MUSIC SHAPELETS FOR FAST COVER SONG RECOGNITION

Diego F. Silva **Vinícius M. A. Souza** **Gustavo E. A. P. A. Batista**
Instituto de Ciências Matemáticas e de Computação – Universidade de São Paulo
{diegofsilva,vsouza,gbatista}@icmc.usp.br

ABSTRACT

A cover song is a new performance or recording of a previously recorded music by an artist other than the original one. The automatic identification of cover songs is useful for a wide range of tasks, from fans looking for new versions of their favorite songs to organizations involved in licensing copyrighted songs. This is a difficult task given that a cover may differ from the original song in key, timbre, tempo, structure, arrangement and even language of the vocals. Cover song identification has attracted some attention recently. However, most of the state-of-the-art approaches are based on similarity search, which involves a large number of similarity computations to retrieve potential cover versions for a query recording. In this paper, we adapt the idea of time series shapelets for content-based music retrieval. Our proposal adds a training phase that finds small excerpts of feature vectors that best describe each song. We demonstrate that we can use such small segments to identify cover songs with higher identification rates and more than one order of magnitude faster than methods that use features to describe the whole music.

1. INTRODUCTION

Recording or live performing songs previously recorded by other composers are typical ways found by several early-career and independent musicians to publicize their work. Established artists also play versions composed by other musicians as a way to honor their idols or friends, among other reasons. These versions of an original composition are popularly called *cover songs*.

The identification of cover songs has different uses. For instance, it can be used for estimating the popularity of an artist or composition, since a highly covered song or artist is an indicative of the popularity/quality of the composition or the author's prestige in the musical world. In a different scenario, a search engine for cover songs can help music consumers to identify different versions of their favorite songs played by other artists in different music styles or language.

Musicians that upload cover versions to websites such as *YouTube, Last.fm* or *SoundCloud* frequently neglect that the original songs may be copyright-protected. Copyright is a legal right created by the law that grants the creator of an original work (temporary) exclusive rights to its use and distribution. Legally speaking, when an interpreter does not possess a license to distribute his/her recording, this version is considered illegal.

For these reasons, cover song recognition algorithms are essential in different practical applications. However, as noted by [12], the automatic identification of cover songs is a difficult task given that a cover may differ from the original song in key, timbre, tempo, structure, arrangement and language of the vocals.

Another difficulty faced by automatic cover song identification systems, particularly those based on expensive similarity comparisons, is the time spent to retrieve recordings that are potential covers. For instance, websites such as *YouTube* have 300 hours of video (and audio) uploaded every minute [1] . A significant amount of these videos is related to music content. Therefore, cover song identification algorithms have to be efficient in terms of query processing time in order to handle such massive amounts of data.

This paper proposes a novel algorithm to efficiently retrieve cover songs based on small but representative excerpts of music. Our main hypothesis is that we can characterize a specific music with small segments and use such information to search for cover songs without the need to check the whole songs.

Our hypothesis is supported by the success of a similar technique used in time series classification, named *shapelets* [16]. Informally, shapelets are time series subsequences, which are in some sense maximally representative of a class. For time series, shapelets provide interpretable and accurate results and are significantly faster than existing approaches.

In this paper, we adapt the general idea of shapelets for content-based music retrieval. For this, we evaluate several different ways to adapt the original idea to music signals. In summary, the main contributions of our proposal are:

- Our method adds a training phase to the task of content-based music information retrieval, which seeks to find small excerpts of feature vectors that best describe each signal. In this way, we make the similarity search faster;

[1] www.youtube.com/yt/press/statistics.html.

- Even with small segments, we demonstrate that we can improve the identification rates obtained by methods that use features to describe the whole music;

- We show how to use our proposal along with a specific retrieval system. However, we note that our method can be added to any algorithm based on a similar sequence of steps, even methods to further speed-up the query. To do this, we simply need to apply such an algorithm on the shapelets, instead of the complete features vectors.

2. BACKGROUND AND RELATED WORK

The task of cover song recognition can be described as the following: given a set, S, of music recordings and a query music, q, we aim to identify if q is a version of one of the songs in S. Thus, a cover song recognition system can be considered a querying and retrieval system.

The state-of-the-art querying and retrieval systems can be divided into five main blocks [12]: i) feature extraction; ii) key invariance; iii) tempo invariance; iv) structure invariance; and v) distance calculation. Figure 1 illustrates these steps. This general framework leaves open which method will be applied in each step.

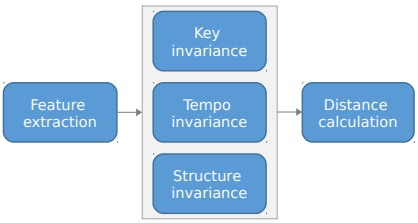

Figure 1. General retrieval system blocks. The feature extraction and distance calculation are required and should appear in this order. The other ones may provide best results, but are optional

Feature extraction is a change of representation from the high-dimensional raw signal to a more informative and lower-dimensional set of features. Chroma-features or pitch class profiles (PCP) are among the most used features for computing music similarity. These features are a representation of the spectral energy in the frequency range of each one of the twelve semitones. A good review of PCP, as well as other chroma-based features, can be found in [7].

Transpose a music for another key or main tonality is a commonly used practice to adapt the song to a singer or to make it heavier or lighter. Key invariance tries to reduce the effects of these changes in music retrieval systems that use tonal information. A simple and effective method to provide robustness to key changes is the optimal transposition index (OTI) [11]. As a first step, this method computes a vector of harmonic pitch class profiles (HPCP) for each song, which is the normalized mean value of the energy in each semitone [5]. When comparing two songs A and B, the method fixes the HPCP of A. For each shift of the HPCP of B, it measures the inner product between the two vectors. The shift that maximizes this product is chosen and the song B is transposed using such a shift value.

Tempo invariance is the robustness to changes between different versions caused by faster or slower performances. One way of achieving tempo invariance is by modifying the feature extraction phase to extract one or more feature vectors per beat [4], instead of a time-based window. Another possibility is the use of specific feature sets, such as chroma energy normalized statistics (CENS) [8]. These features use a second stage in the chroma vector estimation that provides a higher robustness to local tempo variations.

Structure invariance is the robustness to deviations in long-term structure, such as repeated chorus or skipped verses. This invariance may be achieved by several different approaches, such as dynamic programming-based algorithms [3], sequential windowing [13] or by summarizing the music pieces into their most repeated parts [9].

The last step of a querying and retrieval system is the similarity computation between the query and reference data by means of a distance calculation. The most common approaches for this task are dynamic programming based algorithms that try to find an optimal alignment of feature vectors. A well-known example of this approach is the Dynamic Time Warping (DTW) distance function.

In this paper, we present an approach that adds a training phase to this process. This step seeks to find the most significant excerpt of each song in the set S (training set). These small segments are used in a comparison with the query song q. Our method is inspired by the idea of time series shapelets, presented next.

3. SHAPELETS

Time series shapelets is a well-known approach for time series classification [16]. In classification, there exists a training set of labeled instances, S. A typical learning system uses the information in S to create a classification model, in a step known as training phase. When a new instance is available, the classification algorithm associates it to one of the classes in S.

A time series shapelet may be informally defined as the subsequence that is the most representative of a class. The original algorithm of [16] finds a set of shapelets and use them to construct a decision tree classification model. The training phase of such learning system consists of three basic steps:

- **Generate candidates:** this step consists in extracting all subsequences from each training time series;

- **Candidates' quality assessment:** this step assesses the quality of each subsequence candidate considering its class separability;

- **Classification model generation:** this step induces a decision tree. The decision in each node is based on the distance between the query time series and a shapelet associated to that node.

In the first step, the length of the candidates is an intrinsic parameter of the candidates generation. The original algorithm limits the search to a range between a minimum (min_{len}) and maximum (max_{len}) length. All the subsequences with length between min_{len} and max_{len} are stored as candidates.

Given a candidate s, we need to measure the distance between s and a whole time series x. Notice that a direct comparison between them is not always possible since s and x can have very different lengths. Consider l as the candidate's length. The $distance(s, x)$ is defined as the smallest Euclidean distance between the candidate s and each subsequence of x with l observations.

The next steps of the shapelet algorithm are directly related to the classification task. Since this is not our focus, we suppress details of the algorithm from this point.

The general idea of classifying time series by shapelets is to use the distances between candidates and training time series to construct a classification model. First, the algorithm estimates the best information gain (IG) that can be obtained by each candidate. This is made by grouping the training examples that are closer – according a distance threshold – from the training examples that are more distant from the candidate. The best value for the threshold – called *best split point* – is defined by assessing the separation obtained by different values.

Finally, the algorithm uses the IG to create a decision tree. A decision node uses the information of the best shapelet candidate. In order to decide the class of a test example, we measure the distance between the query and the shapelet. If the distance is smaller or equal to the split point, its class is the one associated with the shapelet. Otherwise, the query is labeled as belonging to the other class.

For details on how to find the optimal split point and the decision tree's construction, we refer the reader to [16].

4. OUR PROPOSAL: MUSIC SHAPELETS

In this paper, we propose to adapt the idea of shapelets for a fast content-based music retrieval, more specifically for cover songs identification. Our adaptations are detailed in the next sections.

4.1 Windowing

The original approach to finding subsequence candidates uses sliding windows with different lengths. These lengths are the enumeration of all values in a range provided by the user. The sliding window swipes across the entire time series and such a process is performed for each example in the training set. We found this process to be very time consuming, accounting for most of the time spent in the training phase.

We note that music datasets are typically higher-dimensional than most time series benchmark datasets, in both number of objects as well as number of observations. Thus, we use a reduced set of specific values as window length instead of an interval of values. We empirically noted that it is possible to find good candidates without enumerating all the lengths in a given range.

In addition, the original approach uses a sliding window that starts at every single observation of a time series. We slightly modified it so that the sliding windows skip a certain amount of observations proportional to the window length. This windowing technique with partial overlapping is common in audio analysis.

4.2 Dynamic Time Warping

Shapelets use Euclidean distance (ED) as the similarity measure to compare a shapelet and a whole time series. However, ED is sensitive to local distortions in the time axis, called *warping*. Warping invariance is usually beneficial for music similarity due to the differences in tempo or rhythm that can occur when a song is played live or by different artists.

In order to investigate this assumption, we evaluate the use of ED and Dynamic Time Warping (DTW) to compare shapelets extracted from music data. There is an obvious problem with the use of DTW, related to its complexity. While ED is linear on the number of observations, DTW has a quadratic complexity. Nevertheless, there is a plethora of methods that can be used so that we may accelerate the calculation of the distance between a shapelet and a whole music [10].

4.3 Distance–based Shapelet Quality

Shapelets were originally proposed for time series classification. In cover song identification we are interested in providing a ranking of recordings considering the similarity to a query. Therefore, IG is not the best choice to measure the candidates' quality.

IG in shapelet context finds the best split points and candidates according to class separability. However, music retrieval problems typically have a large number of classes (each class representing a single song) with few examples (different recordings of a certain song), hindering the analysis of class separability.

For this reason, we propose and evaluate the substitution of the IG by a distance-based criterion. We consider that a good candidate has a small distance value to all the versions of the related song and a high distance value to any recording of another song. Thus, we propose the criterion DistDiff, defined in Equation 1.

$$DistDiff(s) = \min_{i=1..n}(distance(s, OtherClass(i))) - \frac{1}{m}\sum_{i=1}^{m} distance(s, SameClass(i)) \quad (1)$$

where s is a candidate for shapelet, $SameClass$ is the set of m versions of the song from were the candidate come from, $OtherClass$ is the set of n recordings that does not represent a version of the same composition than the origin of s and $distance(s, Set(i))$ is the distance between the

candidate and the i-th recording in Set ($SameClass$ or $OtherClass$).

Clearly, we are interested in candidates that provide a high value to the first term and a small value to the second. So, as higher the value of DistDiff, higher the quality of the candidate. In case of draw, we use the minimum average rank of the versions of the song related to s as tie breaking. In other words, if two candidates have the same value of DistDiff, the best candidate is the one that provides the best average ranking positions for the versions of the song from where s comes from.

4.4 Similarity

Since the technique of time series shapelets is interested in class separability, it stores at most one shapelet per class. On the other hand, in our problem we are interested in all examples of each "class label". So, we store one shapelet per recording in the training set, instead one for each composition.

The final step, the querying and retrieval itself, is made in two simple steps. First, our method measures the distance between the query music and each of the shapelets found in the training phase. Finally, the ranking is given by sorting these distances in ascending order.

4.5 Triplets

In a real scenario where the task of music retrieval will be performed, it is highly probable that a specific song has one to three authorized versions such as the original recording in a studio, an acoustic and a live version. Obviously, there are exceptions such as remix and many versions of live performances. Thus, when we extract shapelets from these songs in a conventional way, we have only a few instances for each class in the training set. This may hamper the candidate's quality calculation.

In addition, only a small segment of a song can be uninformative. This fact has been observed in other application domains. For instance, [14] uses features from the beginning, the middle and the end of each recording to perform the genre recognition task.

For these reasons, we also evaluated the idea of representing each recording as three shapelets. Figure 2 illustrate this procedure. The first step of this procedure divides the feature vector into three parts of the same length. After that, we find the most representative subsequence of each segment. Finally, during the retrieval phase, we use the mean distance from a query recording to each of the three shapelets. We will refer to these triple of shapelets as *triplets*.

5. EXPERIMENTAL EVALUATION

In this section, we present the datasets used in our evaluation and the experimental results. We conclude this section discussing the advantages of our method in terms of time complexity in the retrieval phase.

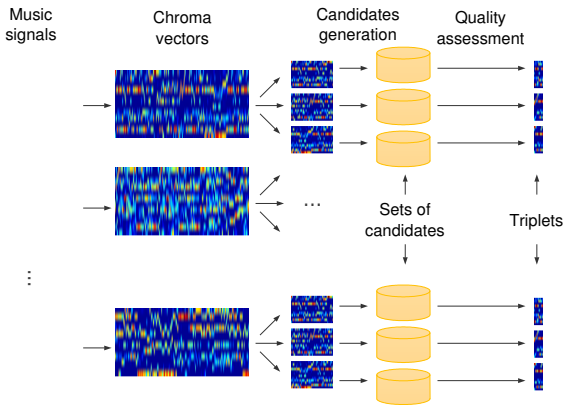

Figure 2. General procedure to generate triplets

5.1 Datasets

We evaluate our proposal in two datasets with different music styles. The first dataset is composed by classical music while the second contains popular songs.

The dataset *123 Classical* was originally used in [1]. This dataset has 123 different recordings concerning 19 compositions from Classical (between 1730 and 1820) and Romantic (between 1780 and 1910) ages. From the 123 recordings, 67 were performed by orchestras and the remaining 56 were played in piano.

We also collected popular songs from videos of *YouTube* and built a dataset named *YouTube Covers*. We made the *YouTube Covers* dataset freely available in our website [15] for interested researchers. This dataset was built with the goal of evaluating our proposal in a more diverse data since the covers songs in the *123 Classical* dataset in general faithfully resembling their original versions.

The *YouTube Covers* dataset has 50 original songs from different music genres such as *reggae, jazz, rock* and *pop music* accompanied of cover versions. In our experiments, we divide this dataset in training and test data. The training data have the original recording in studio and a live version for each music. In the test data, each music has 5 different cover versions that include versions of different music styles, acoustic versions, live performances of established artists, fan videos, etc. Thus, this dataset have a total of 350 songs (100 examples for training and 250 for test). A complete description of *YouTube Covers* dataset is available in our website.

As the *123 Classical* dataset doesn't have a natural division in training/test sets and has a reduced amount of data, we conducted our experimental evaluation in this dataset using stratified random sampling with $1/3$ of data to training and the remaining for test. With this procedure, the number of examples per class in the training phase varies from 1 to 5.

5.2 Evaluation Scenarios

In this paper, we consider two different scenarios to evaluate our method: i) *test set as query* and ii) *training set as*

query. In both, the first stage finds shapelets in the training partition.

In the first scenario, we perform a query when a new song arrives. This setting simulates the scenario in which we would like to know if the (test) query is a cover of some previously labeled song. In other words, we use the unlabeled recordings to find similar labeled ones.

In the second scenario, we simulate the scenario in which the author of one of the training songs wants to know if there are uncertified versions of his/her music in the repository. Thus, we should use his/her original recording as query. Therefore, the training instances are used as queries and we use the shapelets to return unlabeled songs that are potentially covers.

5.3 Experimental Setup

In order to evaluate the proposed method, we compare its results against two competitors. The first one is the DTW alignment of the feature vector representing the whole music. The second one uses a music summarization algorithm to find significant segments of the recordings. For this, we use a method that considers that the most significant excerpts of music are those that are most repeated [2]. After finding such excerpts, the similarity search occurs as proposed in this paper.

As feature sets, we used the chroma energy normalized statistics (CENS), as well as chroma extracted together the beat estimating. In general, CENS results are slightly better. Thus, we focus our evaluation using this feature. To extract the CENS, we used the Matlab implementation provided by the Chroma Toolbox [7] with the default parameters settings.

We used the optimal transposition index (OTI) technique to improve robustness for key variances. Shapelets are not used to decide the shift to provide such an invariance. This is done by using the harmonic pitch class profiles (HPCP) of the complete chroma vector.

Our proposal have two parameters related to the windowing: i) window length and ii) overlapping proportion of consecutive windows. For the first parameter, we use the values 25, 50 and 75 for shapelets and 25 for triplets. For the second parameter, we use $2/3$ of the window length as overlapping proportion

To provide an intuition to the reader about the first parameter. The mean length of the chroma feature vectors in the datasets *123 Classical* and *YouTube Covers* are 215 and 527, respectively. Therefore, a window length of 25 represents approximately 11% and 5%, respectively, of the average length of the recordings in these datasets.

5.4 Evaluation Measures

In order to assess the quality of our proposal, we used three evaluation measures adopted by MIREX [2] for the cover song identification task. Such measures take into account the position of the relevant songs in the estimated ranking of similarity.

[2] http://www.music-ir.org/mirex/wiki/2015:Audio_Cover_Song_Identification

Given a set of n query songs, a retrieval method returns a rank r_i ($i = 1, 2, \ldots, n$) for each of them. The function $\Omega(r_{i,j})$ returns the value 1 if the $j-th$-ranked song obtained for the $i-th$ query is a relevant song or 0 otherwise. In the context of this work, a relevant song is a cover version of the query recording.

The first evaluation measure represents the mean number of relevant songs retrieved among the top ten positions of the ranking (MNTop10). Formally, the MNTop10 is defined according to Equation 2.

$$MNTop10 = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{10}\Omega(r_{i,j}) \quad (2)$$

The mean average precision (MAP) is the mean value of the average precision (AP) for each query song. The AP is defined in Equation 3.

$$AP(r_i) = \frac{1}{n}\sum_{j=1}^{n}\left[\Omega(r_{i,j})\left(\frac{1}{j}\sum_{k=1}^{j}\Omega(r_{i,k})\right)\right] \quad (3)$$

Finally, we also use the mean rank of first correctly identified cover (MFRank). In other words, this measure estimates, on average, the number of songs we need to examine in order to find a relevant one. The MFRank is defined by Equation 4.

$$MFRank = \frac{1}{n}\sum_{i=1}^{n} fp(r_i) \quad (4)$$

where $fp(r_i)$ is a function that returns the first occurrence of a relevant object in the ranking r_i.

For the first two measures, larger values represent better performance. For the last one the smaller values are indicative of superiority.

5.5 Results

In the Section 4, we proposed several adaptations to the original shapelets approach to the music retrieval setting. Unfortunately, due to lack of space, we are not able to show detailed results for all combinations of these techniques. In total, we have 16 different combinations of techniques. All those results are available on the website created for this work [15].

In this section, we present a subset of the results according to the following criteria:

- **OTI**. We show all results with OTI as key invariance method. For the dataset *YouTube Covers*, the use of OTI led to significant improvements. For the *123 Classical* dataset, OTI performed quite similarly to the same method without OTI. This may occur because the problem of key variations is more evident in the pop music. We notice we used the simplest version of OTI, that assesses just one tonal shift.

- **Shapelet evaluation**. We evaluate all results with DistDiff. In most cases, information gain performed

worst than DistDiff. Even more, there are cases where the use of IG causes a significant performance deterioration. For example, when using a single shapelet per recording on *YouTube Covers*, the method using information gain achieved MNTop10 = 0.75, MAP = 25.29% and MFRank = 17.52. By changing this measure by the DistDiff criterion, proposed in this paper, the results become MNTop10 = 1.22, MAP = 47.14% and MFRank = 9.72.

- **Triplet**. We show the results using triplets. In general the use of a single shapelet to describe the training songs did not outperform the use of triplets. Although obtain an improvement in isolated cases, the differences are small in these cases.

Therefore, we will fix our analysis to the methods that use OTI and triplets evaluated by DistDiff criterion. The last remaining decision concerns the use of Euclidean or DTW distances. We show the results obtained with both.

Table 1 shows the results obtained on *123 Classical* dataset and Table 2 shows the results obtained on *YouTube Covers* dataset.

Table 1. Results achieved on the dataset *123 Classical*

Scenario 1 - Test set as query			
	MNTop10	MAP (%)	MFRank
DTW	2.34	97.24	1.12
Summarization	2.27	93.46	**1.00**
Triplets-DTW	**2.39**	97.24	1.02
Triplets-ED	2.38	**98.05**	**1.00**
Scenario 2 - Training set as query			
	MNTop10	MAP (%)	MFRank
DTW	4.73	98.92	**1.00**
Summarization	4.44	91.52	1.02
Triplets-DTW	**4.78**	**99.41**	**1.00**
Triplets-ED	4.71	97.92	**1.00**

Table 2. Results achieved on the dataset *YouTube Covers*

Scenario 1 - Test set as query			
	MNTop10	MAP (%)	MFRank
DTW	1.14	42.49	11.69
Summarization	0.85	32.11	13.82
Triplets-DTW	**1.29**	45.55	**8.45**
Triplets-ED	1.26	**47.80**	8.49
Scenario 2 - Training set as query			
	MNTop10	MAP (%)	MFRank
DTW	2.11	39.19	6.58
Summarization	1.66	29.20	14.46
Triplets-DTW	2.82	52.87	**4.65**
Triplets-ED	**2.87**	**54.95**	5.18

5.6 Discussion

The results show that triplets outperformed similarity estimation by using music summarization and achieved equal or better results than the DTW matching of the whole feature vector.

More importantly, we notice that the querying using shapelets is significantly more efficient than the matching between the whole songs. Although our method requires a training phase that is absent in similarity search with DTW, such a phase is performed only once.

Let l and m be the length of feature vectors of the query and the labeled songs. The complexity to find an alignment based on dynamic programming, such as DTW, is $\mathcal{O}(lm)$. Now, let s be the size of each shapelet of the training song. The complexity to calculate the shapelet-based Euclidean distance between the query and the original song is $\mathcal{O}(ls)$, with $s \ll m$.

Table 3 shows the time in seconds to perform the retrieval step using Triplets-ED and DTW matching the entire feature vectors.

Table 3. Total time (in seconds) to calculate the distance between all the queries (test set) and the training set by using DTW and Triplets-ED

	Dataset	
	123 Classical	*YouTube Covers*
DTW	2,294	14,124
Triplets-ED	148	928

The result of this experiment shows that our method is about 15 times faster to retrieve music by similarity. We argue that our method may be further faster with the use of techniques to speed-up the similarity search – to find the best match between the shapelet and the whole feature vector.

The identification rates were similar for both triplets approaches, alternating the best results between them. Although the time spent to calculate Triplets-DTW is potentially lower than the obtained by a straightforward implementation of Euclidean distance [10], the time spent by our simple implementation is similar to the DTW alignment of the whole feature vector.

6. CONCLUSION

In this paper, we propose a novel technique to content-based music retrieval. Our method is naturally invariant to structure and open to aggregate invariance to key and tempo by the choice of appropriate methods, such as OTI and CENS as feature vector.

We evaluated our method in a cover song recognition scenario. We achieved better results than the widely applied approach of DTW alignment and a similar approach based on a well-known summarization algorithm. Our method is also more than one order of magnitude faster than these methods.

There are several possible extensions for this work. For instance, we can extend our idea to a shapelet-transform [6]. The evaluated scenario also suggests research on incremental learning of shapelets, the retrieval considering that novel songs may arrive, among other tasks. Finally, we intend to investigate how to improve the time cost of DTW similarity search in order to make the time of Triplets-DTW be competitive with Triplets-ED.

7. ACKNOWLEDGMENTS

The authors would like to thank FAPESP by the grants #2011/17698-5, #2013/26151-5, and 2015/07628-0 and CNPq by grants 446330/2014-0 and 303083/2013-1.

8. REFERENCES

[1] Juan Pablo Bello. Measuring structural similarity in music. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7):2013–2025, 2011.

[2] Matthew L. Cooper and Jonathan Foote. Automatic music summarization via similarity analysis. In *International Society for Music Information Retrieval Conference*, 2002.

[3] Emanuele Di Buccio, Nicola Montecchio, and Nicola Orio. A scalable cover identification engine. In *International Conference on Multimedia*, pages 1143–1146, 2010.

[4] Daniel P. W. 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 1429–1432, 2007.

[5] Emilia Gómez and Perfecto Herrera. Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modelling strategies. In *International Society for Music Information Retrieval Conference*, pages 92–95, 2004.

[6] Jason Lines, Luke M. Davis, Jon Hills, and Anthony Bagnall. A shapelet transform for time series classification. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 289–297, 2012.

[7] Meinard Müller and Sebastian Ewert. Chroma Toolbox: MATLAB implementations for extracting variants of chroma-based audio features. In *International Society for Music Information Retrieval Conference*, pages 1–6.

[8] Meinard Müller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *International Society for Music Information Retrieval Conference*, pages 288–295, 2005.

[9] Bee Suan Ong. *Structural Analysis and Segmentation of Music Signals*. PhD thesis, 2007.

[10] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 262–270, 2012.

[11] 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.

[12] Joan Serrà, Emilia Gómez, and Perfecto Herrera. Audio cover song identification and similarity: background, approaches, evaluation, and beyond. In *Advances in Music Information Retrieval*, pages 307–332. Springer, 2010.

[13] Joan Serrà, Xavier Serrà, and Ralph G. Andrzejak. Cross recurrence quantification for cover song identification. *New Journal of Physics*, 11(9):093017, 2009.

[14] Carlos Nascimento Silla Jr, Alessandro Lameiras Koerich, and Celso A. A. Kaestner. The latin music database. In *International Society for Music Information Retrieval Conference*, pages 451–456, 2008.

[15] Diego F. Silva, Vinícius M. A. Souza, and Gustavo E. A. P. A. Batista. Website for this work – `https://sites.google.com/site/ismir2015shapelets/`.

[16] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 947–956, 2009.