Assessing the Suitability of Panako for Music Identification in Movies

Ruben Nair
TU Delft

Abstract

Audio fingerprinting is a technique that allows for fast identification of music. Research concerning this technique first emerged around the 2000s and has lead to several applications, like Shazam [1]. More recently, developments in this area have slowed down, even though there are still new challenges emerging. This paper investigates one of these challenges, music identification of movies, in a systematic way using the open-source fingerprinting framework called Panako. First, clips containing music were extracted from movies and queried using the default settings for Panako. Then, movie soundtracks were modified by layering noise over them, or by time-stretching and pitch-shifting them, and the performance of Panako on these modified audio signals was evaluated using a benchmark. Finally, the best configurations from the previous step were again queried using actual movie clips. These tests showed that both the default configuration and the configurations that performed best on the synthesised data perform poorly in movie music identification. Less than 10% of the clips were identified correctly. The limited scope of this research, combined with the results gathered, show that there should be further investigation into the suitability of Panako for movie music identification.

1 Introduction

Humans have been using fingerprints to identify humans since the late 19th century [2]. The key assumption made that allows for this is that all humans have a unique fingerprint. A fingerprint is a small amount of information that does not encode any personal information about the corresponding human, but can be used to uniquely trace back to them. Around the year 2000, researchers started to investigate and apply a technique called audio fingerprinting, focusing on the fast identification of an audio fragment by linking the fragment to a music piece. An audio fingerprint is similar to a human fingerprint in the sense that it does not contain any information that can be used to reconstruct the original song, but it can be used to identify the original audio file. At that time, this subject was being heavily researched (e.g. [3; 4; 5]). The results of these studies have several different applications. For example, applications like Shazam [1] use audio fingerprinting to allow anyone to identify music that is playing in their surrounding, using their cellphone. Another example is broadcast monitoring, where playlists of songs played on the radio, television or web broadcasts are automatically generated, for example for purposes of royalty collection [4].

However, in more recent years, research in this area has slowed down. The problem of music identification is seen as solved, even though there are still new challenges being discovered. One of these new challenges is the identification of music in movies.

There are several factors that make movie music identification using audio fingerprinting techniques a challenge.
First of all, there is very little existing research tailored towards movie music identification. In [6], three audio fingerprinting algorithms were tested and compared by querying 30-second excerpts of feature-length Hollywood films, recorded with different noise degradations, and matching them to the correct movie audio in the database. However, that approach is different from the research proposed in this paper. The main problem addressed in this paper is identifying the *song* that is present in an audio fragment from a movie, not identifying the *movie* to which the fragment belongs. There is also some literature available that investigates the same audio fingerprinting algorithm as this paper, called Panako. It is tested and compared to other audio fingerprinting algorithms [7], but this is not done in the movie music domain.
The second factor has to do with audio mixing and mastering. As is shown in [8], there are similarities and differences between film mastering and music mastering. Most audio fingerprinting algorithms are designed for music, so the differences with mixing and mastering techniques in films can prove to be an inhibiting factor of the capabilities for the algorithms in the movie music domain.
In this paper, the problem of music identification in movies using audio fingerprinting is investigated systematically, trying to understand how this problem can be solved using the open-source framework called Panako. The research question is: How does Panako perform in music identification in movies? In order to answer this question such that it can be compared to other audio fingerprinting frameworks, a bench-

mark was established. This benchmark, together with the motivation behind it, is described in [9]. The research question is split up into two sub-questions:

1. How well does Panako perform in recognizing music in movies, based on the benchmark?
2. What is the influence of configurable parameters on the performance of Panako?

This paper continues in the following structure. In section 2, background is given into the creation and functionality of the Panako framework. Then, section 3 describes the approach to the investigation of the problem. After that, section 4 presents and discusses the results gathered from the performed tests. This is followed by section 5, which highlights some potential issues regarding responsible research and how they were circumvented. Finally, section 6 concludes the paper and gives some ideas for future work.

2 Panako

The Panako framework is an open-source audio fingerprinting algorithm, developed by Six and Leman [10]. It is based on three previous works. The next three paragraphs will give a short explanation of these works.

First of all, techniques from [11] are used. This paper is written by Shazam [1] and details some of the techniques used by that application in the early 2000s. In this paper they claim that using peaks in the spectrogram representation of an audio signal that contain the highest entropy in its neighbourhood, also known as local maxima, is robust against several types of noise (among others compression, noise and quantization effects). Furthermore, using only the local maxima condenses the amount of information significantly.

In addition to this fingerprinting technique, this paper introduces a scalable method to store and search for fingerprints. Each local maximum in the spectrogram representation of an audio signal, also called an *event point*, is matched with another local maximum in the target zone of the first event point. From this pairing, the two frequencies of the points and the time difference between the points are extracted and run through a hashing algorithm. This hash, combined with the time at which the first point occurred in the audio file, is an audio fingerprint. Several of these fingerprints are created per second of audio, linked to the metadata of the signal and stored in a database. Then, in order to match a new audio signal, the same procedure is applied to that signal and these generated fingerprints are compared to the fingerprints in the database. All exact matches between these fingerprints are counted per signal in the database and the signal with the highest number of exact matches (above a certain threshold, in order to circumvent random hits) is returned as the best match found by the algorithm.

Secondly, the work of [12] is used. This paper focuses on classical music and introduces a technique to identify scores (irrespective of the specific performance of such a score). It relies on a music transcription system for piano and thus is only applied to piano music. However, the ideas it introduces can be applied in other genres too. Different performances of the same score can vary slightly in terms of performance aspects, among which a different tempo (BPM). To accommodate for this, the fingerprints used by this work are different than those in [11]. For the fingerprint extraction from an audio clip, they use 3 "successive events", instead of only two points. This makes the fingerprint tempo independent, since the ratio between the first and second, and first and third event point does not change when the tempo of the audio signal is changed uniformly.

The last paper on which Panako is based is [13]. In this work, they create a spectral representation of the audio using the Constant Q Transform. This technique lends itself well for western music signals: both the frequency bins from the Constant Q Transform and the notes in the western scale are geometrically spaced. As a result, the output of the Constant Q Transform has a constant number of bins per note. An effect of this property is that a pitch shift in the signal corresponds to a constant difference in which bin an audio peak ends up. If the peaks first occurred in bins b_1 and b_2, they now fall into bins $b_1 + K$ and $b_2 + K$ for some constant K. Furthermore, the difference in frequencies (i.e. bins), which is a part of the fingerprint in [13], remains the same:

$$(b_2 + K) - (b_1 + K) = b_2 - b_1 \tag{1}$$

Thus, this approach is robust against pitch-shifting.

The Panako framework combines these techniques and generally works as depicted in Figure 1. The process starts when an audio signal is given as input to the algorithm. Then, during the **Feature extraction**, a spectrogram representation of the audio signal is created, from which local maxima are extracted. The Panako algorithm has two strategies that it can use in order to create the spectrogram representation. One of these strategies is described above, the Constant-Q transform (later also referred to as *NCTEQ*). The second strategy is called the New Fast Fourier Transform (*NFFT*). This strategy uses the Fast Fourier Transform and then runs a streaming minimum/maximum filter over it to find the peaks. The local maxima that are found are then passed to the next step. The procedure in the next phase, the **Fingerprint construction** phase, also depends on the strategy. For the NCTEQ strategy, triplets of these local maxima are combined into a fingerprint which looks as follows:

$$(f_1 - f_2; f_2 - f_3; \tilde{f_1}; \tilde{f_3}; \frac{t_2 - t_1}{t_3 - t_1}); t_1; f_1; t_3 - t_1; id, \tag{2}$$

whereas the NFFT strategy combines two local maxima into the following fingerprint:

$$(f_1; f_2 - f_1; t_2 - t_1); t_1; f_1; t_2; f_2; id \tag{3}$$

In these fingerprints, f_x stands for the frequency bin in which the frequency of the xth event point ended up, and t_x stands for the timestamp of the xth event point. For the NCTEQ

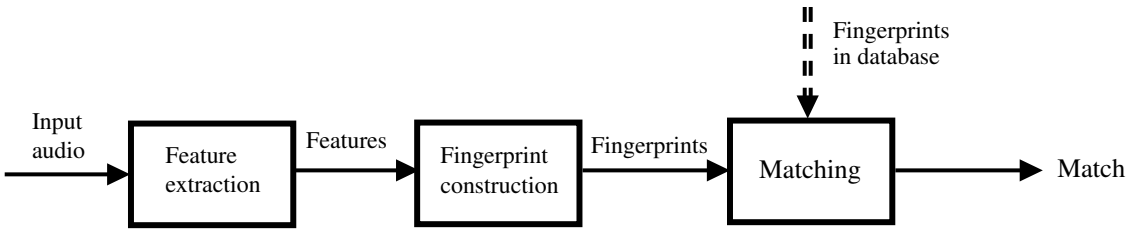


Figure 1: High-level overview of the steps in the audio fingerprinting process for Panako

fingerprints, the terms with tildes over them ($\tilde{f}_1$ and $\tilde{f}_3$) are the bins in which the frequencies of the first and third event point fall after dividing the whole spectrogram into 8 frequency bins. As a result, these terms limit the amount of pitch-shifting that the algorithm can handle, with the benefit that the algorithm can be more discriminative. The part between brackets is hashed into a 32bit and 22bit integer, for the NCTEQ and NFFT strategies respectively. On average, around eight of these fingerprints are generated per second of audio. This set of fingerprints is then fed into the next stage, the **Matching** phase. Here, the hashes of these fingerprints are compared to hashes stored in a database and songs with exact matching hashes are returned. These potential matches are first pruned: songs with four or less exact matching hashes are considered random hits and discarded. All the songs in the resulting list of potential matches are then checked for alignment. The f_1 (the frequency of the first event of a fingerprint) of both fingerprints are compared to find a potential difference. If all event points differ by this same difference, then the audio signals align in frequency, with a pitch-shift difference. For the NCTEQ strategy, fingerprints are also checked for alignment in time. This is done using the t_1 and $t_3 - t_1$ parts of the fingerprints. This information can be used to calculate the offset of the input signal relative to the stored signal. Finally, all matches are returned, ordered by the number of fingerprints that match in both time and frequency.

The described behaviour of the algorithm has been verified for the NCTEQ strategy. In the Panako paper, the algorithm is tested against 30,000 songs that were time-stretched and pitch-shifted. These tests show that after signal modifications of up to 10% (i.e. modification factors between 90 and 110%), the algorithm is still able to identify the correct songs, albeit with significantly lower accuracy.

3 Method

The effectiveness of the Panako framework for music identification in movies was analysed in three steps. All steps made use of a database provided by Muziekweb [14], consisting of 49 movies and their soundtracks, accompanied by 500 random tracks. Each step was performed on the two strategies that Panako has to create a spectrogram, as introduced in section 2: The Constant-Q transform strategy (referred to as NCTEQ) and the New Fast Fourier Transform strategy (referred to as NFFT). These strategies are then compared on their applicability to movie music identification.

The first step was an internal preliminary study. Six of the movies in the database were labelled; start and end times, along with the categories of noise present were annotated. These clips were then extracted from the movies and queried using a certain configuration of Panako to evaluate the performance on actual movie clips.

The second step involved synthesised data. As was mentioned in the introduction, most audio fingerprinting algorithms are designed for music, which is mixed and mastered differently than films. There is very little documentation on the process of film audio mixing and mastering [8], so the synthesised data created for this test does not span all signal manipulations that are present in movies. Instead, the data used for this test consisted of soundtracks layered with specific types of noise that were found to be common in movies during the aforementioned preliminary study. For example, an audio clip of a talking individual was layered with a song from a movie to see the influence of speech on the performance of Panako. This layering was done at different signal-to-noise ratios (SNR). The benchmark consists of soundtracks layered with noise at SNRs of -6, 0 and 6 dB. A negative SNR signifies that the noise is louder than the signal. These aforementioned values therefore specifically denote that the noise is half as loud, equally as loud and twice as loud compared to the soundtrack, respectively. For this test, additional data was generated at an SNR of -9 dB, where the noise is 2.8 times louder than the soundtrack. In [10], it is claimed that Panako is resilient against a certain degree of pitch shifting and tempo change. As a result, this second type of test also consisted of pitch-shifted and tempo-changed soundtracks.
This data was first used in this step on the default parameter settings (i.e. the parameter configurations that are present when downloading the Panako framework) of Panako, to see the score of this configuration on the benchmark criteria. Then, all configurable parameters were investigated to see if this performance could be improved. Although the functionality of Panako is described well in the paper, the same does not hold for the configurable parameters. There is almost no documentation on the function of these parameters, nor on suggested value ranges they can be set to. As a result, a strategy had to be chosen for the parameter tweaking. The chosen strategy was to double and halve each parameter value. This was not always possible. For example, some minimum or maximum values of ranges could not be doubled or halved, as that would produce an invalid range.

Therefore, those parameter modifications were omitted.

As a third and final step, the best configurations found during the second step were tested, individually and all combined, on the real movie clips that were also used for the preliminary study. This was done to see if any improvement in real movie music identification had been made. The best performing configuration in that task was then used to determine the search speed and scalability by querying subsets of the dataset on 4 different database sizes.

During the second step, the performance of a certain configuration of the Panako framework was analysed using the benchmark mentioned in the introduction [9]. This benchmark uses three criteria for evaluation, which are described in the next section.

3.1 Criteria

Robustness The Panako framework should be able to identify the music that is present in a movie, even if there are other audio signals interfering with it. This criterion is measured using the metric *Recall*.

Reliability Reliability has to do with the trust in the correctness of the algorithm. If it finds a match, how likely is it to be correct? This criterion is measured using the metric *Precision*.

Search speed and scalability This criterion revolves around the execution time of the algorithm. On average, how many (milli)seconds does it take the framework to find a match per query, depending on the size of the database? It is calculated as the average query time per database size.

4 Experimental Setup and Results

4.1 Setup

The setup for most of the benchmark runs was the same. The database that Panako uses to match queries to was populated with 1407 songs. This consists of all the soundtracks that were available for this research: the 907 soundtracks of the 49 movies in the provided data set and 500 random other soundtracks.
The test data set was generated as described in [9] and resulted in 14,014 songs (13,230 songs combined with some noise and 784 songs that were pitch-shifted or time-stretched). To run a benchmark on a certain configuration of the Panako framework, all songs in the dataset were queried and the output was analysed[1]. All experiments were performed using Panako version 1.6.

4.2 Hardware specifications

For the experiments in this research, a laptop was used with the following specifications: a CPU with 6 cores (12 logical processors) and 16GB RAM. The laptop is running windows 10. The Panako algorithm was run using WSL2 running the Ubuntu 20.04 distro. All the data was stored on an external HDD.

4.3 Results

4.3.1 Preliminary study

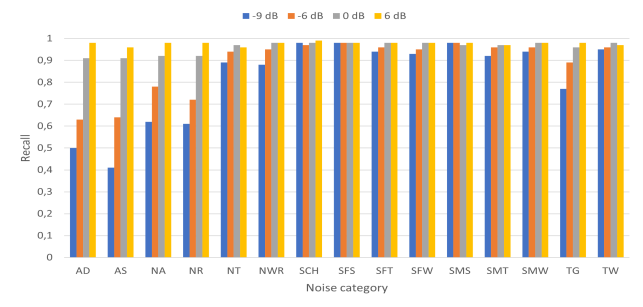
The first test that was performed was done using movie clips. From six movies, clips containing music were manually extracted. These clips were then queried using the default configuration of Panako, for both the Constant Q and the NFFT strategy. Both strategies performed very poorly on the movie clips. Out of 204 clips, the Constant Q strategy was only able to correctly identify 17 of them. The NFFT strategy performed worse; it did not manage to identify any of the clips.
From this small preliminary study, it seems like the default configuration of both strategies of Panako are not suitable for music identification in movies.

4.3.2 Evaluation with synthesised data

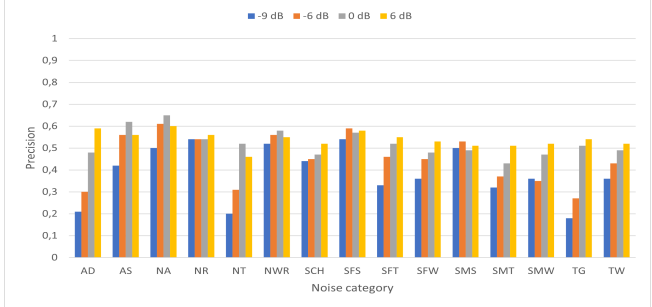
First, in order to establish a baseline of the performance of Panako on the benchmark, the synthesised dataset was tested using the default Panako configurations for both strategies (except that the maximum file size was increased, otherwise most files were too large to be queried). The results of this test can be seen in Figure 2 and Figure 3.

Contrary to the results on real movie data, Figure 2 shows that in this controlled setting, the NFFT strategy outperforms the NCTEQ strategy. Both strategies show high recall, even at low SNRs. The NCTEQ strategy is slightly higher in recall than the NFFT strategy for noises layered with soundtracks at negative SNRs, but has significantly less precision. Even for higher SNRs, over 40% of the matches it finds are incorrect. In contrast, the NFFT strategy rarely returns an incorrect match. For pitch-shifts, NCTEQ has a significantly higher recall, but pays for this in the reliability criterion. Figure 3 shows that the NFFT strategy is unable to find any matches for tempo-changed tracks. This is expected behaviour, since it does not perform time alignment on the fingerprints during the matching phase. As a result, the NCTEQ gives better results in this category.
The NFFT strategy performs very well on all noise categories for high SNRs. For lower SNRs, the recall drops the fastest for the speech categories, sounds of walking on gravel and dining noises. Out of all the speech categories, the strategy performs worst on talking for both male and female voices.

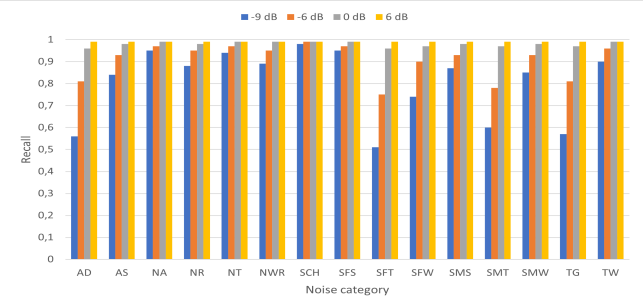
[1] https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-5/rp-group-5-rknair



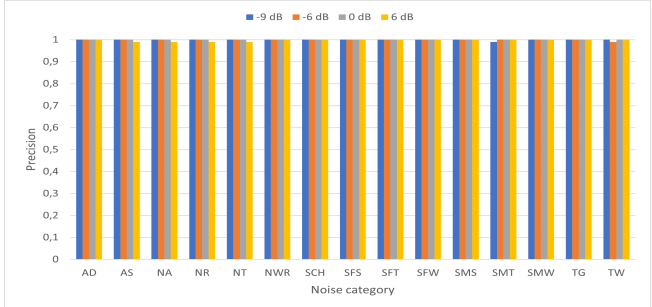
(a) Recall using the NCTEQ strategy with default settings for all noises, at different SNRs



(b) Precision using the NCTEQ strategy with default settings for all noises, at different SNRs

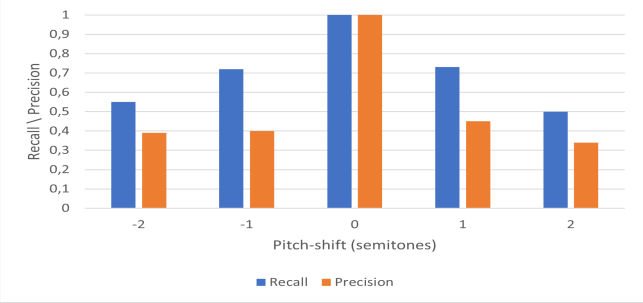


(c) Recall using the NFFT strategy with default settings for all noises, at different SNRs

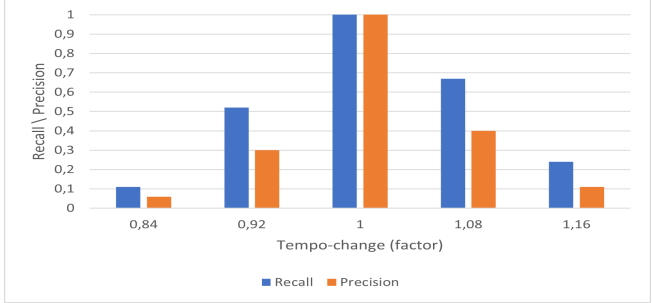


(d) Precision using the NFFT strategy with default settings for all noises, at different SNRs

Figure 2: Recall (left) and Precision (right) against all noises, at different SNRs, for both feature extraction strategies, using default settings. Explanations about the category codes used can be found in Appendix A, or in [9].

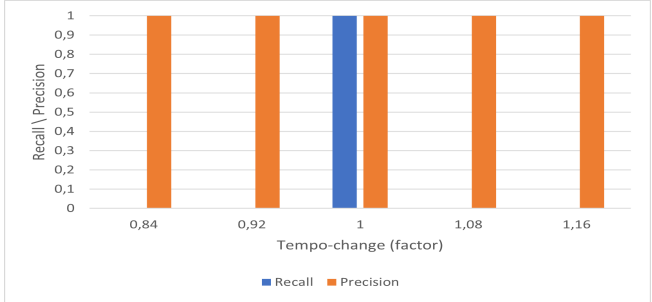


(a) Recall and Precision score for Pitch-shifted tracks using the NCTEQ strategy with default settings



(b) Recall and Precision score for Tempo-changed tracks using the NCTEQ strategy with default settings

(c) Recall and Precision for Pitch-shifted tracks using the NFFT strategy with default settings



(d) Recall and Precision score for Tempo-changed tracks using the NFFT strategy with default settings

Figure 3: Recall and Precision score for Pitch-shifted (left) and Tempo-changed (right) tracks, for both strategies with default settings. More explanation about the chosen values for pitch-shifting and tempo changes can be found in [9].

Once this baseline was established, all configurable parameters were modified. Due to time constraints, a random subsample consisting of 10% of the dataset was used for this test (i.e. 10% per manipulation category, where the SNR for the noise categories was either -9 or -6 dB). This subsample was used for all configurations. The parameters were modified one at a time. The result of parameter changes for the NCTEQ strategy can be found in Figure 4. This graph shows that for the NCTEQ setting, these single changes in parameters can only result in slight improvements in recall or precision, compared to the default settings. The result of parameter changes for the NFFT strategy is shown in Figure 5. Here, three parameter changes result in an improvement in recall of almost 10%, without losing score in precision.

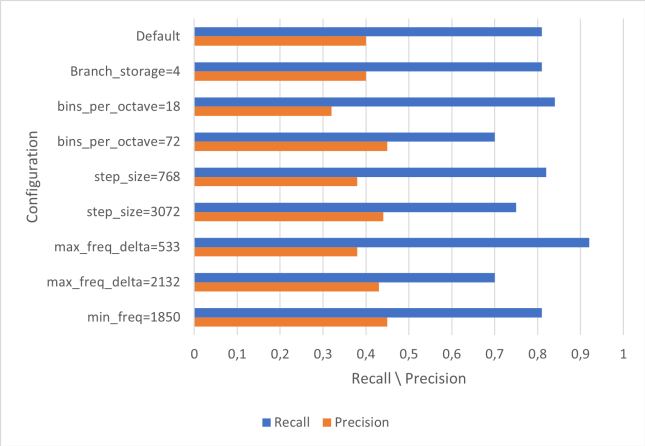


Figure 4: Result of different parameter configurations of Panako on the benchmark score, using the NCTEQ strategy

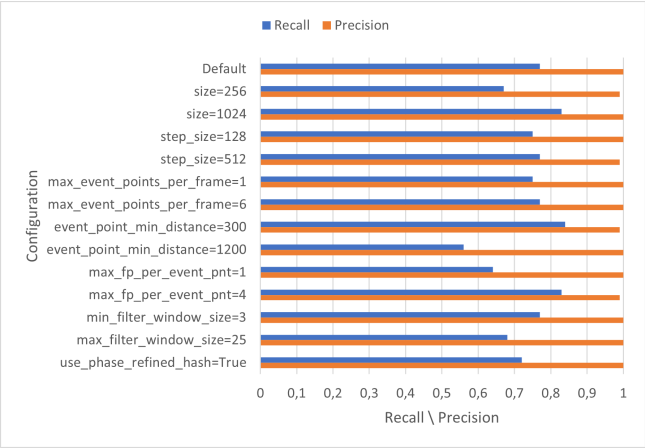


Figure 5: Result of different parameter configurations of Panako on the benchmark score, using the NFFT strategy

4.3.3 Best configurations

For both strategies, the parameter changes that resulted in better performance than the default settings were used to query the real movie clips to see if there was an improvement in movie music identification. They were all applied individually and also all together. The results of this can be seen in Table 1. This table shows that for the NCTEQ strategy, some configurations that performed better on the synthesised data, performed worse on movie clips. This suggests that for this strategy, the noises, pitch shifts and tempo changes that were explored in this paper are not the barrier when it comes to movie music identification. For the NFFT strategy, there is a configuration that can recognize 4 clips as opposed to the 0 matches with the default settings. However, the performance for both strategies is still very poor, with the best configuration finding less than 10% correct matches.

Strategy	Configuration	Number of correct matches
NCTEQ	bins_per_octave=72	8
	step_size=3072	14
	max_freq_delta=533	11
	min_freq=1850	20
	all combined	11
NFFT	size=1024	0
	event_point_min_distance=300	0
	max_fp_per_event_pnt=4	2
	all combined	4

Table 1: Number of correct matches on 204 actual movie clips with the configurations that showed better performance than default settings, and with all these best parameter settings combined.

The best configuration for each strategy on movie data (i.e. min_freq=1850 for NCTEQ and all combined for NFFT) was used to determine the final criterion in the benchmark, the search speed and scalability. This was tested as described in [9] and the results can be seen in Figure 6. This graph shows that the NFFT strategy is almost twice as fast compared to the NCTEQ strategy. There is no clear trend as to how this search speed will change if the database size grows to more realistic sizes of hundreds of thousands of tracks. However, for these relatively smaller sizes, the database size does not appear to be the bottleneck with regard to speed.

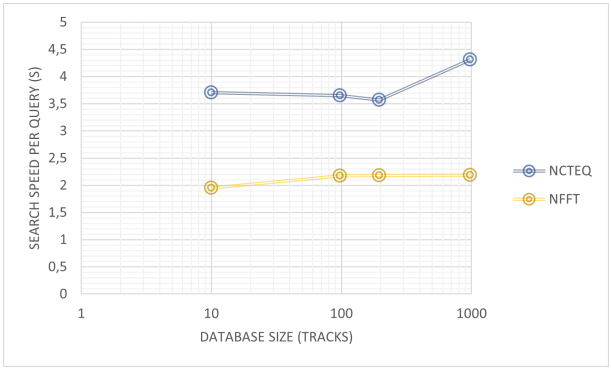


Figure 6: Search speed per query, relative to the number of tracks stored in the database

5 Responsible Research

An integral part of scientific research is allowing for reproducibility, so it can be peer-reviewed and checked by independent sources to verify the correctness. However, achieving that task can introduce some challenges. The most important data used in this research, the soundtracks and movies, were provided by a music library called Muziekweb and fall under copyright protection. Thus, this research cannot link directly to the dataset that was used for evaluation. To combat this, a spreadsheet was created listing the names of all the movies that were present in the dataset. This sheet also links to all the noise files that were layered on top of songs. These noise files were collected from the site freesound.org and all either have a CC0 or Attribution license. Furthermore, the scripts that were used to generate the used dataset from the songs and noise files have been uploaded to a publicly available repository, including instructions on how to run them and recreate the dataset, given that someone has access to the copyrighted data.

6 Conclusions and Future Work

In this paper, the effectiveness of the Panako framework in movie music identification has been investigated. The algorithm was tested on movie clips containing music and on synthesised data consisting of soundtracks layered with different noises and soundtracks that were pitch-shifted or tempo-changed. Using the synthesised data, the impact of configurable parameters on the performance of the algorithm was tested. Based on the results achieved with these tests, no configuration of parameters was found that performed well in movie music identification, with the best configuration identifying less than 10% correct matches. However, the results suggest that within the boundary of the experimental setup, this is likely not caused by the presence of noise, nor by a moderate amount of pitch-shifting or tempo-changing.

These results, combined with the limited scope of this research, point to a variety of possible future works. For this research, only a single type of noise was layered with a movie soundtrack during the evaluation. However, it is very common in movies to have music combined with multiple different sounds playing at once. To get a more representative result, future research in this area can combine different noises and see how this impacts the matching capabilities of Panako. Furthermore, due to time and storage constraints during this research, the size of the experiments was limited. The Panako database contained only 1407 songs during the evaluation, whereas a real music library can have hundreds of thousands or even millions of records stored. Future research could study if the conclusions that are drawn in this paper can hold up in scenarios with more realistic database sizes. Finally, the scope of this research can be extended. There are many other techniques that are used in the film mixing and mastering process (e.g. equalization, compression, noise reduction) [8]. In future research, the impact of these techniques on the effectiveness of Panako (or other audio fingerprinting algorithms) in movie music identification can be investigated.

Acknowledgements

I would like to thank Dr Cynthia Liem and Dr Jaehun Kim for their help and supervision during this research. Furthermore, I would like to thank Casper Hildebrand, Tim Huisman, Natália Struharová and Cas Wever, with whom I have written the paper on the benchmark that is used in this paper [9], for their support.

References

[1] Shazam website. [Online]. Available: https://www.shazam.com/

[2] S. A. Cole, "History of fingerprint pattern recognition," in *Automatic fingerprint recognition systems*. Springer, 2004, pp. 1–25.

[3] P. Cano, E. Batlle, T. Kalker, and J. Haitsma, "A review of audio fingerprinting," *Journal of VLSI signal processing systems for signal, image and video technology*, vol. 41, no. 3, pp. 271–284, 2005.

[4] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *Ismir*, vol. 2002, 2002, pp. 107–115.

[5] P. Cano, E. Batlle, E. Gómez, L. de CT Gomes, and M. Bonnet, "Audio fingerprinting: concepts and applications," in *Computational intelligence for modelling and prediction*. Springer, 2005, pp. 233–245.

[6] T. Pham, M. Giamou, and G. Penn, "An empirical comparison of three audio fingerprinting methods in music and feature-length film," *Canadian Acoustics*, vol. 40, no. 3, pp. 92–93, 2012.

[7] M. Chikanbanjar, "Comparative analysis between audio fingerprinting algorithms."

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

[9] C. Hildebrand, T. Huisman, R. Nair, N. Struharov'a, and C. Wever, "Benchmarking audio fingerprinting implementations for music identification in movies," 2021. [Online]. Available: https://bit.ly/3wYz9ZF

[10] J. Six and M. Leman, "Panako: a scalable acoustic fingerprinting system handling time-scale and pitch modification," in *15th International Society for Music Information Retrieval Conference (ISMIR-2014)*, 2014.

[11] A. Wang *et al.*, "An industrial strength audio search algorithm." in *Ismir*, vol. 2003. Citeseer, 2003, pp. 7–13.

[12] A. Arzt, S. Böck, and G. Widmer, "Fast identification of piece and score position via symbolic fingerprinting." in *ISMIR*. Citeseer, 2012, pp. 433–438.

[13] S. Fenet, G. Richard, Y. Grenier *et al.*, "A scalable audio fingerprint method with robustness to pitch-shifting." in *ISMIR*, 2011, pp. 121–126.

[14] Muziekweb website. [Online]. Available: https://www.muziekweb.nl

A Appendix

Noise abbreviation	Description
AD	Ambient Dining: recording of sounds that can be heard in a restaurant setting
AS	Ambient Street: recording of sounds that can be heard standing in a city, besides a road
NR	Nature Rain
NT	Nature Thunder
NWR	Nature Water River: the sound of water flowing in a river
SCH	Speech Cheering: Sound of people cheering
SFS	Speech, Female Shouting
SFT	Speech, Female Talking
SFW	Speech, Female Whispering
SMS	Speech, Male Shouting
SMT	Speech, Male Talking
SMW	Speech, Male Whispering
TG	Terrain Gravel: Sound of walking over a gravel surface
TW	Terrain Wood: Sound of wood creaking
PS	Pitch-Shift, measured in semitones
TC	Tempo Change: the factor of time scaling

Table 2: Description of noise abbreviations used in result graphs