CHROMA AND MFCC BASED PATTERN RECOGNITION IN AUDIO FILES UTILIZING HIDDEN MARKOV MODELS AND DYNAMIC PROGRAMMING

Alexander Wankhammer, Peter Sciri, Alois Sontacchi, 
Inst. of Electronic Music and Acoustics, 
Univ. of Music and Performing Arts, 
Graz, Austria 
{wankhammer,sciri,sontacchi}@iem.at

ABSTRACT

In this paper we present an algorithm to reveal the immanent musical structure within pieces of popular music. Our proposed model uses an estimate of the harmonic progression which is obtained by calculating beat-synchronous chroma vectors and letting a Hidden Markov Model (HMM) decide the most probable sequence of chords. In addition, MFCC vectors are computed to retrieve basic timbral information that cannot be described by harmony. Subsequently, a dynamic programming algorithm is used to detect repetitive patterns in these feature sequences. Based on these patterns a second dynamic programming stage tries to find and link corresponding patterns to larger segments that reflect the musical structure.

1. INTRODUCTION

Music is one of the oldest forms of human communication and expression of feelings or experiences. Therefore it carries a lot of properties that are strongly related to the culture it stems from. Although all components that form music - from a single narrating voice to a romantic orchestra performing a symphony - can be extremely different, one common component is the existence of repetitive parts.

In the history of occidental music, the beginning of polyphony took place in the eras of late Middle Age and Renaissance [1]. Musical progression was now no longer dependent only on a single voice but had gained an additional dimension: harmony. While being a product of various contemporaneous melodies in Renaissance, Baroque composers have made strong use of harmonic relations to reinforce the dramatical progress [2]. This development has not ended until nowadays' popular music.

Therefore segment detection based on the semantic information offered by the harmonic progression exploits the same basis of information as used in conventional musicological formal analysis.

For processing different types of spectral representation we use the standard Short Time Fourier Transform (STFT) [3] as a basis of analysis. The resulting feature space is computed by means of the Constant-Q Transform (CQT) [4], [5], the chroma [6], [7], [8] and the Mel Frequency Cepstral Coefficients (MFCCs) [9], [10].

Many different approaches regarding segmentation of music have been published and discussed in literature. Levy and Sandler have proposed labelling and constrained clustering of spectral features in [11]. Aucouturier and Sandler have presented an approach using succession of timbres using MFCC in [12] whereas Wang et al in [13] exploit information gathered from the Constant-Q Transform.

form. Eronen uses a combination of MFCC and chroma features for chorus detection in [14] and Ong and Herrera have presented an algorithm in [15] that uses a set of low level descriptors to detect segment boundaries.

Our model focuses on the usage of harmonies extracted from chroma vectors. Additionally it uses MFCC vectors to extend the information basis to timbral features that cannot be detected by means of chords.

In the following sections [2] we want to present the components of our approach. In section [3] we will discuss the extraction of chroma and MFCC features, followed by the determination of the beat and the corresponding beat synchronous chord sequence in sections [4] and [5]. Pattern recognition through dynamic programming and the subsequent structural analysis will be the topic of sections [6] and [7]. The evaluation of the algorithm will be discussed in section [8] concluding the results in section [9].

2. APPROACH

The algorithm consists of the following sequential blocks (see figure [1]):

1. Segment the signal into frames and calculate spectral features for every frame,
2. Perform beat detection and compute average values for all features belonging to one beat,
3. Use chroma vectors to estimate a chord sequence by using a Hidden Markov Model,
4. Perform pattern recognition of chord and MFCC sequences using dynamic programming,
5. Structural analysis: find and group similar repetitive patterns.

3. FEATURE EXTRACTION

Obtaining information on the harmonic events causes the necessity of a spectral description of the signal. After resampling the signal to a frequency of \( f_s = 11025 \text{Hz} \) we use Hanning windowed blocks and an FFT length of \( N = 4096 \) to compute the STFT. With this set of parameters we obtain a frequency resolution of \( \approx 2.7 \text{Hz/bin} \). The hop size is set to \( k = 512 \) which allows for a temporal resolution of about 45ms.
addressed as

c

distributed over all the bins that belong to one tone - those can be

tuning frequency.

over time and the position of the peak determines the most proba-
pitch. The energy of the corresponding STFT bins is summarized

the center of energy within

± 1 quarter-tone around the standard

pitch of 440Hz we perform a tuning step by detecting

1 quarter-tone scale more tightly [16].

seventh that are derived from the overtone scale and would fit a

this measure (even though there do occur notes like the harmonic

seems more appropriate as occidental music was developed upon

1 tempered scale. Theoretically, this could be extended to quarter or

their real-world frequencies as utilized by Bartsch and Wakefield

into the 12 semitones of the well-tempered scale despite of their

To receive the chroma vector we summarize the energy that is

distributed over all the bins that belong to one tone - those can be

addressed as \( c(k_{cq}) = \text{mod}(k_{cq}, 12) \).

Thus, chroma is measure for the spectral energy that is divided

into the 12 semitones of the well-tempered scale despite of their

their real-world frequencies as utilized by Bartsch and Wakefield

A Hidden Markov Model can formally be defined as

\[
\lambda = \{ Q, A, O, B, \pi \}
\]  

where the describing parameters are:

\( Q \) . . . set of available states

\( A \) . . . transition probabilities

\( O \) . . . observations

\( B \) . . . observation or emission probabilities

\( \pi \) . . . initial probabilities

A large-scale study on the performance of approaches for chord

estimation utilizing HMMs [4] published by Papadopoulos discussed

the possibilities of predefining the model or using the EM

algorithm to perform self training. Though our tests have shown
that in some cases a self-trained HMM can perform much better than an untrained HMM, the overall results did not show much significance when analysing structure due to the very high amount of smoothing that inherently comes with self-trained HMM. They will therefore not be used in this approach.

The set of available states $Q$ is defined by musical theory as a group of 24 chords: 12 for every semitone in major (C, C#, D, D#, E, F, F#, G, G#, A, A#, B) and the corresponding 12 chords in minor. The enharmonic equivalents do not carry information for our algorithm and are therefore ignored.

The transition probability matrix $A$ can also be derived from musical theory. Transitions between certain chords have an implicit probability due to their relation on the circle of fifths. In order to guarantee accuracy regarding music theory, those values are chosen according to the “double-nested circle of fifths” as proposed in [7].

The chord sequence computed by the Hidden Markov Model needs to be relabeled according to the relations of its chords on the circle of fifths. More specifically, the symbols are assigned integer values that can be used to calculate the distance between two successive chords. In order to guarantee accuracy regarding music theory, those values are chosen according to the “double-nested circle of fifths” as proposed in [7].

The basic principle of our method is the recursive estimation of a least-cost path that indicates the repetition of a certain pattern. In our algorithm a pattern $V$ of length $M$ is compared with the following rest of the sequence. This is performed for the entire sequence in $i$ steps by shifting the pattern $V$ by the hop size $k$ at every iteration. This pattern can be one-dimensional (chord sequence) or multi-dimensional (MFCC vectors). To measure the similarity of a pattern to all observations we define the following normalized distance for the one dimensional case (eq. 2) and the varied cosine distance for the multidimensional case (eq. 3).

$$
\|\vec{v}_m - \vec{v}_r\| = \begin{cases} 
\frac{|v_m - v_r|}{12 - \text{mod}_12|v_m - v_r|} & \text{if } |v_m - v_r| \leq 12 \\
0 & \text{else}
\end{cases}
$$

$$
d_{\text{MFCC}}(v_m, \vec{v}_i) = 0.5 - 0.5 \frac{v_m \cdot \vec{v}_i}{\|v_m\| |\vec{v}_i|}
$$

As additional parameters for the algorithm, a cost of deletion or insertion factor $e$ can be defined which determines the impact ambiguities that do not match the investigated pattern.

For the computation of the dynamic programming matrix $D_i$ for the $i$-th pattern we perform an iterative, recursive calculation that includes the distance and the cost factor:

$$
D_i[m, r] = \min \left\{ \begin{array}{ll}
D_i[m-1, r] + e & \text{for } m \geq 1 \\
D_i[m, r-1] + e & \text{for } r \geq 1 \\
D_i[m-1, r-1] + d_{m,r} & \text{for else}
\end{array} \right.
$$

where $d$ is the distance of two feature vectors defined in equations 2 and 3 and $e$ is the cost factor defined as $e = (0.1 + d_{m,r})e_0$ where $e_0$ denotes a basic cost that has been set to an empirical value of 1.3

In the general form of the dynamic programming algorithm there are different values for deletion or insertion. As we do not want to distinguish those cases we decided to us only one common value.
The last row of $D_i$ can be seen as a cost function $c_i[r]$ which shows minima at the most likely positions of repetitions.

Processing the algorithm for overlapping fragments (length $M$, hop $k$) of the entire chord or MFCC sequence results in the a series of cost functions $c_i$. Regarding this series as a function of $i$ leads us to the depiction as matching matrix $M[i, r]$. This matrix is similar to Bartsch’s “time-lag surface” in [17], though the basis of his calculation is different (e.g. Bartsch filters along the diagonals to compute similarity between extended regions of the signal).

![Dynamic Programming Matrix](image)

As a final step to ease further computation we apply an expander that decreases small values below a certain threshold to amplify valleys in the dynamic programming matrix that will be used in section 7.1.

Picking values for pattern length and hop size is very delicate - on the one hand we want to conserve as much temporal resolution as possible by choosing a small hop size, on the other hand we must be careful not to pick the pattern length too small. This would result in a too general, weakly defined pattern, which would probably be detected far too many times.

Tests have shown that using a pattern length $M = 8$ beats and a hop $k = 2$ beats leads to the most significant results.

Figure 3: Calculation of the matching matrix $M[i, r]$.

As a final step to ease further computation we apply an expander that decreases small values below a certain threshold to amplify valleys in the dynamic programming matrix that will be used in section 7.1.

Picking values for pattern length and hop size is very delicate - on the one hand we want to conserve as much temporal resolution as possible by choosing a small hop size, on the other hand we must be careful not to pick the pattern length too small. This would result in a too general, weakly defined pattern, which would probably be detected far too many times.

Tests have shown that using a pattern length $M = 8$ beats and a hop $k = 2$ beats leads to the most significant results.

![Matching Matrix](image)

As shown in figure 4 the matching matrix $M[i, r]$ exhibits two major properties:

1. horizontal lines that appear clearly when exact repetitions of the pattern can be detected
2. vertical blocks that indicate transitions between highly and less similar patterns

Figure 4: Matching matrix $M[i, r]$ of “Beatles - I Wanna Be Your Man”; the vertical white lines indicate transitions between segments.

Horizontal lines indicate very high values of similarity for long periods at a constant lag. Tests have shown that those lines only appear very rarely and can only be found in songs that expose a very simple arrangement and/or instrumentation: the more details or inexact detections appear in the analysis, the more blurry the depiction of the matrix gets. Image processing algorithms can diminish the blur, but only up to a certain point.

Vertical blocks, instead, exhibit less exact information on pattern repetitions, as they do not have a clear alignment between time and lag. They indicate a more global change in similarity: parts with higher repetition probability are more biased than parts that might never be repeated at all. The clear advantage of using blocks is that they are much more distinct in cases when there are absolutely no horizontal lines detectable. This is why we will focus on extracting information from the appearance transitions of blocks.

The loss of exactitude that comes with this approach has to be compensated. We therefore separate this section into two stages:

1. Rough detection: transitions in the matching matrix $M$
2. Refinement: comparison of priorly detected segments; grouping

7.1. STAGE I: Rough Detection

To obtain a simple representation of the “events” visible in the matching matrix $M$ we use a varied difference measure $\phi[i]$ called “repetition probability flux”. It can be derived from the first-order forward difference $d[n]$ that can be denoted in the general form

$$d[n] = |x[n] - x[n + 1]|^2$$  (5)

where $n$ indicates a temporal unit.
that should then match segments like verse or chorus. The rough detection step often results in a large amount of segments which are located at the positions of the peaks in the flux function. This method enables us to combine multiple short segments into more meaningful units that fit the musically perceived segments much more accurately.

8. EVALUATION

In order to evaluate the performance of the algorithm the resulting segmentation is compared to a ground truth. This testing corpus is similar to the one used by Levy and Sandler in [11] and consists of 32 pop songs.\(^2\) One half is formed by 16 works by The Beatles as they have emerged to a standard in evaluation of similar tasks. The other half consists of 16 more recent pieces by artists like Björk, Britney Spears, Eminem, Madonna and Michael Jackson.

\(^2\)http://www.elec.qmul.ac.uk/digitalmusic/downloads

Proc. of the 12th Int. Conference on Digital Audio Effects (DAFx-09), Como, Italy, September 1-4, 2009

1. the \(j\)-th segment is compared to the rest of the song using chroma vectors as a representation of spectral events, and beats as the time basis. The last row of the dynamic programming matrix exposes certain minima.
2. the \((j+k)\)th segment is attached in a loop to the segment \(j\) where \(k = 1\) at the first iteration. Again the prolonged segment is compared to the rest of the song.

3. the iteration of \(k\) is stopped if the following condition is fulfilled: the merged segment length exceeds a significant border - it contains data that should not be part of it. Thus, the probability or position of its repetition will change (see figure 6 at the topmost line, where the deep valley is suddenly flattened).
4. the next iteration begins at segment \(j = j + k\)

The deep valleys indicate very distinct detections of repetitions. Figure 6 displays four iterations of the algorithm for the \(j\)th segment. As we can see the first three lines exhibit a very clear valley at 48 beats. In the fourth step we prolonged the segment too much, so that it already contains a part of the following musical unit. The flattening of the valley tells us that the merged segment is not repeated at the position of the valley. Thus, we can assume that we have exceeded a critical length and have to stop the iteration.

As a result we receive a first, rough estimate for segment borders which are located at the positions of the peaks in the flux function.

7.2. STAGE II: Refinement

The rough detection step often results in a large amount of segments that can be identified as repeating patterns in the song. However, many of them are quite short excerpts that would not be declared as a musical unit like verse or chorus. This is due to the fact that the detection of repetitive elements on a basis of 8 beats only carries a limited amount of information about the large scale musical structure of a song.

Thus, we need to combine multiple segments into larger units that should then match segments like verse or chorus. This is done in an iterative approximation starting at \(j = 1\) and using again dynamic programming to measure the similarity:
Defining a ground truth for musical segmentation is a very crucial task. People define transitions between segments very differently e.g. repetitions within a verse can be annotated as one large or two smaller parts. The body used here has been defined by either professional musicians and/or musicologists for the MPEG-7 working group.

Performing the evaluation on basis of the ground truth needs a certain amount of allowed deviation. In this case we followed literature and picked a trust interval of 3s.

Subsequently the performance of the algorithm is measured by precision \( p \), recall \( r \) and the f-value \( f \) which are calculated by relations between correctly detected borders \( \text{truePos} \), incorrectly detected borders \( \text{falsePos} \) and missed borders \( \text{falseNeg} \):

\[
p = \frac{\text{truePos}}{\text{truePos} + \text{falsePos}}
\]
\[
r = \frac{\text{truePos}}{\text{truePos} + \text{falseNeg}}
\]
\[
f = \frac{2pr}{p + r}
\]

The calculated values are depicted for different corpora in table 1.

### Table 1: Performance measures after stage I.

<table>
<thead>
<tr>
<th>Corpus</th>
<th>precision ( p )</th>
<th>recall ( r )</th>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beatles</td>
<td>0.50</td>
<td>0.83</td>
<td>0.61</td>
</tr>
<tr>
<td>Recent</td>
<td>0.70</td>
<td>0.73</td>
<td>0.70</td>
</tr>
<tr>
<td>Overall</td>
<td>0.62</td>
<td>0.77</td>
<td>0.65</td>
</tr>
</tbody>
</table>

#### 9. CONCLUSION

The combination of chords and MFCCs is able to detect small musical units in a very accurate way. However, these segments do not match larger musical structures immediately. As visible in table 1, the recall value is quite high and guarantees a detection of almost every border whereas the precision value is relatively low.

When examining the results by listening to segmented audiofiles, the system is able to generate very distinct segments that can also be very suitable as a loop because of its beat-synchronous cutting.

The task of stage II is to combine shorter patterns into larger segments. This leads to a decrease of recall, but for some songs it leads to a very significant rise of precision. However, stage II comes with the danger of discarding too many - previously correctly detected - borders.

As already mentioned previously, absolute or platonic ground truth for segmentation of music does not exist. For instance, there are many different ways to perceive transitions between a verse and a chorus. There is, for example, no clear definition whether the chorus begins on the first beat, even though it has a pickup in the previous measure or if an immediate repetition of the chorus causes two segments or only one.

Due to this missing ground truth, the evaluated values have to be handled with a lot of care - the Beatles corpus, for instance, shows some ambiguities and inconsistencies concerning the separation of verses into smaller units.

Generally we could observe that the entire system is vulnerable to the used set of parameters. A certain setup can perform perfectly on some songs but have less appropriate results for the rest of the corpus. Changing the parameters slightly leads to accurate results in another group of songs while the formerly perfect group loses accuracy.

#### 10. FURTHER DEVELOPMENT

A very promising path to continue this work is to improve the performance of the second stage.

Many songs have exposed unusually high accuracy after applying the refinement step. The short segments detected by the first stage could be combined to meaningful larger elements that coincided with the reference segmentation very accurately. However, these results did not hold for the entire testing set. We therefore need to enhance the portability of the second stage to many different songs of diverse genres.

### 11. REFERENCES


