Indexing Music in Movies Using Audio Fingerprinting: An Audioneex Study

Cas J. Wever
Delft University of Technology

Abstract

Music indexing, the practice of identifying songs contained in an audio sample, is an approach that is widely used. As an underlying technique, "audio fingerprinting" can be used. In this technique, an audio sample is converted to a fingerprint; a smaller representation of the audio. This fingerprint is compared to a database of fingerprinted songs in order to retrieve the original song. In this research, we aim to answer the question of how music in movies can be identified using an audio fingerprinting platform called Audioneex. Two configurable parameters in the framework have been varied, of which the configurations have been evaluated in terms of a benchmark that has been established on synthesized data. The best performing configuration found in the limits of the parameters has been compared to the base configuration of Audioneex. Based on the results, the selected configuration improved the performance of the Audioneex framework by 1 match for one matching algorithm and by 3 matches for the other available matching algorithm.

1 Introduction

Music indexing [1], the practice of identifying songs contained in an audio sample, is widely used. As an underlying technique for music indexing, "audio fingerprinting" can be used. When a framework implements this technique, it converts an audio sample to so-called fingerprints, a smaller representation of the sample. This is then compared to other fingerprints in a database of already existing fingerprints, after which the fingerprint of the song in the database that is most similar is then returned. Music indexing using this approach has multiple uses [2][3]: it can be used to identify audio based on short samples. This property is used in applications like Shazam [4], for example, where one queries a song in a time span of ten seconds to retrieve its name and author. It is also possible to verify the integrity of audio using this technique. When a person alters an audio sample, this can be detected. Next to this, illegal copies of audio can be detected using audio fingerprinting. These are but a few examples for which music indexing based on fingerprinting is used.

1.1 Research

Although it has been used for a long time, music indexing has never extensively been evaluated on music from movies, as is shown in section 2. Since music that occurs in movies is mixed and manipulated to align perfectly with its corresponding scene, this opens up an entirely different field of possible difficulties in the recognition of music. If this field is not evaluated, it might form an impairment on indexing applications similar to those mentioned above, such as the detection of illegal movie distribution. To investigate what issues music from movies introduces and what can be done to mitigate possible challenges, research has been conducted on an audio fingerprinting framework called Audioneex[1]. It is a part of research conducted in a group, where each member has performed research on a different audio fingerprinting framework. In order to perform this overlapping research, a collective benchmark has been established.
This research aims to answer the following question:

How can music in movies be identified using Audioneex?

In order to answer the research question, it has been divided into sub-questions:

1. How does Audioneex perform in practice in music identification in movies?
2. How can parameters be configured to improve Audioneex' performance in terms of the benchmark?

1.2 Method of research

To conduct the research, a collective benchmark has been established in [5]. This is an important step in the evaluation of the performance, as it introduces a way to conduct evaluation research that can be compared. Next to this, it can be informative to compare individual research papers on evaluating music indexing implementations on movies. From such a comparison study, one could identify what choices in the design of an audio fingerprinting framework could lead to a performance peak, for example. Based on this benchmark, the Audioneex framework has been analysed using a dataset of over 10,000 synthesised samples. These samples have

[1]https://www.audioneex.com

been fabricated to resemble music in movies as accurately as possible. The framework is evaluated based on three criteria: *Robustness*, *Reliability* and *Search speed and scalability*. The first two criteria have also been used to evaluate different configurations of the framework, where the best performing configuration in terms of these criteria has been selected to be used in the evaluation of the third criterion. All details about the criteria of the benchmark have been defined in [5]. It is highly suggested to read this work to understand all concepts discussed in this research.

1.3 Structure

The paper is structured in the following manner: first, in section 2, works that are related to this research will be described. Then, in section 3, the algorithm will be analysed. Succeeding this, the methodology of the research will be explained in section 4. After this, in section 5, the results of the research will be presented. As the next item, in section 6, the reproducibility of the research will be analysed. In section 7, the paper will reflect on the results and will draw conclusions based on the findings in section 5. Finally, in section 8, the paper will be concluded and possible future work will be discussed.

2 Related Works

Audio fingerprinting frameworks have been evaluated in different settings. In this chapter, we will investigate some of these evaluations. In literature about audio fingerprinting, different methods are used to evaluate such frameworks. In [6], the proposed system is run on over 100,000 fingerprinted songs that were introduced to alignment shifts. There is, however, no mention of the type of song evaluated. In [7], different types of signal degradations and modifications are introduced in the test samples. Audioneex itself has been evaluated by its developer in [8]. It has been "evaluated using 10,000 query audio clips of 5 to 10 seconds extracted from a set of 1,000 music recordings of different genres". Different audio modifications were applied to these samples, among which equalization, echo, and tempo scaling. This evaluation was not specifically tailored towards the identification of movie music, however. Many such examples can be found where modifications on data or the data itself were not based on the type that occurs in movies. This is due to the fact that research in the field of audio fingerprinting frameworks has been considered done since the late 2000s. The works that do exist, such as the aforementioned examples, are not tailored to the use case of music indexing in movies. The missing part of the evaluation of audio fingerprinting frameworks lies in evaluating on a dedicated testing set of selected or modified songs specific to movies.

3 Algorithm Analysis

In this section, we will introduce an overview of the inner workings of the Audioneex platform. We will only go into depth on specific aspects used by this research only. The full description on which this chapter is based can be found at [8].

An in-depth structural overview of the inner workings of the algorithm can be viewed at Figure 1.

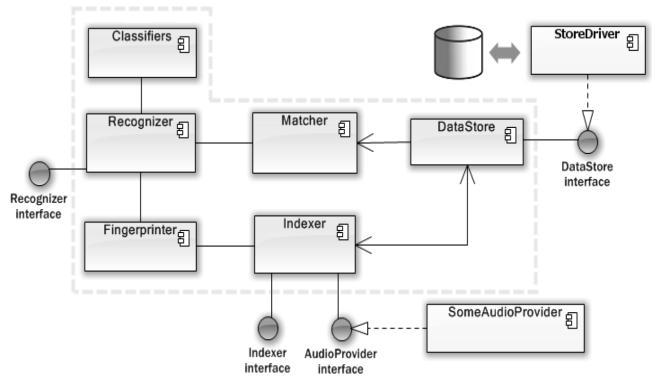

Figure 1: A structural overview of the Audioneex platform by A. Gram, 2019 (https://audioneex.readthedocs.io/en/latest/_images/arch.png)

3.1 Fingerprinting

The fingerprinting and matching algorithms employed by the Audioneex platform are based on the perceptual aspects of sound. These aspects represent the way the human brain represents audio. When an audio sample is fingerprinted, its signal is first resampled to the frequency range of 100-3000 Hz, where "most of the useful information to human listeners lie". After this, the Short Time Fourier Transform is used to transpose the signal to the frequency domain.

Based on these frequencies, the algorithm first looks for frequency peaks in the spectrum, that could indicate "relevant audio events". These audio events might indicate the presence of a music note or another relevant sample feature from which it can be recognized. In the process of finding these events, a parameter, which we from this moment on will call k, is used to determine the sensitivity to the consistency of the event [8]. The reasoning behind this is that these relevant audio events "produce peaks of consistent intensity". This parameter determines how many local maxima in the frequency spectrum are selected as potential points of interest (POIs).

Varying the value of k could have an impact on Audioneex' performance with respect to movie music identification. With a high k value, more frequency peaks are selected as POIs. On one hand, this might result in more information on the sound sample in the final fingerprinted representation, which could improve performance. On the other hand, it could result in the selection of frequency peaks that are less consistent, thus potentially belonging to noise. This would reduce the performance of the framework. With a low k value, fewer frequency peaks are selected as POIs. This might result in less information on the sound sample in the final representation, thus reducing performance. It might also ensure fewer frequency peaks are selected that belong to noise, which would increase the performance of the framework.

After collecting the audio events, non-maximum suppression filtering is applied, which selects the maximum peaks within windows of 400 ms x 340 Hz. This results in the final set of POIs. Following this step, the algorithm applies a neighbourhood mapping, which is based on the *Census Transform* [9], to the set of POIs. This results in a set of binary descriptors of the audio sample, which is a vector that describes the local audio data encapsulated around a certain frequency peak. After obtaining these vectors, a clustering algorithm is applied, which reduces their feature space from 720 to 100 dimensions. These new vectors of 100 dimensions are called the ”auditory words”.

The reduction of the feature space might have an impact on the performance of Audioneex, as it reduces the amount of information we retain from the original fingerprint. This information might be vital to the recognition of music in movies. However, because of the time constraints imposed on this research, we are not considering the dimensionality of the auditory words.

The fingerprint of the sample is formed out of an ordered sequence of local fingerprints. A local fingerprint is formed by an auditory word, the time location of that fingerprint, the quantization error of the auditory word, and another parameter called *f*, which is not discussed by the author. It is finally stored in the database, using the *inverted index* structure, in an inverted list. Each posting in the inverted lists is formed by the local fingerprint, the fingerprint ID, the local fingerprint's time location, and the quantization error of the auditory word.

3.2 Identifying

When an unknown sample is queried, the algorithm quantizes it using the database of auditory words and the Hamming distance. In this process, the sample is transformed into a list of the most similar stored auditory words from the database. For each of these auditory words, the corresponding inverted list is retrieved. Based on more calculations described in [8], the top-k most similar fingerprints are selected.

Next, from subsequences of the sample, fully connected graphs are created. The nodes of these graphs consist of the local fingerprints belonging to that subsequence and the edges of the “position vectors between nodes in the time-frequency space”. Subsequences of the top-k fingerprint candidates are then aligned with the sample subsequences. From these aligned candidate subsequences, graphs are created, which are matched using *Pairwise Geodetic Hashing*. Finally, the most similar fingerprint is returned.

For the identification process, Audioneex provides two types of matching algorithms. Firstly, it provides the algorithm described above, the *MSCALE* matching type. Secondly, it provides the *XSCALE* matching type. The only information provided on these matching algorithms by the developers of Audioneex is that XSCALE is “A modified version of the standard algorithm designed to increase the search speed at large scales by trading off some accuracy.” [2]. These different matching algorithms could have an impact on the performance of Audioneex, as there seems to be a tradeoff between accuracy and search speed. Therefore, we will consider the matching algorithm type a configurable parameter in the evaluation of the framework.

[2]https://audioneex.readthedocs.io/en/latest/api/constants.html#_CPPv4N9Audioneex10eMatchTypeE

4 Methodology

4.1 Established benchmark

To determine the performance of a music indexing framework, a benchmark had to be established.
The criteria chosen for the benchmark are:

1. *Robustness*: This criterion represents how well the framework responds to signal degradation and/or interference. It is measured using Recall.
2. *Reliability*: This criterion represents how much the output of the framework can be trusted to be correct. It is measured using Precision.
3. *Search speed and scalability*: This criterion represents how fast and scalable the framework is with respect to stored comparable fingerprints. It is measured by the average search speed per query per database size.

4.2 Performance analysis

To answer the research question, the Audioneex framework will be run on the evaluation data set belonging to the benchmark. The results of the tests will be quantified using the evaluation criteria as defined in the benchmark. Based on the robustness and reliability criteria, parameter configurations will be tested. The final best performing configuration in terms of the criteria will be used to test the framework on its search speed and scalability.

After testing Audioneex on 114 samples from 4 movies that were manually labeled, it did turn out not to perform well on this data. From the 114 samples, it was only able to correctly identify up to 14. For the purpose of finding a cause for this, this research will be performed on synthesised data. Synthesizing data allows isolating specific areas of mixing and mastering that occur in movies. From this approach, problems that occur when evaluating on actual movie data could be identified individually. This allows finding individual remedies for those problems.

To generate this data, 15 categories of noise that occur in movies have been selected based on frequently occurring noises in movie data labeled by the research group. These categories are displayed in Appendix A. Two songs from each provided movie have been randomly selected, which have been overlayed with three different samples from each noise category. Additionally, three amplitude changes have been performed to simulate mixing and mastering. Apart from these modifications, tempo changes and pitch shifts have been introduced for a subset of the movie soundtracks. A description of the synthesis of the data can be found in [5].

4.3 Setup

Since the proposed way of generating data resulted in over 10,000 records, generating results one record at a time was quite unfeasible. Therefore, it was critical to create a script to run the Audioneex framework and generate results automatically. For this, winux (Windows and Linux) example programs 1 and 3 from the Audioneex GitHub page were used[3]. Program 1, the fingerprinting program, creates fingerprints of

[3]https://github.com/a-gram/audioneex

the movie soundtrack data set and stores them in the database. *Tokyo Cabinet*[4] was used as the fingerprinting database in this program. We have modified the fingerprinting program to store the name of the song belonging to the fingerprint as its metadata. This was required to easily determine if a match was correct in the following steps. Program 3, the identifying program, identifies an audio sample from a specified database of fingerprints. It was used to identify the synthesized data on the fingerprint databases. The output of this program has been modified to ease determining if a match was correct in the following steps. All modified programs and used scripts can be found on the research's GitLab page[5].

4.4 Parameter modification

As mentioned in section 3, there are two possible matching types: MSCALE and XSCALE. These have been used as a configurable parameter. Next to this, the parameter *k* has been varied. We have used the values 4, 5, 6, 7, and 8 for this parameter. As there was no clear mathematical relation between this parameter and the Audioneex framework, we have chosen these five values based on the parameter's base value in the system, *k*=6, and the "best range" as indicated in the documentation of the code by Audioneex' developer, which was ranged from 5 to 7. For the evaluation, multiple fingerprint databases were used based on the parameter configurations. This resulted in eight fingerprint databases. For the first tests, the same value was used for both the fingerprinting and identifying *k*. Each identifying program was thus run on the database generated by the fingerprinting program with the same fingerprinting and identifying *k*. The identifying programs were executed twice; once on the XSCALE configuration, once on the MSCALE configuration.

All parameter combination configurations can be found in Table 1.

XSCALE	*MSCALE*
4	4
5	5
6	6
7	7
8	8

Table 1: Configuration settings with respect to configurable parameter *k*

Since XSCALE was designed to improve search speed at the cost of losing accuracy compared to MSCALE, we expect the MSCALE matching type to perform better in terms of precision and recall but worse in terms of the average query time.

We do not have a clear hypothesis on the performance of the platform with respect to the *k* configuration. As already mentioned, a higher *k* could lead to more information about the sample, but also to information from the noise being included. A lower *k* could lead to less information about the sample, but also to information from the noise being excluded. Additionally, to study the effects of a difference between the identifying and fingerprinting *k*, the fingerprint databases generated by the best performing *fingerprinting k* was taken, on which the *identifying k* was altered and run.

We think evaluating the relation between the fingerprinting and identifying *k* is important, as noiseless, unmixed versions of a soundtrack are used to create the fingerprint data set, while noisy, mixed versions are used as the input for the identification process. Therefore, for fingerprinting, only frequency peaks that actually belong to the original soundtrack are used in the fingerprint, while in the identification process there may be peaks that do not belong to this track. Because of this, we expect different configurations of *k* to have a different impact with respect to the step of the process they are used in.

Finally, to compare the performance on movie data with the original configuration of Audioneex with the performance with the final configuration with the best performance with respect to the benchmark, these two configurations were run on clips taken from the movie "Amadeus".

Finally, we compare the performance of the original configuration of Audioneex with that of the selected best configuration on actual movie data. The configuration is run with both the XSCALE and MSCALE algorithms. The test is performed on 114 clips with music taken from 4 manually labeled movies: Amadeus, Boogie Nights, Samsara, and Ocean's Eleven.

4.5 Hardware specifications

All computing and building processes have been performed on an Ubuntu Windows subsystem. This subsystem was installed on a Windows HP laptop with an Intel i7 core with 2.20GHz, 16 GB RAM, of which 15.8 GB usable. The synthesised data was stored on a 1TB external HDD.

4.6 Execution

After the example programs and parameters were modified, Audioneex was built in static mode, with program setting ID3 and with examples enabled. This was done for every value taken for *k*. After this, the different fingerprint databases were generated by running the fingerprinting programs per configuration on the original movie data set. Finally, the identifying programs were run on the fingerprint databases belonging to the selected *k* value. From this, the output was parsed and combined, resulting in a count of FN, FP, and TP values per noise category, per *k* and per matching algorithm. The execution time of each configuration was recorded.

5 Results

We will present the results of the research in the following structure: First, the performances in terms of precision and recall of the MSCALE and XSCALE matching types are discussed. This will be done with respect to the categories and the values of parameter *k*. Then, the effects of choosing a different value for the fingerprinting *k* than for the identifying *k* will be discussed. After this, the results of the search speed and scalability experiment on the best-performing configuration will be shown. Finally, we will compare the performance

[4]https://dbmx.net//tokyocabinet/

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

of the base configuration with the performance of the selected best configuration on actual movie data.

In figures 2, 3, 4 and 5, the performance of MSCALE and XSCALE are displayed in terms of the fingerprinting and identifying *k*, which is set to the same value. The different performances are set out against the different noise categories, pitch shifts and tempo changes.

k=4 and *k*=5 form two configurations that have lower performance than the other values of *k*. The recall and precision values of the other values for *k* do not differ more than 0.1. Since *k*=6, 7, and 8 are relatively clustered together and generally form a higher trend than *k*=4 and 5, we will perform the analysis on interesting points on *k*=6, 7 and 8.

In the AD category, Audioneex has a high precision value, but a recall value between 0.5 and 0.6. MSCALE performs better in terms of precision in this category. The AS and NR categories have a low value for precision for both MSCALE and XSCALE while in the meantime they have a recall value of over 0.8. Deviating values for precision and recall can be spotted for the NWR category. The precision values fall just below 0.4 for MSCALE and just exceed 0.4 for XSCALE. Additionally, the recall values fall just below 0.5 for MSCALE and just exceed 0.6 for XSCALE. More deviating values are found at the SMS and TG noise categories, where the precision values for both matching types align with the noise categories around them, but where the recall values drop significantly. Finally, one notices the precision and recall values for pitch shifts in the MSCALE matching type drop to a near-zero value. In the meantime, the pitch shifts precision values for XSCALE do not fall below 0.65.

In Table 2, we have displayed the results of the average recall and precision values over all noise categories, pitch shifts, and tempo changes. When we considered these results, we noticed an upwards trend in the precision values and a downwards trend in the recall values, as the value for *k* increased. Therefore, we decided to run additional tests with higher values for *k*. The results of these additional tests are displayed in Figure 6. To properly show the trend, we decided to omit the values 4 and 5 for *k*, as these lie far from the values starting at 6. The identified trend continued up to *k*=10 and started to go backward at *k*=11. The tradeoff between recall and precision makes it difficult to choose a value for *k* that has the highest overall performance. But since we are interested in a higher probability that a match, when given, is correct, we opted for the fingerprinting program where the value of *k* is set to 10.

To study the effects of the fingerprinting and identifying *k* being different values, we ran the fingerprinting program where the value of *k* is set to 10 on the provided database. We then chose five identifying programs, where the values of *k* were set to 8, 9, 10, 11, and 12. Its results are shown in Table 3. The score difference between the configurations does not exceed 0.0027 for recall and 0.0022 for precision. Based on the chosen values, the effects of choosing different fingerprinting and identifying values for *k* seem to be minimal. However, in a real-life case, millions of songs might be queried, where such a small difference might make an impact.

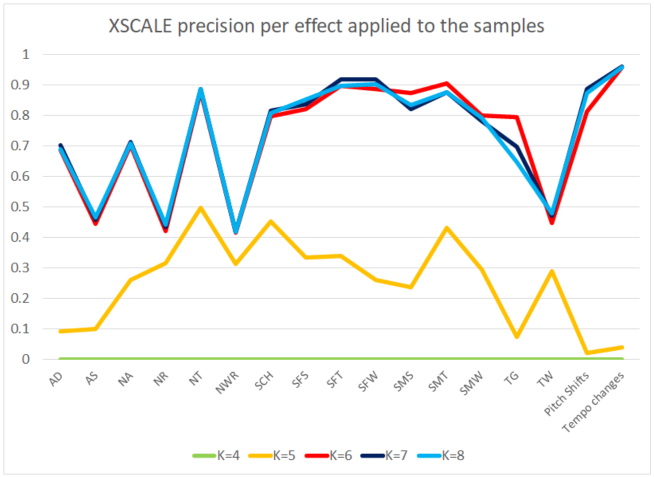

Figure 3: The precision of XSCALE over all modification and manipulation effects, displayed per selected value of *k*

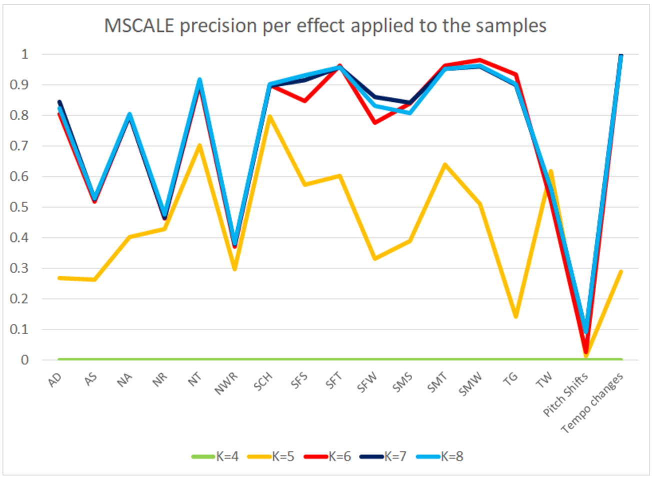

Figure 2: The precision of MSCALE over all modification and manipulation effects, displayed per selected value of *k*

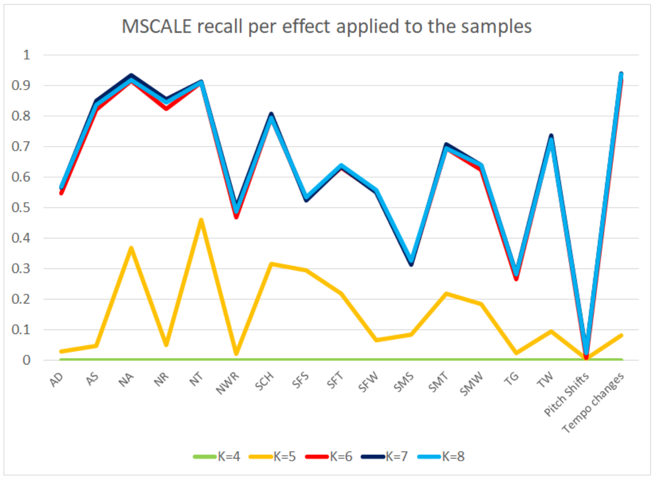

Figure 4: The recall of MSCALE over all modification and manipulation effects, displayed per selected value of *k*

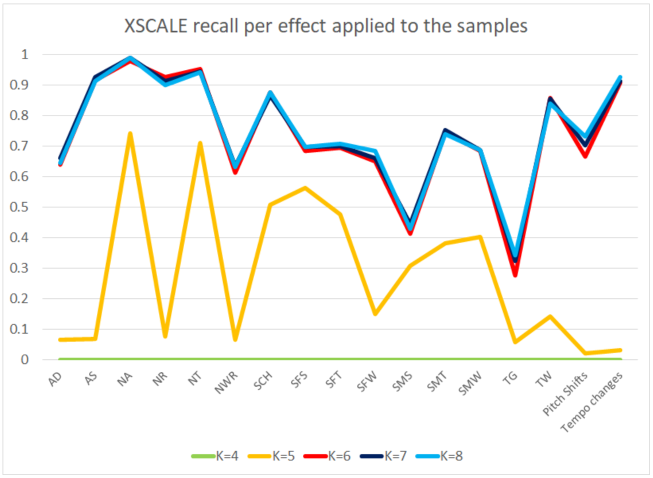

Figure 5: The recall of XSCALE over all modification and manipulation effects, displayed per selected value of *k*

Again, the tradeoff between recall and precision toughens the choice of the best value for the identifying *k*. We decided to opt for a value of 10, again, since we are interested in a higher probability of a correct match.

We ran the search speed and scalability test on fingerprinting databases generated by the fingerprinting program with the value of *k* set to 10 and with the XSCALE matching type.

	Recall	Precision
k=4	0	0
k=5	0.2069	0.4009
k=6	0.6595	0.7288
k=7	0.6787	0.7407
k=8	0.6783	0.7409

Table 2: Configuration settings with respect to configurable parameter *k*, rounded to four decimals

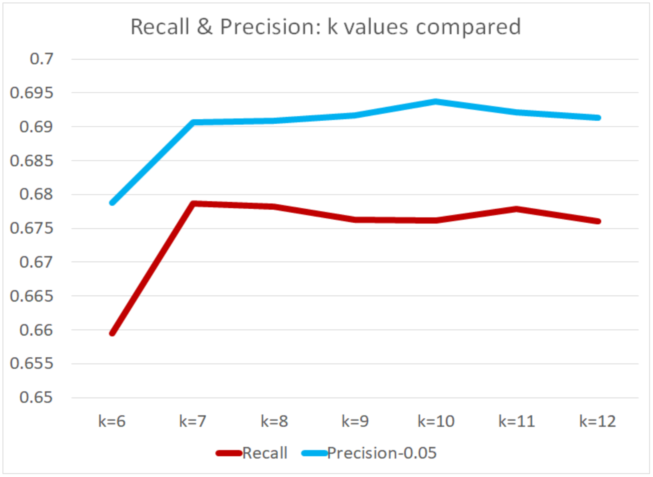

Figure 6: Performance of *k* ranged 6-12 in terms of Precision and Recall, averaged over MSCALE and XSCALE. Precision is subtracted by 0.05 for visualization purposes

The identifying program with the same settings as the fingerprinting program was used. The results of the test have been displayed in Figure 7. The search speed per query increases by about 50 ms when the size of the fingerprint database is increased from 98 to 196 songs. When we add 784 songs to this database, the average query time only increases by about 21 ms. This would imply there is no static increment in average query time per song added to the database.

For the final test, the programs with the values of the fingerprinting and identifying *k* both set to 10 were selected. The test was run on both the MSCALE and XSCALE matching types. The results are shown in Table 4. The selected configuration has only improved on the original XSCALE configuration by one true positive with a difference of 1 false negative. On the MSCALE configuration, it has improved on 3 true positives with a difference of 1 false positive and 2 false negatives.

	Recall	Precision
k=8	0.6771	0.7418
k=9	0.6760	0.7424
k=10	0.6761	0.7437
k=11	0.6762	0.7427
k=12	0.6744	0.7415

Table 3: Configuration settings with respect to indexing parameter *k*=10, rounded to four decimals

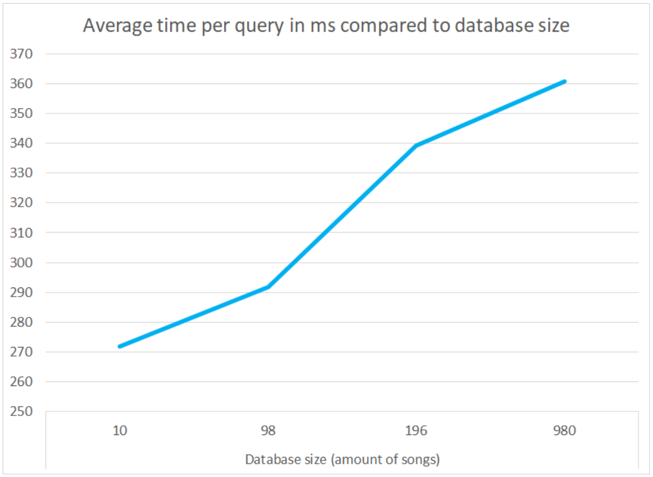

Figure 7: Search speed of Audioneex in ms per fingerprint database size

	FP	FN	TP
Original XSCALE	11	89	14
New XSCALE	11	88	15
Original MSCALE	9	98	7
New MSCALE	8	97	10

Table 4: Final results in terms of FP, FN and TP for the base and selected best configurations of Audioneex

6 Responsible Research

In this paper, we have tried to highlight the methods of evaluation as clearly as possible. Especially with the amount of possible future work, as mentioned in section 8, it is important to be able to run more experiments on Audioneex in a similar fashion. We will talk about several limitations that did present themselves in this research and what we did to mitigate their effects.

6.1 Reproducibility

The samples that were used as noise can be found online[6] and used publicly, as they are licensed under either the Attribution License, Attribution Non-Commercial License, or Creative Commons 0 License. The synthesized data, however, is not, as the movie soundtracks on which this data is based is copyrighted. This complicates the reproducibility of the research. In order to remedy this, a list has been published on the benchmark paper's GitLab page[7] of all noise samples and movie tracks used to generate the data. Additionally, all scripts and modified source codes that have been used in order to generate, fingerprint, or identify the data have been published on the research's GitLab page[8]. If one were to reproduce the research, one could use the aforementioned scripts and programs. Next to this, the music from the used data set might contain different or modified versions of the songs mentioned in the lists, however. This could present different results in possible reproduced research.

6.2 Synthetic data limitations

Next to possible reproducibility issues, we decided to use synthetic data as a substitution for actual movie data, as the framework performed poorly on the latter. This has negative consequences on the reliability of the research. To mitigate this, we have labeled six movies manually, from which a list of noises that occur during music in movies has been established. Different signal-to-noise ratios, pitch shifts, and tempo changes have been used together with these noises to synthesize the data. Each song was combined with a single type of noise, however. Even though this combination was also combined with different signal-to-noise ratios, it does not fully represent music in movies. In movies, different noises can occur in a single sound sample. This also has negative consequences on the reliability of the research. However, the solution applied in this research has led to useful insights into the causes for the poor performance on movie data.

7 Discussion

Although we have attempted to be as extensive as possible in the evaluation of Audioneex, there are some limitations to the research. In this section, we will present these limitations and discuss their impact on the result of the research.

The data that has been used has been structurally synthesised based on various audio modifications and manipulations. It does, however, not fully represent music as used in movies. The modifications and manipulations that were performed only cover a small range of possible alterations that can be performed when mixing audio for a movie. Additionally, the overlaying of multiple noise categories in a single sound sample was not considered in this research. Therefore, the research is not fully representative of research performed on music taken directly from movies.

Next to this, the range of values for both the fingerprinting and identifying k parameter that was used in this research covers but a small amount of all possible values for this parameter. Additionally, next to the parameters considered in this research, there are many more configurations that can be considered. The results of this research are therefore limited to the selected parameters and configurations.

Finally, during the last stages of the research, it was discovered there were some duplicated songs in the database of the benchmark. Regrettably, there was not enough time left for this research to remedy this fact. This could have an impact on the final recall and precision values, as a correct match on a duplicate song could be classified as incorrect, resulting in more false positive values.

8 Conclusions and Future Work

In this research, an audio fingerprinting framework called Audioneex has been evaluated in the context of music in movies.

8.1 Conclusion

The main question we strived to answer was *How can music in movies be identified using Audioneex?* From this subquestion, we attempted to answer how Audioneex performs in practice on movie data and how parameters can be configured to improve the framework's performance. To answer these questions, different evaluation criteria were used to evaluate Audioneex' performance: *Robustness*, *Reliability* and *Search speed and scalability*. The framework was tested on 114 songs taken from different movies, but only correctly identified up to 14 songs. Because of this poor performance, we performed the configurations of different parameters and their evaluations on synthesized data. This enables research to approach possible issues in movie music individually.

To attempt to answer the second subquestion, two different configurable parameters have been identified within the Audioneex framework. These have been altered and combined, from which different configurations emerged. These configurations were individually tested in terms of the criteria, of which the best performing configuration with respect to the selected parameters and their selected ranges was selected. The performance of this configuration was compared to the performance of the base configuration of Audioneex on a data set of songs taken directly from movies. Based on the results, the selected configuration improved the performance of the Audioneex framework by 1 match for the XSCALE matching algorithm and by 3 matches for the MSCALE matching algorithm.

[6]https://freesound.org/

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

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

8.2 Future work

Since this research was very compact and limited, much future work is possible. First of all, more parameter combinations and configurations could be explored in order to determine the optimal configuration of Audioneex when considering movie music. One way this could be done is by considering different parameters and evaluating their combination on the global performance of the platform on the data set. Another way is to look at a specific noise category that performs badly and considering the effect of a specific parameter on the performance within that specific category. This way, combinations of parameters that improve on different categories can be established.

Additionally, the effects of the clustering of information as done by the framework are worth investigating. As this clustering results in information loss, it might have an impact on the performance of Audioneex. This can be considered research on the tradeoff between the fingerprint size and the performance in terms of the established benchmark.

Next to this, other noise categories or signal manipulations can be considered when generating the data set. This allows the researcher to explore the different strengths and weaknesses of the platform in a larger range, thereby possibly identifying problems for movie music indexing.

Furthermore, research on the combination of different noise categories in one sound sample can be conducted. This simulates the actual mixing and mastering that happens more closely.

Finally, research can be performed on an extended range of signal-to-noise ratios, pitch shifts, and tempo changes. This allows for an in-depth evaluation of the effect of these sound manipulations on the performance of Audioneex or an audio fingerprinting framework in general.

9 Acknowledgements

I would like to thank J. Kim and C. Liem for pointing me in the right directions of research. Additionally, I would like to thank the research group, C. Hildebrand, T. Huisman, R. Nair and N. Struharová, for the collaborative work.

References

[1] I. Shakra, G. Frederico, and A. El Saddik. Music indexing and retrieval. In *2004 IEEE Symposium on Virtual Environments, Human-Computer Interfaces and Measurement Systems, 2004. (VCIMS).*, pages 83–87, 2004.

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

[3] P. Dunker and M. Gruhne. Audio-visual fingerprinting and cross-modal aggregation: Components and applications. In *2008 IEEE International Symposium on Consumer Electronics*, pages 1–4, 2008.

[4] A. Wang. The shazam music recognition service. *Communications of the ACM*, 49(8):44–48, 2006.

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

[6] C.J.C. Burges, J.C. Platt, and S. Jana. Distortion discriminant analysis for audio fingerprinting. *IEEE Transactions on Speech and Audio Processing*, 11(3):165–174, 2003.

[7] S. Baluja and M. Covell. Audio fingerprinting: Combining computer vision data stream processing. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 2, pages II–213–II–216, 2007.

[8] A. Gramaglia. A binary auditory words model for audio content identification. https://github.com/a-gram/audioneex, 2014.

[9] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In J. Eklundh, editor, *Computer Vision — ECCV '94*, pages 151–158, Berlin, Heidelberg, 1994. Springer Berlin Heidelberg.

A Table of noise categories

Category	Code	Description
Ambient	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
Nature	NR	Nature Rain
	NT	Nature Thunder
	NWR	Nature Water River: the sound of water flowing in a river
Speech	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
Terrain	TG	Terrain Gravel: Sound of walking over a gravel surface
	TW	Terrain Wood: Sound of wood creaking

Table 5: Description of noise categories and their codes