combinations of noise and structural alterations in the evaluation.

Something worth mentioning is that during an analysis of false matches, we discovered a duplicate track in the song database stored under different names. This caused false positives to emerge, even though the matched songs were the same. Due to the time constraints of the project and the time it takes to run a complete evaluation, this problem was not solved. This means that the calculated metric values are not 100% accurate.

7 Responsible Research

A crucial part of scientific research is ensuring the reproducibility of the research such that its correctness can be verified and that other researchers can build upon this research.

This, however, poses a problem when the data used in the research is copyrighted. This is the case for the collection of movies and their original soundtrack supplied by Muziekweb. As we could not make this collection public, we opted for listing the movies contained in the data set in a spreadsheet. The noise samples, on the other hand, are all licensed under the Creative Commons license. All necessary attributions for these samples were collected in another spreadsheet. In the layering process, only a small fragment of the noise sample looped over the entire song. To allow other researchers to use the same fragments of the samples for data synthesis, the complete set of noise sample fragments has been made available.

All of the aforementioned reproducibility aspects, along with the scripts that were used to generate the evaluation data, can be found on GitLab [4]. If one were to have a collection of movies and their associated soundtrack, then the noise layer-

[4]The research group Gitlab repository containing all aspects regarding reproduction of the benchmark https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-5/rp-group-5-common

ing, pitch-shifting and tempo-changing scripts could be used to generate evaluation data in an identical way.

Finally, the code used to run the evaluation of SoundFingerprinting can be found on the GitLab [5] repository specific to this paper.

8 Conclusions and Future Work

In this paper, the performance of the audio fingerprinting framework SoundFingerprinting has been evaluated according to a benchmark for music identification in movies. The framework was evaluated on actual movie clips and noise-layered, tempo-changed and pitch-shifted version of movie soundtracks. The framework was unable to identify actual movie clips and showed varying performance for the altered versions of movie soundtracks. The framework was evaluated on two configurations, Default and HighPrecision, of which HighPrecision showed to be most suitable for this field.

Furthermore, this evaluation concluded that the confidence threshold suggested by the developer of SoundFingerprinting was not suitable for this use case, after which a new threshold was suggested. However, even with this more suitable threshold, some categories of evaluation data rendered the framework unable to identify the underlying soundtrack. This evaluation focused on identifying problematic areas for identification, evaluating each category of noise and structural alteration separately. In movies, however, the final mix consists of a combination of music, interfering noise and structural alterations. Considering that SoundFingerprinting already showed unsatisfactory performance in separate categories, the current implementation of the framework appears to be unsuitable for music identification in movies.

To further evaluate the suitability of SoundFingerprinting for music identification in movies, the evaluation data set should be expanded. To widen the search space of the evaluation, additional noise categories and mixing and mastering techniques found to be recurring in movies should be evaluated. Next to that, the evaluation data did not contain combinations of noise and/or structural alterations, which would more accurately represent actual movie data. Finally, other frequency ranges, fingerprint windowing settings and the 'minimum hash table votes' could be explored to find a configuration more suitable for this field.

Acknowledgements

I would like to thank Cynthia Liem and Jaehun Kim for their help and supervision during this research. Additionally, I would like to thank the research group, Casper Hildebrand, Ruben Nair, Natália Struharová and Cas Wever.

References

[1] P. Cano and E. Batlle, "A review of audio fingerprinting," *Journal of VLSI Signal Processing*, vol. 41, pp. 271–284, 11 2005.

[2] G. Wikhede, "A comparison of music mastering and film final mixing: How to enhance the listener's experience," 2014.

[3] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," vol. 32, 01 2002.

[4] A. Wang, "An industrial strength audio search algorithm," 01 2003.

[5] C. Hildebrand, T. Huisman, R. Nair, N. Struharová, and C. Wever, "Establishing a benchmark for audio fingerprinting frameworksin the context of music identification in movies," 2021. [Online]. Available: https://bit.ly/3wYz9ZF

[6] S. Baluja and M. Covell, "Content fingerprinting using wavelets," 12 2006, pp. 198 – 207.

[7] C. Sergiu, "Duplicate songs detector via audio fingerprinting," https://www.codeproject.com/Articles/206507/Duplicates-detector-via-audio-fingerprinting/, 06 2020.

[8] C. Shannon, "Communication in the presence of noise," *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, jan 1949. [Online]. Available: https://doi.org/10.1109/jrproc.1949.232969

[9] A. F. C. Jacobs and D. Salesin, "Fast multiresolution image querying," *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995.

[10] C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, no. 3, pp. 462–467, 1968.

[11] M. C. S. Baluja and S. Ioffe, "Permutation grouping: intelligent hash function design for audio image retrieval," 05 2008, pp. 2137 – 2140.

[12] C. Sergiu, "How does audio fingerprinting work," https://emysound.com/blog/open-source/2020/06/12/how-audio-fingerprinting-works.html, 06 2020.

[5] The personal GitLab repository containing the code used to run SoundFingerprinting https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-5/rp-group-5-timhuisman

9 Appendix

Category	Description	Noise code
Ambient	Street noises	AS
	Dining noises	AD
Nature	Rain	NR
	Thunder	NT
	Wind (air)	NA
	Water (river)	NWR
Speech	Male talking	SMT
	Female talking	SFT
	Male shouting	SMS
	Female shouting	SFS
	Male whispering	SMW
	Female whispering	SFW
	Cheering	SCH
Terrain	Gravel noises	TG
	Wood creaking	TW

Table 2: *Noise categories contained in the benchmark.*