SUMMARIZING AND COMPARING MUSIC DATA AND ITS APPLICATION ON COVER SONG IDENTIFICATION

Diego Furtado Silva
Departamento de Computação
Universidade Federal de São Carlos
São Carlos, Brazil
diegofs@ufscar.br

Felipe Vieira Falcão, Nazareno Andrade
Departamento de Sistemas e Computação
Universidade Federal de Campina Grande
Campina Grande, Brazil
{felipev,nazareno}@lsd.ufcg.edu.br

ABSTRACT

While there is a multitude of music information retrieval algorithms that have distance functions as their core procedure, comparing the similarity between recordings is a costly procedure. At the same, the recent growth of digital music repositories makes necessary the development of novel time- and memory-efficient algorithms to deal with music data. One particularly interesting idea on the literature is transforming the music data into reduced representations, improving the memory usage and reducing the time necessary to assess the similarity. However, these techniques usually add other issues, such as an expensive preprocessing or a reduced retrieval performance. In this paper, we propose a novel method to summarize a recording in small snippets based on its self-similarity information. Besides, we present a simple way to compare other recordings to these summaries. We demonstrate, in the scenario of cover song identification, that our method is more than one order of magnitude faster than state-of-the-art adversaries, at the same time that the retrieval performance is not affected significantly. Additionally, our method is incremental, which allows the easy and fast update of the database when a new song needs to be inserted into the retrieval system.

1. INTRODUCTION

With the arising of digital music platforms and the consequent growth of music data repositories, we have witnessed an increasing interest in fast methods for mining this kind of data. Organizing, searching, and finding patterns in large repositories require algorithms that are efficient in memory and time while providing an accurate performance.

Several algorithms proposed for different music mining and information retrieval rely on comparing (dis)similarities among the recordings of interest. One issue regarding this approach is the scalability of these methods, since comparing distances is usually a costly procedure.

The cover song identification (CSI) is one task that usually is assessed by similarity-based methods. Most work on advancing the knowledge in CSI is based on creating or adapting new similarity measures and algorithms for comparing the recordings [8, 11, 19] or fusing features or distances to improve the retrieval performance [6, 17, 25]. While some efforts point to the direction of improving CSI runtime [4, 10, 18, 24], the majority of papers on CSI rely on quadratic algorithms to compare each pair of songs [3, 21, 26–28], which may difficult its application on large databases.

One particular idea on speeding up the CSI was presented by Silva, Souza and Batista [24]. The authors proposed a training phase to find what they called "triplets", which are three short excerpts of each original recording that *summarize* them, i.e., that represent the songs with a reduced amount of data. When comparing a new query against these summarized data, the runtime for the distance calculations drastically reduces, since it is proportional to the length of the feature vectors under comparison.

While summarizing tracks can significantly improve retrieval runtime, the triplets technique has some contrivances that make its use difficult. Most importantly, its training phase is costly in time and memory. Also, it considers that more than one original or authorized version of each song is available: the method depends on measuring the distance of each candidate excerpt to several other segments of the same "class label." Finally, once a new recording is added to the dataset, the training phase must be recomputed, since the method to choose the summaries also relies on comparing songs from different labels to analyze the class separability.

In this work, we also leverage the idea of summarizing recordings for fast retrieval. However, we proposed a new suite of methods for summarizing and comparing music data that makes these two usually costly steps simpler and faster. The methods put forward are based on a fast subsequence similarity join algorithm that achieves good retrieval performances and can easily and quickly increment the reference dataset when a new original recording is presented.

© Diego Furtado Silva, Felipe Vieira Falcão, Nazareno Andrade. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Diego Furtado Silva, Felipe Vieira Falcão, Nazareno Andrade. "Summarizing and Comparing Music Data and Its Application on Cover Song Identification", 19th International Society for Music Information Retrieval Conference, Paris, France, 2018.

The suit of methods proposed in this work has the following contributions:

- considerably higher speed to summarize recordings compared to the state-of-the-art;
- a reduction of an order of magnitude in the runtime of the comparison and retrieval phase compared to recent proposals of scalable algorithms;
- no significant loss of retrieval performance is incurred in process of speeding up summarization; and
- because our summarization method solely relies on the recording being summarized, the method is naturally parallelizable and incremental.

Figure 1 illustrates the pipeline of our method.

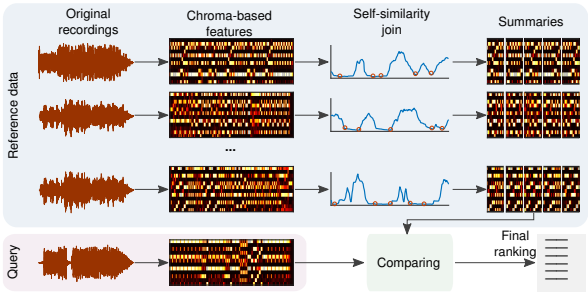


Figure 1: The pipeline of our method consists of summarizing the reference dataset using similarity joins and, for each query, comparing it to the summaries to achieve a ranking by similarity.

This paper is organized as follow: Section 2 introduces a background on the task of cover song recognition and a few related work. Section 3 presents our summarizing and comparing techniques, which composes the proposed suite for fast similarity recover. Section 4 presents the experimental evaluation of our method. Section 5 discuss some ideas on how further improve our proposal. Finally, Section 6 concludes this work.

2. BACKGROUND AND RELATED WORK

The main focus of this paper is the cover song identification (CSI) task. A cover song is a generic name to refer for any recording that is a new version of an original recording. While it may represent an attempt to make a faithfully reproduction of the original work, covers usually widely differs on many characteristics, such as key, timbre, structure and tempo, which makes CSI a difficult task.

To deal with this variation, several methods provide invariance to these issues to CSI algorithms. One example is the Optimal Transposition Index (OTI) [20], which provides key invariance. This algorithm starts by calculating and storing a global pitch profile of the recordings under comparison. Using these profiles, it estimates the difference in key of the songs and transpose one of their feature vectors so that the tracks have the same (estimated) key.

Tempo differences also motivate the need for invariance. Several similarity methods for CSI proposed in the literature are based on a dynamic programming algorithm to dynamically align the compared recordings [6,8,21,25]. This kind of algorithm provides invariance to tempo at the cost of relying on a costly alignment algorithm.

For this reason, techniques which are chiefly concerned with the runtime of CSI systems usually apply lock-step measures such as the Euclidean distance. In these cases, providing tempo invariance in the feature level is a common approach. One option for that is smoothing the feature vectors, an approach that is adopted by some chroma-based features definitions, such as the Chroma Energy Normalized Statistics (CENS) [16].

Many methods for CSI in the literature, if not all, use a pipeline that includes techniques to provide invariances and a distance measure calculation. One example is the already mentioned Triplets [24]. Specifically, this method uses CENS and OTI in its process.

Triplets summarizes the CENS from each reference (original) recording in three short excerpts that are maximally close to excerpts (subsequences) from the same song and far from the excerpts from other pieces. Once a query is presented to the CSI system, it rotates the query according OTI and compares it to the summaries. On the one hand, the summarization significantly improves the runtime to assess a query. On the other hand, the step of finding the triplets is prohibitive thanks to the high number of distance calculations it requires.

Another work from the same research group proposes to identify covers assessing subsequence similarity joins by using the similarity matrix profile, or SiMPle [23]. The SiMPle is a representation of the subsequence similarity join, which is the task of finding the nearest neighbor of each subsequence from a frame-level feature vector among all the subsequences of another vector. Particularly, this operation is called AB-join. The operation of calculating of the best match between a subsequence of a song to itself (disregarding trivial matches) is referred to as self-join.

The join operation returns two pieces of information: the SiMPle and the SiMPle index. While the SiMPle stores the distance of each subsequence to its nearest neighbor, the SiMPle index indicate which subsequence is such neighbor.

The main intuition behind SiMPle in the CSI domain is that comparing a query to its corresponding original version tends to return small distances. Conversely, comparing the query to a recording of another song tends to produce high subsequence distances. As such, the authors defined the distance between a query B and a candidate original recording A as:

$$dist(A,B) = median(SiMPle(B,A)) \qquad (1)$$

While in the SiMPle paper the SiMPle-based CSI is performed over the AB-join operation, the authors demonstrate other applications relying on the self-join procedure. For instance, it is possible to use the SiMPle and the SiMPle index to find music thumbnails (most repeated subse-

quence, given by the neighbors stored in the index) and patterns that are faithfully reproduced in different times of the song or are the most different excerpts in the recording (small and high values in the SiMPle, respectively).

However, computing the distance between a high number of subsequence pairs is a costly operation. To speed up the SiMPle calculation, the authors used MASS, a Fast Fourier Transform-based algorithm to perform a fast subsequence similarity search under the Euclidean distance [14]. The Euclidean distance (ED) between two vectors of n elements is calculated as:

$$ED(A, B) = \sqrt{\sum_{i=1}^{n}(a_i)^2 + \sum_{i=1}^{n}(b_i)^2 - 2(A \cdot B)} \quad (2)$$

When using the ED to find the best match between a subsequence and a long feature vector, we may slide the short sequence along the longer one. The main idea behind MASS is substituting the required dot product by a FFT-based algorithm for calculating the cross-correlation between the compared vectors, often referred to as sliding dot-product. Besides, we can pre-calculate the quadratic sums required by the ED and reuse it when necessary.

Consider n the length of the long feature vector and m the length of the assessed subsequence. The main advantage of using MASS is that it reduces the subsequence similarity search's complexity from $\mathcal{O}(n\,m)$ to $\mathcal{O}(n\,log\,n)$ by using the FFT to find the windowed dot-products instead a brute-force approach [29]. Moreover, these dot-products can be reused to calculate SiMPle even faster [22].

In this paper, we propose a new method that sums up the advantages of Triplets and SiMPle in a single solution. Our algorithm is described in the next section.

3. SUMMARIZING AND COMPARING RECORDINGS

We propose the Summarizing and Comparing (SuCo) method. As the name suggests, it is split into two main procedures. The first one summarizes the reference recordings based on SiMPle. To ensure the time efficiency of our method, we use the faster version of SiMPle's algorithm [22]. The second part refers to the way that a query is compared to these summaries to estimate a distance value between the query and each reference recording.

3.1 Summarization

Summarizing music files is not a novel procedure. Although a few algorithm use it as a intermediate step (e.g. Triplets for CSI), it is usually the final procedure in some specific tasks, such as thumbnailing [1]. In this work, we use the SiMPle to summarize music data as the first step of our algorithm. Using this representation of subsequences similarities, we summarize the music files in five excerpts [1] using two different approaches, which are described in the next sub-sections.

[1] The number of summaries per song is a parameter, which we set to 5. For details, please refer to Section 4.2.

The summarization step can be seen as a training phase, similar to what is done by Triplets. However, while summarizing in Triplets depends on comparing each recording with the entire dataset, our approach processes each recording independently. This implies that *(i)* summarization in SuCo is naturally parallelizable, and *(ii)* once a new original recording is added, it is only necessary to summarize it and add this summary to our set of summaries. The latter operation takes only hundredths of a second.

3.1.1 Thumbnailing

Thumbnailing relies on summarizing a recording in one short segment that best represents it. A thumbnail enables for example a listener to quickly identify a song or its marked characteristics.

A possible definition of a good thumbnail is the excerpt of the song that is most times repeated [1]. Based on this definition, Silva et al. [22] use the subsequence that most appears in the SiMPle index as the thumbnail of a track. This is the subsequence that is most times considered the nearest neighbor of other fragments. In practical terms, the thumbnail is the mode of the SiMPle index. In case of a tie, the subsequence chosen is the one with lowest mean distance to the segments that point at it in the SiMPle index.

In SuCo we use this same step as our first summary, and combine it with four other in a set of five segments that summarizes each recording. The extra segments are consider that is desirable to extract summaries that faithfully describe the song but they need to be diverse. In other words, we avoid describing music with similar excerpts, as this aggregates little information to the retrieval step.

That said, after we choose a summary, we exclude from the choices of next summaries all the subsequences that have it as the nearest neighbor. From a practical standpoint, we keep the count of times that each subsequence is denoted as the nearest neighbor in the SiMPle index and use this information to decide the next summary. When we select a subsequence as a summary, we turn to zero the count regarding each of the subsequences that point at the current summary in the SiMPle index.

Similarly, we make subsequences around each picked summary also ineligible for next summaries. Let p be the position of the current thumbnail and w a constant defined as one-quarter of the assessed subsequences' length. We turn to zero the neighbor count for all subsequences starting at $p_i \in [p - w, p + w]$.

3.1.2 Diverse repeated pattern

While the thumbnail-based summaries rely on the SiMPle index, we also propose a summarization method based on the distances stored by SiMPle. We count on the fact that small distances mean faithfully repeated patterns. These excerpts are very likely to be more precisely repeated because they are more significant to describe the song. For instance, a guitar solo is not similar to other points of the song. Also, that is not a good summary for the cover song recognition, since many covers skip, modify or poorly perform these parts of the song.

As in the thumbnail version, this summarization process also aims to pick diverse patterns. For this, we use a technique based on that proposed by Dau and Keogh [7]. The main idea of this process is to, after picking a subsequence as a summary, increase the value in the SiMPle of the subsequences that are similar to the current summary. This procedure reduces the chance that we choose similar summaries to describe a song.

More precisely, after choosing a summary, we calculate the distance from it to all the subsequences in the song, using MASS. This provides us a vector of distance which has the same length than SiMPle. Then, we normalize this vector in the interval $[0, 1]$, being that the position storing 1 represents the most distant subsequence and 0 appears at the position of the current summary. Finally, we perform a point-by-point division of the SiMPle by this vector. As similar subsequences have a (normalized) distance close to zero, the division will make its relative positions in SiMPle significantly increase its values. Consequently, they will unlikely be chosen as a summary in the next iterations. We refer the reader to the paper that first proposed this procedure [7] for a formal definition of it.

In addition to the described procedure, we also make ineligible the subsequences that are around the picked summaries. Similarly to the thumbnail, we set a region $p_i \in [p - w, p + w]$ of ineligible subsequences. The difference here is that we set as infinite the values at these positions in the SiMPle.

3.1.3 Pitch profiles

In addition to the summaries, the global pitch class profile are also stored in our procedure. This profile is the normalized sum of each bin of the chroma vectors that describe the recording [20]. This is necessary to apply OTI and, consequently, provide invariance to key differences when calculating the distances for a new query recording.

3.2 Distance Calculation

When a new query is presented to the CSI system, SuCo must compare it to each song in the reference database and return a ranking by similarity. In our proposal, we match the query with each summary of each original recording. For this, we again take advantage of the algorithm MASS.

Given a query q, the steps to compare q with the original recordings are:

1. Calculate the global pitch profile of q and the statistics required by MASS, i.e., its sliding quadratic sums and its FFT (which is used to calculate the sliding dot-product). We only need to calculate these values once and, then, use them in every posterior distance calculations.

2. From an original recording r, compare its pitch profile with the profile obtained from q. Then, rotate the chroma vector of each summary of r accordingly.

3. Using the values calculate in the previous steps, calculate the distances between q and each summary of r. Store the lowest distance value, i.e., the distance between the summary and the subsequence from q that best matches it.

4. The final distance between q and r is given by the geometric mean of the distances stored in the previous step. The geometric mean benefits low values in its calculation, favoring the match between q and the reference songs with one or more summaries with a good approximate match.

4. EXPERIMENTAL EVALUATION

This section presents an experimental evaluation of the proposed methods. For the sake of reproducibility, all code and detailed results are provided in a supplementary website [2].

This section is split in distinct topics regarding different phases of our evaluation. First, we describe the datasets we used. Next, we present the experimental setup regarding feature extraction, parameter setting, evaluation measures, and adversary methods. Finally, we present the results of experiments regarding time and retrieval performances.

For simplicity, we refer to our summarizing and comparing methods by thumbnails and diverse repeated patterns as SuCo-thumb and SuCo-repeat, respectively.

4.1 Datasets

The datasets used in our experiments include popular and classical music with different sizes. We opted to use the same data as in the paper that proposed SiMPle, so that results are directly comparable. The datasets are:

- **YouTubeCovers:** This dataset is composed of 50 popular songs of different genres, with seven recordings each. The data is split in pre-defined reference/training and test partitions. The reference set comprises the original (studio) recording and a live version performed by the same artist for each song. The test set is, therefore, composed of five different versions of each song in the dataset.

- **Mazurkas:** This dataset is a collection of classical music. It comprises 2914 distinct recordings of 49 Chopin's Mazurkas obtained from the Mazurka Project [3]. The number of performances of each piece varies between 41 and 95. Unlike the YouTubeCovers dataset, the Mazurkas is not split into default partitions. We therefore assess this dataset using the leave-one-out approach.

4.2 Experimental Setup

To detail our experimental setup, we next describe the applied feature sets, the parameters of our method, and how we compare results.

[2] https://sites.google.com/view/sucomusic
[3] www.mazurka.org.uk/

4.2.1 Feature Extraction

Although some work explores the combination of different features to improve retrieval performance [17, 25], the usual procedure in the literature is to apply chroma-based features [8, 9, 12, 21, 23] that describe pitch perceived over time. More recently, deep learning-based methods have been used to extract cleaner chroma features from audio. In this work, we extract deep-chroma features [13] to describe the recordings in our datasets, using the Madmom tool library [5].

Since the Euclidean distance is sensitive to tempo differences, features are smoothed to provide robustness against this issue. Specifically, we used the technique applied by CENS, which uses a Hann window to smooth features in the time axis. Moreover, we reduced the dimensionality of the temporal axis of the chroma vectors by a factor of five. At the end of this procedure, each recording is represented by a vector containing two (smoothed) deep-chroma values per second of audio.

4.2.2 Parameter Settings

The two parameters of our method are the number and length of subsequences used to describe reference songs. We assessed three values of relative summary length: 10, 20, and 30 seconds (i.e., 20, 40, and 60 consecutive features). Using 10 seconds provide the worst results, and these results are not shown, while they point that using too small windows hampers performance.

Similarly, we assessed results using 1, 3 and 5 subsequences as the summaries set. We notice that while varying the set size does not significantly affect runtime, but the retrieval performance is clearly superior when using five segments.

Given the results of this parameter exploration, we henceforth present the results using five summaries of 30 seconds for the YouTubeCovers data. Because some recordings in the Mazurkas dataset are too short to apply summarization with 30 seconds per summary, we present results on this dataset using 20-second summaries.

4.2.3 Evaluation Measures and Compared Algorithms

Our evaluation consider three common evaluation measures: mean average precision (MAP); precision at 10 (P@10); and mean rank of first correct match (MR1). These measures allow us to compare SuCo against results presented in the literature.

For the YouTubeCovers, our experiments compare SuCo, Triplets [24], SiMPle [23], and a recent technique based on the 2-D Fourier Transform (which we refer as 2D-FT) [19]. The Mazurkas dataset has been less often used in the literature, so it is only possible to compare SuCo against SiMPle with this dataset.

Because previous evaluations of SiMPle did not use deep-chroma features, to isolate whether accuracy improvements in SuCo compared to previous ideas in SiMPle happen due to its use of deep-chromas or algorithmic improvements, we also run SiMPle with the same deep-chroma feature vector used by SuCo.

4.3 Runtime Performance

A central goal of SuCo is to create a fast method for similarity-based music information retrieval. We thus first focus on evaluating its runtime in our datasets [4].

Note that this evaluation does not consider the duration of feature extraction, since it is common to all methods. Also, although we report the total runtime of SuCo's summarizing and comparing procedure, in practice the summarization is only performed once. For the SiMPle-based CSI, the reported runtime regards only the retrieval phase, as it does not rely on a training phase.

In the YouTubeCovers dataset, SuCo-thumb and SuCo-repeat run in 136 and 134 seconds, respectively. On the other hand, the SiMPle-based CSI took 4,192 seconds to assess the same dataset. That is, while our method takes a little more than 2 minutes to run, SiMPle (which is considered a fast algorithm) needs more than one hour. SuCo is more than 30 times faster than SiMPle.

This difference can be further observed in the Mazurkas dataset. While SuCo-thumb and SuCo-repeat take around 10 and 13 hours, respectively, to run the complete process, SiMPle only assess around 240 queries – out of 2914 – in the same runtime. This shows that in this larger dataset, SuCo is two order of magnitudes faster than SiMPle. Indeed, we aborted the execution of SiMPle for this dataset.

To break down the runtime for summarization and comparison in the SuCo pipeline, we isolate the runtime for summarizing in the YouTubeCovers dataset. This dataset has 100 reference recordings, and the summarization step for it takes around 20 seconds. This means that summarizing a new reference track takes around 0.2 seconds, and therefore that incrementing the training set using SuCo is nearly instantaneous.

4.4 Retrieval Performance

After efficiency, our second evaluation criteria is accuracy. Table 1 presents the results for our accuracy evaluation measures in the YouTubeCovers dataset. The SiMPle-deep refers to running regular SiMPle algorithm on the deep-chroma features.

Algorithm	MAP	P@10	MR1
Triplets [24]	0.48	0.13	8.49
SiMPle [23]	0.59	0.14	7.91
2D-FT [19]	0.65	0.14	8.27
SiMPle-Deep	0.78	0.17	3.66
SuCo-thumb	0.65	0.15	5.13
SuCo-repeat	0.74	0.17	3.80

Table 1: Results on the YouTubeCovers dataset

The results of SuCo and SiMPle-deep are a significant improvement over previous results presented in the literature for this dataset. SiMPle-deep presented the best results

[4] This version of SuCo is implemented in Matlab. The experiments were carried out in a desktop computer with 16 Intel(R) Core(TM) i7 − 2600K CPU @ 3.20GHz and 64Gb of memory running Ubuntu 16.04. Also, at any time, there was only one process computing SuCO.

in this experiment, while SuCo-repeat achieved a very similar performance.

Table 2 presents the results for the Mazurkas classical music dataset.

Algorithm	MAP	P@10	MR1
SiMPle [23]	0.88	0.95	2.33
SuCo-thumb	0.83	0.93	2.83
SuCo-repeat	0.85	0.94	2.77

Table 2: Results on the Mazurkas dataset

Like in the YouTubeCovers data, SiMPle displays a slightly better performance than SuCo, in this case even without the deep-learned chroma features.

Taken together, the results on the two datasets point that SuCo is able to attain an accuracy very close to the best performing method while providing a much higher performance, with much lower runtime.

Besides, we notice that we may spend an extra few time to enhance our distance calculation, improving our retrieval performance and not significantly affecting the runtime. We discuss this topic in the next section.

5. ON REFINING THE EUCLIDEAN DISTANCE

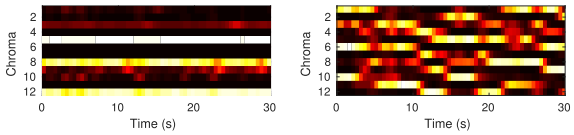
The main purpose of using the Euclidean distance is its time efficiency and the possibility of exploring algorithms to further speeding up its application. However, we understand that it has a negative impact on the efficacy, since ED is sensitive to different distortions in the data.

While we reduce the impact of tempo variances by smoothing the feature vectors, our method is still sensitive to major differences. Applying a distance measure which is more robust to tempo differences in the entire SuCo pipeline could completely compromise our runtime performance. However, we believe that "refining" the distance calculation by some of these functions can improve our results.

To assure this argument, we made a subtle modification of the comparison algorithm. Once the best subsequence match is found using ED, we re-calculate the distance of the matched pair using the Open-End Dynamic Time Warping (DTW) [15] with a relative warping window of 10% of the subsequence length. For simplicity, we will refer to this optimization as SuCo-DTW.

Before presenting the results, we discuss another characteristic that may affect the ED calculation: the complexity of the data. Batista et al. [2] show that more complex time series, i.e., with high variations between consecutive observations, tends to present higher distances to its neighbors. Figure 2 illustrates an example of a simpler and a more complex chroma pattern.

To circumvent this issue, we estimate the complexity of each summary by the standard deviation of its chroma dimensions. The mean complexity of the twelve dimensions is taken as the summary's complexity estimate. Finally, we adjust the distance of a query to each summary by diving it



(a) Low-complex chroma (b) High-complex chroma

Figure 2: Examples of two different summaries with clearly different complexities

by the complexity estimate. For simplicity, we refer to this approach as SuCo-complexity.

To test our assumptions, we ran an experiment using these strategies on the YouTubeCovers dataset. Table 3 presents the results. In this experiment both SuCo-DTW and SuCo-complexity use the diverse repeated patterns as the summarizing method. As previously noted, we did not run the complete SiMPle-deep for the Mazurkas data, due to its impracticable runtime.

Algorithm	MAP	P@10	MR1
SiMPle-Deep	0.78	0.17	3.66
SuCo-DTW	**0.80**	**0.18**	**3.42**
SuCo-complexity	0.78	0.17	5.09

Table 3: Results on the YouTubeCovers dataset

Finally, refining the distance calculation does not severely affect the algorithm runtime. For instance, calculating the estimate complexities and "fix" the whole distance matrix in the YouTubeCovers dataset takes only 0.4 seconds. Also, the total runtime for running SuCo-DTW takes 459 seconds for the entire pipeline. Although it is slower than SuCO-thumb and SuCo-repeat, it is still around ten times faster than SiMPle and presents better retrieval performance. Investigating this kind of efficiency-precision trade-offs is part of our future works, presented with other concluding remarks in the next section.

6. CONCLUDING REMARKS

This paper presents and evaluates SuCo, a suite of methods for summarizing and comparing music data for fast content-based information retrieval. The techniques developed focus on the identification of cover songs. Our results demonstrate that it is possible to achieve results that are close to state-of-the-art algorithms while performing up to two orders of magnitudes faster depending on the dataset. Further improvements on the precision of SuCo with simple post-processing methods were also explored.

Future work may explore the SuCo pipeline in varied applications which rely on similarity comparisons. It also seems promising to further investigate methods to be added to this pipeline to improve precision. Future work may for example experiment with varied features and similarity measures, as well as with fusing different approaches to improve the retrieval efficacy [22].

7. ACKNOWLEDGEMENT

This work was funded by grants #2013/26151-5 and #2018/11755-6 São Paulo Research Foundation (FAPESP).

8. REFERENCES

[1] Mark A Bartsch and Gregory H Wakefield. Audio thumbnailing of popular music using chroma-based representations. *IEEE Transactions on Multimedia*, 7(1):96–104, 2005.

[2] Gustavo EAPA Batista, Eamonn J Keogh, Oben Moses Tataw, and Vinicius MA De Souza. Cid: an efficient complexity-invariant distance for time series. *Data Mining and Knowledge Discovery*, 28(3):634–669, 2014.

[3] Juan Pablo Bello. Audio-based cover song retrieval using approximate chord sequences: Testing shifts, gaps, swaps and beats. In *International Society for Music Information Retrieval Conference*, volume 7, pages 239–244, 2007.

[4] Thierry Bertin-Mahieux and Daniel PW Ellis. Large-scale cover song recognition using the 2d Fourier transform magnitude. In *International Society for Music Information Retrieval Conference*, pages 241–246, 2012.

[5] Sebastian Böck, Filip Korzeniowski, Jan Schlüter, Florian Krebs, and Gerhard Widmer. Madmom: A new Python audio and music signal processing library. In *ACM Multimedia Conference*, pages 1174–1178. ACM, 2016.

[6] Ning Chen, Wei Li, and Haidong Xiao. Fusing similarity functions for cover song identification. *Multimedia Tools and Applications*, 77(2):2629–2652, 2018.

[7] Hoang Anh Dau and Eamonn Keogh. Matrix profile v: A generic technique to incorporate domain knowledge into motif discovery. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 125–134. ACM, 2017.

[8] Daniel PW Ellis and Graham E Poliner. Identifying-cover songs' with chroma features and dynamic programming beat tracking. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, pages IV–1429. IEEE, 2007.

[9] Jiunn-Tsair Fang, Yu-Ruey Chang, and Pao-Chi Chang. Deep learning of chroma representation for cover song identification in compression domain. *Multidimensional Systems and Signal Processing*, 29(3):887–902, 2018.

[10] Eric J Humphrey, Oriol Nieto, and Juan Pablo Bello. Data driven and discriminative projections for large-scale cover song identification. In *International Society for Music Information Retrieval Conference*, pages 149–154, 2013.

[11] Jesper Hojvang Jensen, Mads G Christensen, Daniel PW Ellis, and Soren Holdt Jensen. A tempo-insensitive distance measure for cover song identification based on chroma features. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2209–2212. IEEE, 2008.

[12] Maksim Khadkevich and Maurizio Omologo. Large-scale cover song identification using chord profiles. In *International Society for Music Information Retrieval Conference*, pages 233–238, 2013.

[13] Filip Korzeniowski and Gerhard Widmer. Feature learning for chord recognition: The deep chroma extractor. In *International Society for Music Information Retrieval Conference*, pages 37–43, 2016.

[14] Abdullah Mueen, Yan Zhu, Michael Yeh, Kaveh Kamgar, Krishnamurthy Viswanathan, Chetan Gupta, and Eamonn Keogh. The fastest similarity search algorithm for time series subsequences under euclidean distance, August 2017. `http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html`.

[15] M Muller. Dynamic time warping (DTW). *Information Retrieval for Music and Motion*, pages 70–83, 2007.

[16] Meinard Muller, Frank Kurth, and Michael Clausen. Chroma-based statistical audio features for audio matching. In *Applications of Signal Processing to Audio and Acoustics, 2005. IEEE Workshop on*, pages 275–278. IEEE, 2005.

[17] Julien Osmalsky, Jean-Jacques Embrechts, Peter Foster, and Simon Dixon. Combining features for cover song identification. In *International Society for Music Information Retrieval Conference*, pages 462–468, 2015.

[18] Julien Osmalskyj, Sébastien Piérard, Marc Van Droogenbroeck, and Jean-Jacques Embrechts. Efficient database pruning for large-scale cover song recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 714–718. IEEE, 2013.

[19] Prem Seetharaman and Zafar Rafii. Cover song identification with 2d Fourier transform sequences. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 616–620. IEEE, 2017.

[20] Joan Serra, Emilia Gómez, and Perfecto Herrera. Transposing chroma representations to a common key. In *IEEE CS Conference on The Use of Symbols to Represent Music and Multimedia Objects*, pages 45–48, 2008.

[21] Joan Serra, Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(6):1138–1151, 2008.

[22] D. F. Silva, C. M. Yeh, Y. Zhu, E. Keogh, and G. E. A. P. A. Batista. Fast similarity matrix profile for music analysis and exploration. *IEEE Transactions on Multimedia*, 2018 (in press).

[23] Diego F Silva, Chin-Chia M Yeh, Gustavo Enrique de Almeida Prado Alves Batista, and Eamonn Keogh. Simple: Assessing music similarity using subsequences joins. In *International Society for Music Information Retrieval Conference*, pages 23–29, 2016.

[24] Diego Furtado Silva, Vinícius Mourão Alves de Souza, and Gustavo Enrique de Almeida Prado Alves Batista. Music shapelets for fast cover song regognition. In *International Society for Music Information Retrieval Conference*, pages 441–447, 2015.

[25] Christopher J Tralie. Early MFCC and HPCP fusion for robust cover song identification. In *International Society for Music Information Retrieval Conference*, pages 294–301, 2017.

[26] Christopher J Tralie and Paul Bendich. Cover song identification with timbral shape sequences. In *International Society for Music Information Retrieval Conference*, pages 38–44, 2015.

[27] Wei-Ho Tsai, Hung-Ming Yu, Hsin-Min Wang, and Jorng-Tzong Horng. Using the similarity of main melodies to identify cover versions of popular songs for music document retrieval. *Journal of Information Science & Engineering*, 24(6), 2008.

[28] Fan Yang and Ning Chen. Cover song identification based on cross recurrence plot and local alignment. *J. East China Univ. Sci. Technol*, 42(2):247–253, 2016.

[29] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Mining and Knowledge Discovery*, 32(1):83–123, 2018.