IDENTIFYING POLYPHONIC PATTERNS FROM AUDIO RECORDINGS USING MUSIC SEGMENTATION TECHNIQUES

Oriol Nieto and Morwaread M. Farbood
Music and Audio Research Lab
New York University
{oriol, mfarbood}@nyu.edu

ABSTRACT

This paper presents a method for discovering patterns of note collections that repeatedly occur in a piece of music. We assume occurrences of these patterns must appear at least twice across a musical work and that they may contain slight differences in harmony, timbre, or rhythm. We describe an algorithm that makes use of techniques from the music information retrieval task of music segmentation, which exploits repetitive features in order to automatically identify polyphonic musical patterns from audio recordings. The novel algorithm is assessed using the recently published JKU Patterns Development Dataset, and we show how it obtains state-of-the-art results employing the standard evaluation metrics.

1. INTRODUCTION

The task of discovering repetitive musical patterns (of which motives, themes, and repeated sections are all examples) consists of retrieving the most relevant musical ideas that repeat at least once within a specific piece [1, 8]. Besides the relevant role this task plays in musicological studies, especially with regard to intra-opus analysis, it can also yield a better understanding of how composers write and how listeners interpret the underlying structure of music. Computational approaches to this task can dramatically simplify not only the analysis of a specific piece, but of an entire corpus, potentially offering interesting explorations and relations of patterns across works. Other potential applications include the improved navigation across both large music collections and stand-alone pieces, or the development of computer-aided composition tools.

Typically the task of automatically discovering musical patterns uses symbolic representations of music [3]. Methods that assume a monophonic representation have been proposed, and operate on various musical dimensions such as chromatic/diatonic pitch, rhythm, or contour [4, 9, 10]. Other methods focusing on polyphonic music as input have also been presented, mostly using geometric representations in Euclidean space, with a different axis assigned to each musical dimension [6, 11]. Similar techniques have also been explored [7, 11, 12] that attempt to arrive at a compressed representation of an input, multidimensional point set. Other methods using cognitively inspired rules with symbolic representations of music have also been proposed [6, 16]. Working with the score of a musical piece instead of its audio representation can indeed reduce the complexity of the problem, however this also significantly narrows the applicability of the algorithm, since it is not necessarily common to have access to symbolic representations of music, particularly when working with genres such as jazz, rock, or Western popular music.

Methods using audio recordings as input have also been explored. A good recent example is [3], where the authors first estimate the fundamental frequency (F0) from the audio in order to obtain the patterns using a symbolic-based approach. Another one uses a probabilistic approach to matrix factorization in order to learn the different parts of a western popular track in an unsupervised manner [20]. Yet another method uses a compression criterion where the most informative (i.e., repeated) parts of a piece are identified in order to automatically produce a musical "summary" [17].

In this paper, we propose a method using audio recordings as input in an attempt to broaden the applicability of pattern discovery algorithms. We make use of tools that are commonly employed in the music information retrieval task of *music segmentation* combined with a novel score-based greedy algorithm in order to identify the most repeated parts of a given audio signal. Finally, we evaluate the results using the JKU Patterns Development Dataset and the metrics proposed in the Music Information Retrieval Evaluation eXchange (MIREX) [1].

The outline of this paper is as follows: In Section 2 we review a set of music segmentation techniques that will be used in our algorithm; in Section 3 we detail our method to extract musical patterns, including the score-based greedy algorithm; in Section 4 we present the evaluation and the results; and in Section 5 we draw various conclusions and identify areas for future work.

2. MUSIC SEGMENTATION TECHNIQUES

The task of music segmentation is well-established in the music informatics literature (see [18] for a review). Its goal is to automatically identify all the non-overlapping musical segments (or sections) of a given track, such that the concatenation of all of them reconstructs the entire piece. Once these segments are identified, they are labeled based on their similarity (e.g., verse, chorus, coda). Therefore, this task can be divided into two different subproblems: the discovery of the *boundaries* that define all the segments, and the grouping of the segments into different *labels*. In this work we will use tools that focus mainly on the former subproblem.

There is general agreement among researchers that any given boundary is defined by at least one of these three characteristics: repetition, homogeneity, and/or novelty [18]. In our case, we center the discussion on the repetition boundaries, since, as we will see in Section 3, repetition is the defining feature of the musical patterns.

2.1 Extracting Repetitive Boundaries

In this subsection we review a standard technique to extract boundaries characterized by repetition (also known as a sequence approach), from an input audio signal x. For a more detailed explanation, we refer the reader to [13]. The process can be summarized in three different steps:

i The signal x is transformed into a series of feature vectors $C = (\mathbf{c}_1, ..., \mathbf{c}_N)$ that divide x into N frames and capture specific frame-level characteristics of the given signal. In our case, we will only focus on harmonic features, more specifically on chromagrams (or pitch class profiles).

ii C is used in order to obtain a self-similarity matrix (SSM) S, a symmetric matrix such that $S(n, m) = d(\mathbf{c}_n, \mathbf{c}_m), \forall n, m \in [1 : N]$, where d is a distance function (e.g. Euclidean, cosine, Manhattan).

iii The resulting matrix S will contain diagonal paths (or semi-diagonal in case of slight tempo variations) or stripes that will indicate the repetition of a specific part of the audio signal x. These paths can be extracted using greedy algorithms (e.g., as described in [13]). The final boundaries are given by the endpoints of these paths.

An example of an SSM using the Euclidean distance with the identified boundaries is shown in Figure 1. As can be seen, the annotated boundaries are visually associated with the paths of the matrix. The identification of patterns, as opposed to the task of segmentation, allows overlapping patterns and occurrences, so we base our algorithm on greedy methods to extract paths from an SSM.

2.2 Transposition-Invariant Self-Similarity Matrix

It is common to analyze pieces that contain key-transposed repetitions. It is therefore important for an algorithm to be invariant to these these transpositions when identifying

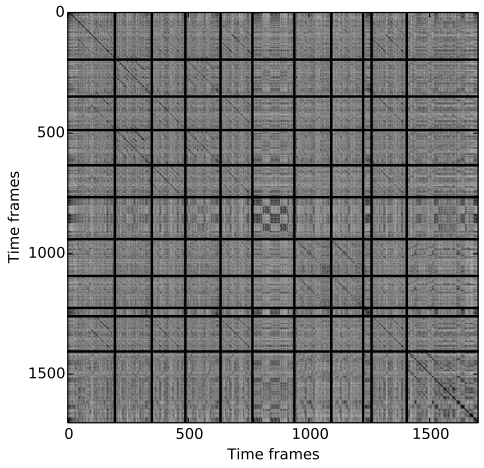

Figure 1. Self-similarity matrix for Chopin's Op. 24 No. 4, with annotated boundaries as vertical and horizontal lines.

repetitions. One effective method for solving this problem [14] involves a technique that can be described in two steps: (1) compute 12 different SSMs from harmonic representations (e.g. chromagrams), each corresponding to a transposition of the 12 pitches of the Western chromatic scale, and 2) obtain the transposition-invariant SSM by keeping the minimum distance across the 12 matrices for all the $N \times N$ distances in the output matrix. Formally:

$$\mathcal{S}(n, m) = \min_{k \in [0:11]}\{S_k(n, m)\}, \forall n, m \in [1 : N] \quad (1)$$

where $\mathcal{S}$ is the transposition-invariant SSM, and S_k is the k-th transposition of the matrix S.

3. IDENTIFYING MUSICAL PATTERNS

The discovery of patterns and their various occurrences involves retrieving actual note collections (which may nest and/or overlap), and so this task can be seen as more complex than structural segmentation, which involves labeling a single, temporal partition of an audio signal. We define a repeating musical pattern to be a short idea that is repeated at least once across the entire piece, even though this repetition may be transposed or contain various time shifts. Therefore, each pattern is associated with a set of occurrences that will not necessarily be exact. The patterns and their occurrences may overlap with each other, and this is perfectly acceptable in the context of pattern discovery. An optimal algorithm for this task would (1) find all the patterns contained in a piece and (2) identify all the occurrences across the piece for each pattern found. In this section we describe our algorithm, which uses audio recordings as input and finds polyphonic patterns as well as a list of all the discovered occurrences for each of the patterns. A block-diagram of the entire process is depicted in Figure 2.

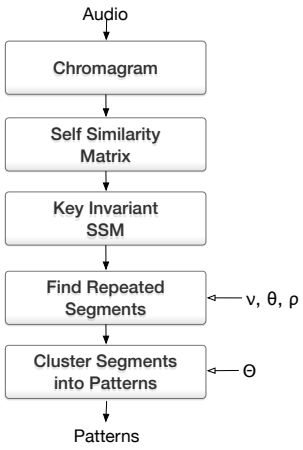

Figure 2. Block diagram of the proposed algorithm.

3.1 Rhythmic-Synchronous Harmonic Feature Extraction

Given a one-channel audio signal x sampled at 11025 Hz representing a piece of music, we compute the spectrogram using a Blackman window of $N_w = 290$ milliseconds, with a hop size of $N_w/2$. We then compute a constant-Q transform from the spectrogram starting at 55 Hz (corresponding to the note A1 in standard tuning) comprising four octaves. Finally, we collapse each of the 12 pitches of the western scale into a single octave to obtain a chromagram, a matrix of $12 \times N$, which is commonly used to represent harmonic features [18]. We normalize the chromagram such that the maximum energy for a given time frame is 1. In this harmonic representation we can no longer differentiate between octaves, but its compactness and the energy of each pitch class will become convenient when identifying harmonic repetitions within a piece.

We then use a beat tracker [5] in order to average the time frames into rhythmic frames. Instead of using the traditional beat-synchronous approach, which is typically employed in a segmentation task, we divide each beat duration by 4 and aggregate accordingly, thus having $N = 4B$ time frames, where B is the number of beats detected in the piece. The motivation behind this is that patterns may not start at the beat level, as opposed to the case for long sections. Furthermore, adding a finer level of granularity (i.e., analyzing the piece at a sixteenth-note level instead of every fourth note or at the beat level) should yield better results.

3.2 Finding Repeated Segments

We make use of the transposition-invariant SSM $\mathcal{S}$ described in Section 2.2, computed from the chromagram of a given audio signal using the Euclidean distance, in order to identify repeated segments. As opposed to the task of segmentation, the goal here is to find *all* possible repeated segments in $\mathcal{S}$, independent of how short they are or the amount of overlap present. The other major difference is that we do not aim to find all of the segments of the piece, but rather identify all of the repeated ones. Repeated segments appear in $\mathcal{S}$ as diagonal "stripes", also known as paths. If the beat-tracker results in no errors (or if the piece contains no tempo variations), these stripes will be perfectly diagonal.

3.2.1 Quantifying Paths with a Score

We propose a score-based greedy algorithm to efficiently identify the most prominent paths in $\mathcal{S}$. Starting from $\mathcal{S} \in \mathbb{R}^{N \times N}$, we set half of its diagonals to zero, including the main one, due to its symmetrical properties, resulting in $\hat{\mathcal{S}}$, s.t. $\hat{\mathcal{S}}(n,m) = 0$ if $n \leq m$ and $\hat{\mathcal{S}}(n,m) = \mathcal{S}(n,m)$ if $n > m, \forall n, m \in [1:N]$. We then compute a score function σ for each possible path in all the non-zero diagonals of $\hat{\mathcal{S}}$, resulting in a search space of $N(N-1)/2$ possible positions in which paths can start.

Before introducing the score function σ, we define a trace function given a square matrix $X \in \mathbb{R}^{N_x \times N_x}$ with an offset parameter ω:

$$\text{tr}(X, \omega) = \sum_{i=1}^{N_x - \omega} X(i, i+\omega), \omega \in \mathbb{Z} \tag{2}$$

As can be seen from this equation, when $\omega = 0$ we have the standard trace function definition.

The score function σ uses various traces of the matrix that comprises a possible path in order to quantify the degree of repetition of the path. If a possible path starts at indices n, m and has a duration of M time frames, then the matrix that the path defines is $P \in \mathbb{R}^{M \times M}$, s.t. $P(i,j) = \hat{\mathcal{S}}(n+i-1, m+j-1), \forall i, j \in [1:M]$. We now can define the score σ as the sum of the closest traces to the diagonal of P (i.e., those with a small ω) and subtract the traces that are farther apart from the diagonal (i.e., where ω is greater). We then normalize in order to obtain a score independent from the duration M of the possible path:

$$\sigma(\rho) = \frac{\left(\sum_{\omega=-(\rho-1)}^{\rho-1} \text{tr}(P, \omega)\right) - \text{tr}(P, \pm\rho)}{M + \sum_{i=1}^{\rho-1} 2(M-i)} \tag{3}$$

where $\rho \in \mathbb{N}$ is the maximum offset to be taken into account when computing the traces of P. The greater the ρ, the greater the σ for segments that contain substantial energy around their main diagonal (e.g., paths that contain significant rhythmic variations), even though the precision decreases as ρ increases.

Examples for various $\sigma(\rho)$ can be seen in Figure 3. For a perfectly clean path (left), we see that $\rho = 1$ gives the maximum score of 1. However, the score decreases as ρ increases, since there is zero energy in the diagonals right next to the main diagonal. On the other hand, for matrices extracted from audio signals (middle and right), we can see that the scores $\sigma(1)$ are low, indicating that the diagonals next to the main diagonal contain amounts of energy similar to the main diagonal. However, when $\rho > 1$, the score is substantially different from a matrix with a path (middle) and a matrix without one (right).

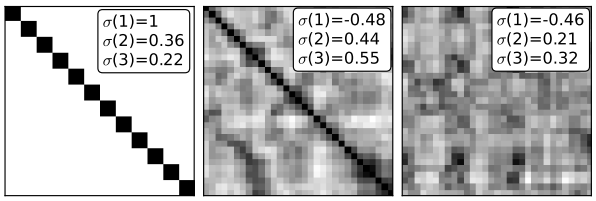

Figure 3. Three examples showing the behavior of the path score $\sigma(\rho)$. The one on the left shows a synthetic example of a perfect path. The one in the middle contains a real example of a path in which there is some noise around the diagonal of the matrix. In the example on the right, a matrix with no path is shown.

3.2.2 Applying the Score

For all $N(N-1)/2$ positions in which paths can potentially start in $\hat{\mathcal{S}}$, we want to extract the most prominent ones (i.e., the ones that have a high σ). At the same time, we want to extract the paths from beginning to end in the most accurate way possible. The algorithm that we propose assigns a certain σ to an initial possible path $\hat{z}$ of a minimum length of ν time frames, which reduces the search space to $(N-\nu+1)(N-\nu)/2$. If the score σ is greater than a certain threshold θ, we increase the possible path by one time frame, and recompute σ until $\sigma \leq \theta$. By then, we can store the path $\hat{z}$ as a segment in the set of segments $\mathcal{Z}$. In order to avoid incorrectly identifying possible paths that are too close to the found path, we zero out the found path from $\hat{\mathcal{S}}$, including all the ρ closest diagonals, and keep looking, starting from the end of the recently found path.

The pseudocode for this process can be seen in Algorithm 1, where $|\mathbf{x}|$ returns the length of the path $\mathbf{x}$, $\{x\}$ returns the path in which all elements equal x, the function ComputeScore computes the $\sigma(\rho)$ as described in Section 3.2.1, OutOfBounds($\mathbf{x}, X$) checks whether the path $\mathbf{x}$ is out of bounds with respect to X, IncreasePath($\mathbf{x}$) increases the path $\mathbf{x}$ by one (analogously as DecreasePath), and ZeroOutPath($X, \mathbf{x}, \rho$) assigns zeros to the path $\mathbf{x}$ found in X, including all the closest ρ diagonals.

Algorithm 1 Find Repeated Segments

Require: $\hat{\mathcal{S}}, \rho, \theta, \nu$
Ensure: $\mathcal{Z} = \{z_1, \ldots, z_k\}$
  for $\hat{z} \in \hat{\mathcal{S}} \wedge |\hat{z}| = \nu \wedge \hat{z} \neq \{0\}$ **do**
    $b \leftarrow$ False
    $\sigma \leftarrow$ ComputeScore($\hat{z}, \rho$)
    while $\sigma > \theta \wedge \neg$OutOfBounds($\hat{z}, \hat{\mathcal{S}}$) **do**
      $b \leftarrow$ True
      $\hat{z} \leftarrow$ IncreasePath($\hat{z}$)
      $\sigma \leftarrow$ ComputeScore($\hat{z}, \rho$)
    end while
    if b **then**
      $\mathcal{Z}$.add(DecreasePath($\hat{z}$))
      ZeroOutPath($\hat{\mathcal{S}}, \hat{z}, \rho$)
    end if
  end for
  return $\mathcal{Z}$

An example of the paths found by the algorithm is shown in Figure 4. Parts of some segments are repeated as standalone segments (i.e., segments within segments), therefore allowing overlap across patterns as expected in this task. Observe how some of the segments repeat almost exactly across the piece—there is a set of patterns at the top of the matrix that appears to repeat at least three times. The next step of the algorithm is to cluster these segments together so that they represent a single pattern with various occurrences.

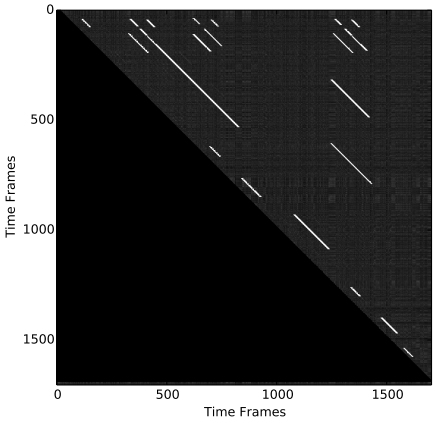

Figure 4. Paths found (marked in white) using the proposed algorithm for Chopin's Op. 24 No. 4., with $\theta = 0.33, \rho = 2$.

3.3 Clustering the Segments

Each segment $z \in \mathcal{Z}$, defined by the two indices in which it starts (s_i, s_j) and ends (e_i, e_j) in $\hat{\mathcal{S}}$, contains two occurrences of a pattern: the one that starts in s_i and ends in e_i and the one that occurs between the time indices s_j and e_j. In order to cluster the repeated occurrences of a single pattern, we find an occurrence for each segment $z \in \mathcal{Z}$ if one of the other segments in $\mathcal{Z}$ starts and ends in similar locations with respect to the second dimension of $\hat{\mathcal{S}}$. Note that we set to zero the bottom left triangle of the matrix as explained in Section 3.2.1, so we cannot use the first dimension to cluster the occurrences. Formally, a segment $\hat{z}$ is an occurrence of z if

$$(s_j^z - \Theta \leq s_j^{\hat{z}} \leq s_j^z + \Theta) \wedge (e_j^z - \Theta \leq e_j^{\hat{z}} \leq e_j^z + \Theta) \quad (4)$$

where s_j^z represents the starting point of the segment z in the second dimension of $\hat{\mathcal{S}}$ and analogously e_j^z is the ending point, and Θ is a tolerance parameter.

3.4 Final Output

At this point, we have a set of patterns with their respective occurrences represented by their starting and ending timeframe indices. Even though the algorithm is not able to distinguish the different musical lines within the patterns, we can use the annotated score to output the exact notes that occur during the identified time indices, as suggested in the MIREX task [1]. If no score is provided, only the time

points will be presented. In order to overcome this limitation in future work, the audio should be source-separated to identify the different lines and perform an F0 estimation to correctly identify the exact melody that defines the pattern (and not just the time points at which it occurs). Progress toward this goal has been presented in [2].

3.5 Time Complexity Analysis

Once the rhythm-synchronous chromagram is computed, the process of calculating the transposition-invariant SSM is $O(13N^2) = O(N^2)$, where N is the number of time frames of the chromagram. The procedure to compute the score given a path has a time complexity of $O(2\rho M) = O(\rho M)$, where ρ is the required parameter for the computation of the score, and M is the length of the path from which to compute the score. The total process of identifying segments is $O\left(\frac{(N-\nu+1)(N-\nu)}{2}\rho M\right) = O((N-\nu)^2\rho M)$, where ν is the minimum number of time frames that a pattern can have. Asymptotically, we can neglect the clustering of the segments, since the length of $\mathcal{Z}$ will be much less than N. Therefore, the total time complexity of the proposed algorithm is $O(N^2 + (N-\nu)^2\rho M)$.

4. EVALUATION

We use the JKU Patterns Development Dataset [1] to evaluate our algorithm. This dataset is comprised of five classical pieces annotated by various musicologists and researchers [1]. This dataset is the public subset of the one employed to evaluate the Pattern Discovery task at MIREX, using the metrics described below.

4.1 Metrics

Two main aspects of this task are evaluated: the patterns discovered and the occurrences of the identified patterns across the piece. Collins and Meredith proposed metrics to quantify these two aspects, which are detailed in [1]; all of these metrics use the standard F_1 accuracy score, defined as $F_1 = 2PR/(P+R)$, where P is precision (such that $P = 1$ if all the estimated elements are correct), and $R = 1$ is recall (such that $R = 1$ if all the annotated elements are estimated).

Establishment $\mathbf{F}_1$ Score (F_{est}): Determines how the annotated patterns are *established* by the estimated output. This measure returns a score of 1 if at least one occurrence of each pattern is discovered by the algorithm to be evaluated.

Occurrence $\mathbf{F}_1$ Score (F_{o}): For all the patterns found, we want to estimate the ability of the algorithm to capture all of the occurrences of these patterns within the piece independently of how many different patterns the algorithm has identified. Therefore, this score would be 1 if the algorithm has only found one pattern with all the correct occurrences. A parameter c controls when a pattern is considered to have been discovered, and therefore whether it counts toward the occurrence scores. The higher the c, the smaller the tolerance. In this evaluation, as in MIREX, we use $c = .75$ and $c = .5$.

Three-Layer $\mathbf{F}_1$ Score (F_3): This measure combines both the patterns established and the quality of their occurrences into a single score. It is computed using a three-step process that yields a score of 1 if a correct pattern has been found and all its occurrences have been correctly identified.

4.2 Results

The results of the proposed algorithm, computed using the open source evaluation package `mir_eval` [19], are shown in Table 1, averaged for the entire JKU Dataset, along with an earlier version of our algorithm submitted to MIREX [15], another recent algorithm called SIARCT-CFP [2] that is assessed using both audio and symbolic representations as input in [3], and "COSIATEC Segment," a method that only uses symbolic inputs [12]. We use this latter method for comparison because it is the only symbolic method in which we have access to all of the resulting metrics, and SIARCT-CFP since it is the most recent method that uses audio as input. The parameter values use to compute these results, $\nu = 8, \theta = 0.33, \rho = 2$, and $\Theta = 4$, were found empirically. We can see how our algorithm is better than [15] in all the metrics except running time; it also finds more correct patterns than [3] (the current state-of-the-art when using audio as input).

Our algorithm obtains state-of-the-art results when extracting patterns from audio, obtaining an F_{est} of 49.80%. This is better than the symbolic version of [2] and almost as good as the algorithm described in [12]. The fact that our results are superior or comparable to the two other algorithms using symbolic representations indicates the potential of our method.

When evaluating the occurrences of the patterns, we see that our algorithm is still better than [15], but worse than [2] (at least for $c = .5$, which is the only reported result). Nevertheless, the numbers are much lower than [12]. In this case, working with symbolic representations (or estimating the F0 in order to apply a symbolic algorithm as in [2]) yields significantly better results. It is interesting to note that when the tolerance increases (i.e. $c = .5$), our results improve as opposed to the other algorithms. This might be due to the fact that some of the occurrences found in the SSM were actually very similar (therefore they were found in the matrix) but were slightly different in the annotated dataset. A good example of this would be an occurrence that contains only one melodic voice. Our algorithm only finds points in time in which an occurrence might be included, it does not perform any type of source separation in order to identify the different voices. If the tolerance decreases sufficiently, a polyphonic occurrence would be accepted as similar to a monophonic one corresponding to the same points in time.

Our three layer score (F_3) is the best result when using audio recordings, with an F-measure of 31.74% (unfortunately this metric was not reported in [2]). This metric aims to evaluate the quality of the algorithm with a single

[1] https://dl.dropbox.com/u/11997856/JKU/JKUPDD-Aug2013.zip

Alg	P_{est}	R_{est}	F_{est}	$P_{\text{o}(.75)}$	$R_{\text{o}(.75)}$	$F_{\text{o}(.75)}$	P_3	R_3	F_3	$P_{\text{o}(.5)}$	$R_{\text{o}(.5)}$	$F_{\text{o}(.5)}$	Time (s)
Proposed	54.96	51.73	**49.80**	37.58	27.61	**31.79**	35.12	35.28	**32.01**	45.17	34.98	38.73	454
[3]	14.9	60.9	23.94	–	–	–	–	–	–	62.9	51.9	**56.87**	–
[15]	40.83	46.43	41.43	32.08	21.24	24.87	30.43	31.92	28.23	26.60	20.94	23.18	**196**
[3]	21.5	78.0	33.7	–	–	–	–	–	–	78.3	74.7	76.5	–
[12]	43.60	63.80	50.20	65.40	76.40	68.40	40.40	54.40	44.20	57.00	71.60	63.20	7297

Table 1. Results of various algorithms using the JKU Patterns Development Dataset, averaged across pieces. The top rows of the table contain algorithms that use deadpan audio as input. The bottom rows correspond to algorithms that use symbolic representations as input.

score, including both pattern establishment and occurrence retrieval. Our results are still far from perfect (32.01%), but when compared to an algorithm that uses symbolic representations [12] (44.21%), it appears our results are not far from the state-of-the-art for symbolic representations.

Finally, our algorithm takes more than twice as long as [15]. However, our method is more than 16 times faster than [12], indicating it is efficient in terms of computation time. This algorithm is implemented in Python and available for public download. [2]

5. CONCLUSIONS

We presented a method to discover repeating polyphonic patterns using audio recordings as input. The method makes use of various standard techniques typically used for music segmentation. We evaluated our method using the JKU Pattern Development Dataset and showed how it obtains competent results when retrieving all the occurrences of the patterns and state-of-the-art results when finding patterns. When the algorithm is compared to others that use symbolic representations, we see that it is comparable or superior in terms of the correct patterns found. In future work, source separation might be needed to successfully identify patterns that only comprise a subset of the different musical lines.

6. REFERENCES

[1] Tom Collins. Discovery of Repeated Themes & Sections, 2013.

[2] Tom Collins, Andreas Arzt, Sebastian Flossmann, and Gerhard Widmer. SIARCT-CFP: Improving Precision and the Discovery of Inexact Musical Patterns in Point-set Representations. In *Proc. of the 14th International Society for Music Information Retrieval Conference*, pages 549–554, Curitiba, Brazil, 2014.

[3] Tom Collins, B Sebastian, Florian Krebs, and Gerhard Widmer. Bridging the Audio-Symbolic Gap: The Discovery of Repeated Note Content Directly From Polyphonic Music Audio. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*, pages 1–12, London, UK, 2014.

[4] Darrell Conklin and Christina Anagnostopoulou. Representation and Discovery of Multiple Viewpoint Patterns . In *Proc. of the International Computer Music Conference*, pages 479–485, La Havana, Cuba, 2001.

[5] Daniel P. W. Ellis and Graham E. Poliner. Identifying 'Cover Songs' with Chroma Features and Dynamic Programming Beat Tracking. In *Proc. of the 32nd IEEE International Conference on Acoustics Speech and Signal Processing*, pages 1429–1432, Honolulu, HI, USA, 2007.

[2] https://github.com/urinieto/MotivesExtractor

[6] James C Forth. *Cognitively-motivated Geometric Methods of Pattern Discovery and Models of Similarity in Music*. PhD thesis, Glodsmiths, University of London, 2012.

[7] James C. Forth and Geraint A. Wiggins. An Approach for Identifying Salient Repetition in Multidimensional Representations of Polyphonic Music. In Joseph Chan, Jacqueline W. Daykin, and M. Sohel Rahman, editors, *London Algorithmics 2008: Theory and Practice*, pages 44–58. UK: College Publications, 2009.

[8] Berit Janssen, W Bas De Haas, Anja Volk, and Peter Van Kranenburg. Discovering Repeated Patterns in Music: State of Knowledge, Challenges, Perspectives. In *Proc. of the 10th International Symposium on Computer Music Multidisciplinary Research*, Marseille, France, 2013.

[9] Olivier Lartillot. Multi-Dimensional motivic pattern extraction founded on adaptive redundancy filtering. *Journal of New Music Research*, 34(4):375–393, December 2005.

[10] Kjell Lemström. *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, Finland, 2000.

[11] David Meredith. Point-set Algorithms For Pattern Discovery And Pattern Matching In Music. In Tim Crawford and Remco C. Veltkamp, editors, *Proc. of the Dagstuhl Seminar on Content-Based Retrieval.*, Dagstuhl, Germany, 2006.

[12] David Meredith. COSIATEC and SIATECCompress: Pattern Discovery by Geometric Compression. In *Music Information Retrieval Evaluation eXchange*, Curitiba, Brazil, 2013.

[13] Meinard Müller. *Information Retrieval for Music and Motion*. Springer, 2007.

[14] Meinard Müller and Michael Clausen. Transposition-Invariant Self-Similarity Matrices. In *Proc. of the 8th International Conference on Music Information Retrieval*, pages 47–50, Vienna, Austria, 2007.

[15] Oriol Nieto and Morwaread Farbood. MIREX 2013: Discovering Musical Patterns Using Audio Structural Segmentation Techniques. In *Music Information Retrieval Evaluation eXchange*, Curitiba, Brazil, 2013.

[16] Oriol Nieto and Morwaread Mary Farbood. Perceptual Evaluation of Automatically Extracted Musical Motives. In *Proc. of the 12th International Conference on Music Perception and Cognition*, pages 723–727, Thessaloniki, Greece, 2012.

[17] Oriol Nieto, Eric J. Humphrey, and Juan Pablo Bello. Compressing Audio Recordings into Music Summaries. In *Proc. of the 13th International Society for Music Information Retrieval Conference*, Porto, Portugal, 2012.

[18] Jouni Paulus, Meinard Müller, and Anssi Klapuri. Audio-Based Music Structure Analysis. In *Proc of the 11th International Society of Music Information Retrieval*, pages 625–636, Utrecht, Netherlands, 2010.

[19] Colin Raffel, Brian Mcfee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. mir_eval: A Transparent Implementation of Common MIR Metrics. In *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014.

[20] Ron Weiss and Juan Pablo Bello. Unsupervised Discovery of Temporal Structure in Music. *IEEE Journal of Selected Topics in Signal Processing*, 5(6):1240–1251, 2011.