Delft University of Technology, Bachelor Seminar of Computer Science and Engineering

Analysing the performance of the OLAF framework in the context of identifying music in movies

Casper W.R. Hildebrand[1]

Supervisors:
Cynthia C.S. Liem[1]

Jaehun Kim[1]

[1]Delft University of Technology, Netherlands

27th June 2021

Abstract

This paper presents the findings of a benchmark performed on the audio fingerprinting framework OLAF in the context of movie music. The goal is to find a music identification framework suitable for automatically identifying a song from a movie clip. This research aims to find how well OLAF performs in this context with regard to the criteria determined in the benchmark and to see if the performance can be increased. The OLAF framework makes use of parameters, which are individually tweaked and benchmarked with synthesised data. These findings are then combined to find a set of parameters that is used to perform this same benchmark on real clips from movies. The found parameter setup slightly increased the performance, going from 1 true positive and 133 false negatives in the original setup to 5 true positives, 3 false positives and 126 false negatives.

1 Introduction

Software to identify music has been around for decades. This software, however, is often aimed towards identifying short samples in noisy environments. The use case that will be researched in this work is the identification of music in movies, which brings a different set of challenges with it. Currently, it is a tedious process to identify music in movies. A music identification framework that could identify music in movies would drastically cut down the time required to find metadata about the songs that appear in movies.

OLAF, the audio fingerprinting framework that is subject to this research, is such a music identification framework. This framework will be further introduced in Section 2. It makes use of an audio representation technique called audio fingerprinting. According to [1], an audio fingerprint is defined as "(...) a content-based compact signature that summarizes an audio recording."

An advantage according to [2] is that "Audio fingerprinting techniques aim at successfully performing content-based audio identification even when the audio signals are slightly or seriously distorted." This is useful in the context of movies, as the situations that appear in movies are often scenes with layered audio, e.g. a restaurant scene with music playing in the background or a traffic scene where the driver puts on a radio. Audio fingerprinting frameworks should be robust to these changes to a certain extent.

To measure the performance of OLAF, a benchmark has been set up, such that an arbitrary music identification framework can be benchmarked[1]. In this work, the performance of OLAF according to this benchmark will be researched, as well as the limits to what OLAF is able to identify. This paper aims to find the answer to the following question:

How well does the OLAF framework perform in identifying music in movies?

This main question is divided up into two parts:

1. How can parameters be configured to improve OLAF's performance with regards to the benchmark?
2. How does OLAF perform in practice with identifying samples in real movies?

The paper is outlined as follows. Section 2 introduces the OLAF framework and mentions an extension made on OLAF for this research. Section 3 describes the methodology of how the framework will be evaluated. Section 4 reports the results of performing the benchmark and discusses the findings. Section 5 will discuss the limitations of the research. Section 6 describes how principles of responsible research are applied in the context of this research. Finally, Section 7 concludes the paper and makes recommendations for further research.

[1]It is heavily suggested to read the white paper about this benchmark [3] before continuing to read this work, as this work heavily relies on the metrics and data generated in the white paper.

2 The OLAF framework

In this section, the OLAF framework will shortly be introduced and the past performance will be analysed. This will provide a baseline for how well the framework should perform in practice.

2.1 Introduction of the framework

Overly Lightweight Acoustic Fingerprinting, or OLAF for short, is an audio fingerprinting framework designed and implemented by J. Six in 2020. OLAF is, according to [4]: "(...) a portable, landmark-based, acoustic fingerprinting system released as open-source software". It is based on the algorithm implemented in [5]. According to [4], three points set OLAF apart from other audio fingerprinting frameworks: it can run on embedded systems, it runs fast on traditional computers and it can run in the browser. Of these points, the second is the most applicable to the benchmark, as search speed and scalability is one of the criteria determined in the benchmark.

2.2 Implementation of the framework

To get a better understanding of how OLAF works, a short description of the fingerprinting process of the framework will follow. The process starts with an audio file, OLAF accepts many commonly used audio file formats. Then each audio file is split into blocks, on which a Fast Fourier Transform is applied. The output of this transform is used to extract event points. An event point is a combination of a time index and a frequency. Once some threshold of event points has been met, fingerprints are created from these event points. Whether the audio files are queried, stored in the database or deleted from the database, the fingerprinting process is the same. Since this process is deterministic, the resulting fingerprints from an audio file will always be the same, given the same parameters are used.

OLAF relies on a database library called LMDB, which offers storage based on a B+-tree [4]. According to [6], even with millions of records in the tree, the amount of necessary accesses to the database is still reasonable when searching for a record.

2.3 Prior performance of the framework

The OLAF paper mentions that the speed of the OLAF framework is over thirty times faster than that of Panako [7], an audio fingerprinting framework implemented by J. Six in 2014. Storing data in the database was 1429 ± 205 times faster than real-time. An hour of audio is analysed and consequently stored in about 2.5 seconds. Similarly, with ten thousand songs in the database and four minutes per song, querying was 891 ± 99 times faster than real-time. An hour of audio is queried in about four seconds. Since the queries will be based on music in movies, the clips will mostly range from less than a minute to a few minutes. Therefore, it will generally take a fraction of a second to query a single track. The OLAF paper also mentions that the retrieval performance is similar to that of [5].

2.4 Selectors for OLAF's results

Scripts on GitLab [8] are used to further process the results, counting the true positives, false positives and false negatives. The time of queries is also measured. With this data, all metrics can be quantified or calculated. One problem that occurs in practice is that OLAF returns multiple results. To combat this, three selectors have been implemented. The job of these selectors is to select one of the results OLAF presents. An explanation about the three implemented selectors will now follow.

2.4.1 Ideal selector

This selector will make use of the original song name in the query to determine if a query contains a true positive, false positive or false negative. If a true positive is contained in the results, this selector marks that query as having a true positive. If there are no true positives, it will mark the query result as a false positive if there is one, or a false negative if there are no results. A false positive therefore is taking precedence over false negatives. This is to mimic a real-life selector more closely, as those will usually prefer some result over no result. This indicates how well OLAF performs, with regards to reliability and robustness, at identifying music when information about the correct match is available. In practice, however, this information will not be available, so this selector is not applicable to a real case. Its goal is to act as an upper bound, i.e. showing how well a selector could theoretically perform.

2.4.2 Random selector

A selector should on average never perform worse than random. To have a baseline for this worst-performance, a random selector has been implemented. It picks a pseudo-random result from the list, a selector should never be worse than picking randomly. In cases where OLAF returns a lot of false positives alongside true positives, this random selector will have hugely varying results. Therefore, this selector will be run 100 times for each run of the framework with a certain set of parameters, of which the average will be taken. This helps combat runs whose results deviate far from the average, as theoretically one single run of this random selector could be as good as the ideal selector or miss the right result every time.

2.4.3 Exact matches selector

An exact match is defined as a collision with the database. A single fingerprint can potentially return thousands of hits with the database. In reality, the exact matches can be used to select a result. For every result OLAF returns for a query, the number of exact matches are included. The results are sorted by most matches descending by default, which means the first result OLAF returns for a query is the one with the most exact matches. This is one of the most practical selectors to implement that uses information provided by OLAF to determine a match. The precision and recall of this selector should be between that of the random selector and the ideal selector.

3 Methodology

This section will discuss the methodology of how the research is set up. This includes the criteria that will be used, data generation and how the benchmark will be performed.

3.1 Benchmark criteria

Before a benchmark can be performed to test the performance of the OLAF framework, a benchmark will first need to be set up. To determine the quality of the results of the benchmark, metrics have been determined in the white paper [3].

$$\frac{TP}{FN + TP} \tag{1}$$

Robustness: The framework should be able to identify music, even though there are movie noises over it. The metric is defined as the formula for recall seen in Equation 1, where TP stands for the number of true positives and FN stands for the number of false negatives.

Reliability: This represents how much the result the algorithm gives can be trusted to be correct. Reliability is calculated using the formula for precision found in Equation 2. TP stands for the number of true positives and FP stands for the number of false positives.

$$\frac{TP}{FP + TP} \tag{2}$$

Search speed and scalability: This metric measures the average time it takes to query from start to finish and how the size of the database affects the search speed, i.e. how much slower does the algorithm perform with a larger database? The expectation is that the query time would grow logarithmically with the database size, due to OLAF relying on a B+-tree.

In this work, the definitions above will be used to determine the quality of a music identification framework. These metrics are based on [9] and [10] and are further elaborated on in [3], where the process of defining these metrics is described.

3.2 Generating query data

To be able to conclude anything about what the algorithm performs well or poorly at, the search space has been explored. Some common categories of sounds in movies that play over music have been selected. The selected categories and the reasoning for these categories can be found in [3].

3.3 Performing the benchmark

Now that criteria have been established, the approach to measure the benchmark will be briefly discussed. This approach is discussed more in-depth in [3]. The data set that will be used is provided by Muziekweb and consists of 49 movies and their corresponding soundtracks. These soundtracks are fingerprinted and inserted into the database. This is under the assumption that high-quality soundtracks (i.e. soundtracks that have no distortions applied to them) are available for the movies in the database. Researching the effect of distorted data in the database is out of scope for this research. The research will be executed as follows: All available movie soundtrack data will be stored in the database. This will be done in FLAC format. The synthesised data is in WAV format, which is uncompressed. Since FLAC uses lossless compression and WAV is uncompressed, OLAF should produce the same fingerprints since fingerprints are produced from spectrogram peaks [5]. After the available soundtrack data is stored, querying can begin. The querying process is simple: OLAF will fingerprint the query data and compare these generated query fingerprints to those already stored in the database. If a match is found, it is recorded for further processing.

3.4 Varying the parameters

OLAF has an extensive list of variable parameters that can be optimised for each of the metrics. Depending on the use case, the emphasis may lie on robustness, reliability or search speed. The default parameters, which will be used as a baseline, can be found on the OLAF repository[2]. Due to time constraints, not every combination will be tested. The default values for each parameter will be used, except for the parameter that is being varied. The value for this parameter depends on the default value that is given to that parameter. Generally, the other values that will be tested are the default value both halved and doubled. For each selected parameter a description will follow, alongside the selected values.

Parameter	Abbreviation	Description	Default value	Values to test
Minimum Match Count	MMC	Amount of exact matches required for OLAF to display a match	6	3, 12, 24
Max Fingerprints Per Point	MFP	Determines the amount of times an event point can be reused	3	1, 5
Max Fingerprints	MF	Used for determining the maximum amount of fingerprints the fingerprint extractor can hold	300	150, 600
Max Event Points	MEP	Similar to the previous parameter, but for event points instead of fingerprints	60	30, 120
Event Point Threshold	-	Combines n event points into fingerprints, where n is the value of the parameter. This value is only changed when varying the MEP parameter and uses the value of $\frac{MEP}{2}$	30	15, 60

Table 1: Description of parameters to vary

These parameters have been chosen based on their effects impacting either fingerprint creation, search speed or match selection.

3.5 Hardware & software specifications

The hardware and software specifications of the system the fingerprinting is done on will impact the search speed. OLAF was run on a Linux subsystem (Ubuntu version 20.04) for Windows. The computer used for this research was a laptop equipped with a hexa-core Intel i7-8750H @ 2.20GHz processor. It has 16.0 GB DDR4 RAM, of which 15.9 GB is usable. All data relating to the research was stored on a 5400 RPM HDD.

[2] https://github.com/JorenSix/Olaf/blob/master/src/olaf_config.c

4 Results

Now that preliminary research has been completed and the setup has been described, the performance of OLAF will be evaluated based on the benchmark. First, the results using the different selectors will be discussed. The best performing parameter setup from the most matches selector will be used for any tests related to recall or precision after that. The search speed of different parameter setups and scalability of the framework will be reviewed. Finally, some tests on real movie data will be done.

4.1 Robustness & Reliability

In the results below, the x-axis will consist of all the varied parameters. The letters represent the varied parameter and the number behind it represents the value used for the parameter. The abbreviations for each parameter can be found in Table 1. The ideal selector's scores can be seen in Figure 1. As expected, the reliability is very high: when a match is found, there is a high probability that it is correct. The way the selector has been set up does affect the reliability negatively, because of the preference for false positives over false negatives.

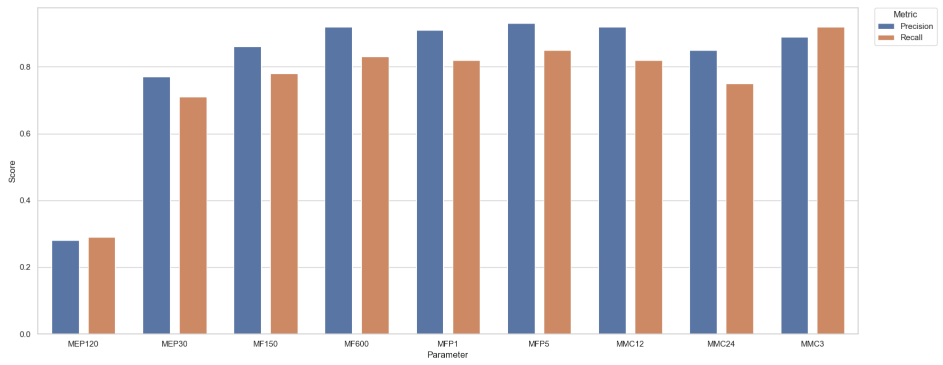

Figure 1: Recall and precision with ideal selector

The performance of the exact match selector, which can be seen in Figure 2, is better than the ideal selector when it comes to precision. The recall, however, is far lower. This selector chooses either the correct match or no match, but only rarely picks a false positive. This is a nice result, as this selector can be implemented based on just the information OLAF provides. For most parameter combinations there is a high probability that it is the correct one when OLAF returns a result.

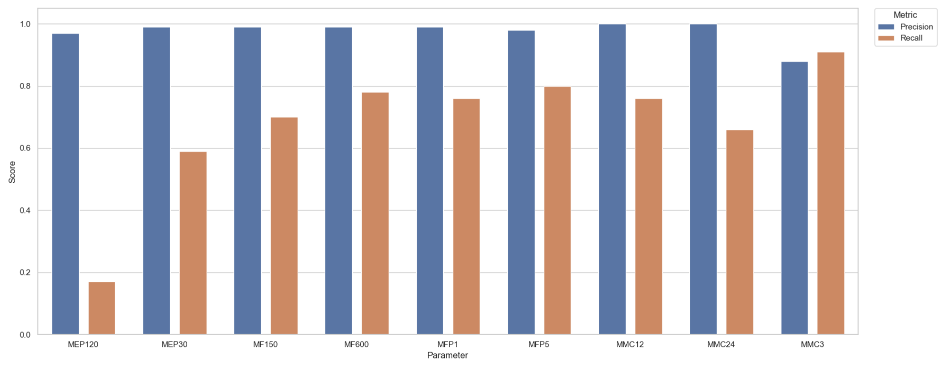

Figure 2: Recall and precision with exact matches selector

Finally, the random selector is considered in Figure 3. The reliability for this selector is also surprisingly high, except for the last parameter configuration. This makes sense, as this parameter configuration allows for a query to return a wider range of results. While the exact match selector can still pick out the most correct result with this configuration, the random selector picks a random result, which will then be incorrect more often due to selecting potential matches with fewer exact matches. For most other parameters, OLAF only returns one result for most queries. That means there is essentially no randomness, meaning the selector performs the same as the two previous selectors.

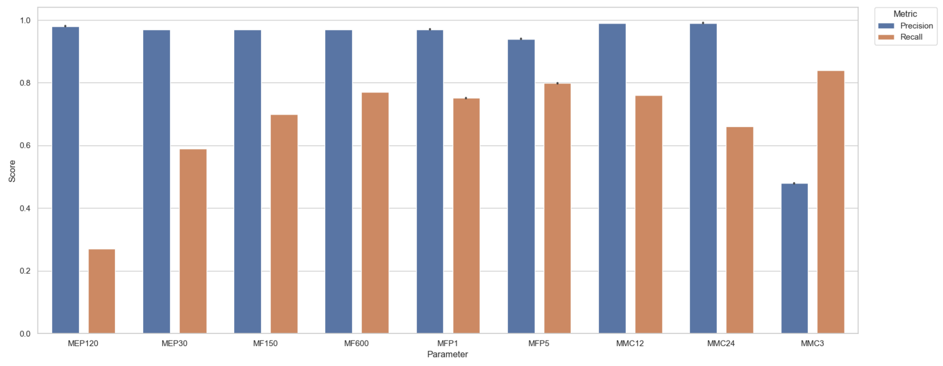

Figure 3: Recall and precision with random selector

Further analysis will be done using the parameter MFP5. This parameter setup sacrifices some precision for a larger gain in recall. Across all noise categories using the most matches selector, the average recall for this parameter is 0.80 and the precision is 0.98.

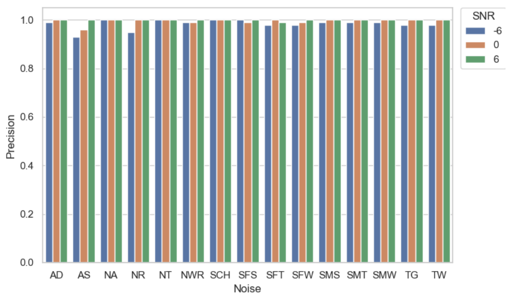

Figure 4: Precision for most matches selector per noise category using different signal-to-noise ratios for parameter MFP5

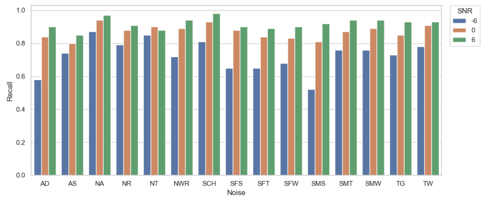

Figure 5: Recall for most matches selector per noise category using different signal-to-noise ratios for parameter MFP5

As expected with such a high overall precision score, for almost every noise category the precision is close to 1 (see Figure 4. Each abbreviation in Figure 4 and Figure 5 is short for a noise category, which can be found in [3]. The AS (street noises) category performs the worst but still has a very high precision. There is a clear pattern in the performance of OLAF when it comes to different signal-to-noise ratios. For every noise category, the negative signal-to-noise ratio also performs the worst. This is logical, as fewer event points will be extracted from the song and more event points will be extracted from the layered sample. A negative signal-to-noise ratio means the sample is louder than the song. This mimics music in the background of a movie scene. A positive signal-to-noise ratio performs better, as more event points will be extracted from the song instead.

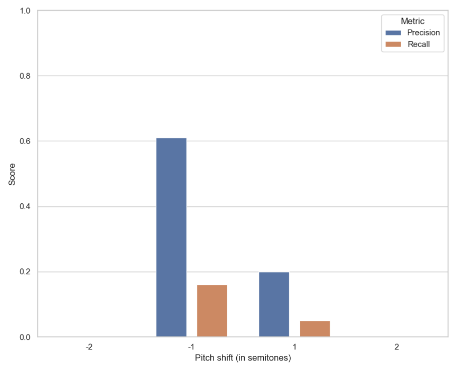

Figure 6: Recall and precision for pitch shifts with most matches selector and parameter MFP5

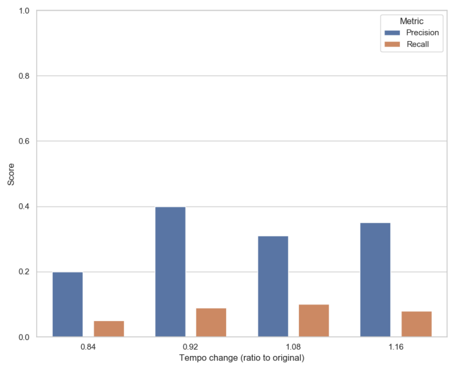

Figure 7: Recall and precision for tempo changes with most matches selector and parameter MFP5

From Figure 6 and Figure 7, it can be seen that OLAF performs poorly when the audio is either pitch shifted or has its tempo changed. If the pitch shifting is too far away from the original track, no results are returned. If there is a pitch shift that is still relatively close to the original track, some results are returned, but the recall and precision are far from what was seen with the layering tests. There is no tempo change value for which no results have been returned, but the recall and precision are still very low for every tested value.

4.2 Search speed & Scalability

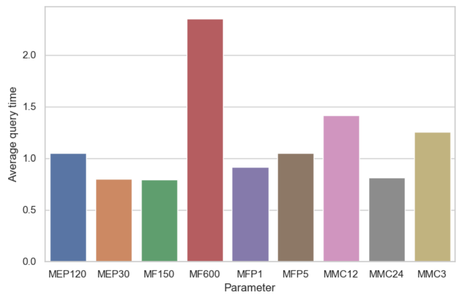

Figure 8: Query times per category for parameter MFP5

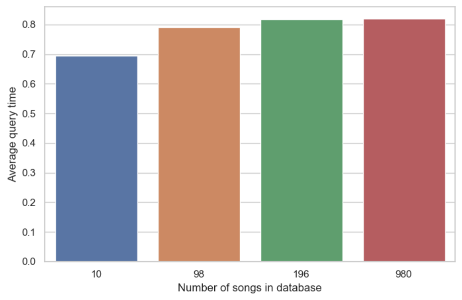

Figure 9: Query times for different database sizes with parameter MFP5

When considering Figure 8, it can be seen that queries on average take around 1000 ms $\pm$ 500 ms for most parameters. There is one clear deviating parameter, which is Max Fingerprints with a value of 600. The query time is about twice as large as that of the default parameter setup. This parameter setup also performs worse regarding robustness and reliability than the default setup, so there is no benefit to using this configuration.

From the tests ran on OLAF regarding scalability (see Figure 9), the claim that OLAF runs fast on traditional computers [4] definitely holds. A database size increase by $\frac{980}{10} = 98$ only results in an average query time $\frac{0.82}{0.70} \approx 1.17$ times longer.

4.3 Tests on real movie data

With this new parameter setup, the default parameter setup can be compared to the original setup regarding recall and precision. 134 clips where music was playing were taken from movies. These are queried against the same database as the benchmark. The results are selected by the same principle as the most matches selector. The default parameter setup performed poorly: one single true positive was found, while OLAF returned 133 false negatives. Since there are no false positives, there is a precision of 1. However, the recall is very low: $\frac{1}{133+1} \approx 0.0075$. There is a slight improvement when using MFP5: using this parameter returned 5 true positives, 3 false positives and 126 false negatives. Therefore, the precision is $\frac{5}{5+3} = 0.625$. The recall is $\frac{5}{5+126} \approx 0.038$.

5 Discussion

In this section, the limitations of the research and the impact of these limitations on the results will be discussed.

5.1 Research data

In the movie data set, there is an issue with duplicate songs on soundtracks. Since file names were used for determining whether a song is a correct match or not, this turns some true positives into false positives. Therefore, the resulting precision could be affected negatively. This would not be an issue when performing the benchmark in real life, as these matches are still the correct result. OLAF has a command to deduplicate a folder, which removes any duplicate audio files. This was discovered late in the research process and due to time constraints, the experiments could not be started over.

The selected noise categories and manipulations only cover a small part of the search space. The noise categories could be expanded and, additionally, layered over one another to create a track more accurate to real movies.

5.2 Parameter setup

The parameters decided in subsection 3.4 only cover a part of the parameters that could be varied. Similarly, the values used for these parameters only cover a part of the possible values. There is a high likelihood that there is some more optimal set of parameters for music identification in movies.

6 Responsible research

To perform this research responsibly, there are two different aspects to consider: the integrity of the research and the reproducibility of the research.

6.1 Academic integrity

When using the term academic integrity, it can be defined as the values, behaviour and conduct of academics in all aspects of their practice [11]. Included in this conduct is properly acknowledging previous work. In this field specifically, that includes acknowledging the creators of the data that has been used in the synthesis process. All noise data was retrieved from Freesound[3] and all contributors will be acknowledged on GitLab [8].
One question that may be asked is whether the synthesised data accurately represents movie data. No complete answer can be given as this could be a whole other work of research. A problem with using data from a medium as broad as movies is that the search space of noise categories and manipulations is very large. Therefore, it is hard to make any general claims about the performance of a framework on movie data as a whole.

6.2 Reproducibility of the research

As this research is conducted on synthesised data based on copyrighted data, the synthesised data can not be published. This makes reproducing the research more difficult. The scripts that have been used to generate the data, as well as the scripts to analyse the results, are available on GitLab [8]. Together with the description provided in this paper, it should be possible to reproduce the results as accurately as possible. According to [5], the hashes that are created from an audio sample are reproducible, therefore, the same database and query data with the same configurations should lead to the same result.

7 Conclusion & Recommendations for future work

In this paper, the performance of the OLAF framework has been analysed according to the benchmark. Both recall and precision delivered satisfactory scores in the benchmark across the noise categories. OLAF can generally handle layered noises well, however, it struggles with pitch-shifted and tempo altered data. The results on actual movie clips regarding recall have slightly improved due to the use of a different parameter setup, but the precision has decreased. This new parameter setup is more useful than the default setup, however, as the new setup returns more matches.

There are many ways to expand on this research, of which some will be highlighted. The noise categories determined in the benchmark are incomplete and could be expanded to include a wider variety of categories. Similarly, only three sounds per noise category have been used, but to get a more representative sample, more sounds from the same category can be used to layer over soundtrack data. Lastly, the final combination of parameters used for querying the actual movie data likely is sub-optimal for the use case. Some optimisation strategy could lead to parameters with better results than have been achieved thus far.

[3]https://freesound.org/

8 Acknowledgements

I would like to thank my supervisors Dr J. Kim and Dr C. Liem for their feedback and help with this research. Other than that, I would like to thank my research group T. Huisman, R. Nair, N. Struharová and C. Wever for the collaborative work and fruitful discussions. Lastly, I would like to thank my friends, family and girlfriend for all their support during this busy period.

References

[1] P. Cano, E. Batlle, E. Gómez, L. de C.T.Gomes, and M. Bonnet, *Audio Fingerprinting: Concepts And Applications.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 233–245. [Online]. Available: https://doi.org/10.1007/10966518_17

[2] T.-K. Hon, L. Wang, J. D. Reiss, and A. Cavallaro, "Audio fingerprinting for multi-device self-localization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 10, p. 1623–1636, 2015.

[3] C. Hildebrand, T. Huisman, R. Nair, N. Struharová, and C. Wever, "Benchmarking audio fingerprinting implementations for music identification in movies," 2021. [Online]. Available: https://bit.ly/3wYz9ZF

[4] J. Six, "Olaf: Overly Lightweight Acoustic Fingerprinting," 2020. [Online]. Available: https://program.ismir2020.net/static/lbd/ISMIR2020-LBD-418-abstract.pdf

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

[6] D. Comer, "Ubiquitous B-Tree," *ACM Comput. Surv.*, vol. 11, no. 2, p. 121–137, Jun. 1979. [Online]. Available: https://doi-org.tudelft.idm.oclc.org/10.1145/356770.356776

[7] 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.

[8] C. Hildebrand, "Personal GitLab." [Online]. Available: https://gitlab.ewi.tudelft.nl/cse3000/2020-2021/rp-group-5/rp-group-5-childebrand

[9] B. McFee, J. W. Kim, M. Cartwright, J. Salamon, R. M. Bittner, and J. P. Bello, "Open-source practices for music signal processing research: Recommendations for transparent, sustainable, and reproducible audio research," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 128–137, 2018.

[10] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system with an efficient search strategy," *Journal of New Music Research*, vol. 32, no. 2, p. 211–221, 2003.

[11] B. Macfarlane, J. Zhang, and A. Pun, "Academic integrity: a review of the literature," *Studies in Higher Education*, vol. 39, no. 2, pp. 339–358, 2014. [Online]. Available: https://doi.org/10.1080/03075079.2012.709495