Harmonisation in modern rhythmic music using Hidden Markov Models

Nikolas Borrel-Jensen
Founder, Livetake
Denmark
Email: nb@livetakeconcert.com

Andreas Hjortgaard Danielsen
Machine Learning Specialist, Alexandra Institute
Denmark
Email: andreashd@gmail.com

***Abstract*—A method for harmonising rhythmic music is presented. It uses a hidden Markov model to learn harmonisations of different artists in different genres and allows new chord sequences to be generated with respect to a given melody. The main focus of this article is on how to perform the feature extraction for the chords and the melody lines necessary for the method to perform well. We show the results of the harmoniser by discussing 3 different harmonisations using a simple music theoretical analysis of the results.**

***Keywords*-Automatic harmonisation; algorithmic composition; hidden Markov models; musical feature extraction.**

I. Introduction

The goal of harmonisation is to provide a series of chords underlying a given melody such that the correspondence between melody and chords makes sense musically. In this text we describe a probabilistic method for automatic harmonisation of rhythmic music which can be used by the composer as inspiration for harmonising a piece in progress. We have created a database of music consisting of tunes from two different genres: Melodic rock and standard jazz. These tunes are used to learn the parameters of a hidden Markov model, which in turn can be used to generate new harmonisations given new input melodies. The system implementing this method along with the database is called *Cremo*, an abbreviation of *Cre*ative *Mo*delling.

The harmonisations generated by this model are not meant to be used directly as the final harmonisation, but rather as a source of inspiration for the composer and therefore we avoid the challenge of voicing the chords. We perform a feature extraction that simplifies each chord in order to reduce the dimensionality of the problem. We also perform feature extraction on the melody for making it more likely to find similar patterns in the database. How to extract these features is the main focus of this paper and will be discussed thoroughly in the following sections.

Several texts exist on automatic harmonisation, most notably the article by Moray Allan [1] in which a method for harmonising Bach chorales is presented. Chorales consist of four voices: A soprano, an alto, a tenor and a bass. Considering each soprano note as an observable variable and the three remaining voices as a triad chord, a hidden Markov model is used to learn the style of Bach chorales in order to generate new chorales using the Viterbi algorithm or by sampling.

Hörnel and Menzel [2] have developed a system called *HARMONET* for harmonising 4-part Bach chorales in the style of a composer, given a one-part melody. HARMONET integrates feed-forward neural networks and symbolic algorithms. The neural network is responsible for aesthetic conformance to a given set of training examples by a composer, whereas conventional algorithms do the bookkeeping task, such as observing pitch ranges and obeying the rules for harmonising chorales.

The paper is organised as follows: In Section II the hidden Markov model is explained and how it is used as a model for harmonisation is clarified. Section III and IV describes how chords and melodies are represented, respectively, and how feature extraction is done. In Section V we discuss the results of the method by harmonising 3 different melodies and finally in Section VI we conclude on the method and present future enhancements. Throughout the text we assume that the reader is familiar with basic music theory, i.e. knows about diatonic scales, scale degrees and chord progressions.

II. Hidden Markov Models for harmonisation

Following the same observations as Moray Allan [1], we consider the chords as an underlying Markov chain emitting a melody according to a hidden Markov model (HMM). This makes sense musically as the notes played in a melody is dependent on the underlying chord progression. A HMM is a statistical model in which we have a sequence of observable variables $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ and a corresponding sequence of latent variables $\{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$. The latent variables form a Markov chain, such that

$$p(\mathbf{z}_n|\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N) = p(\mathbf{z}_n|\mathbf{z}_{n-1}) \quad (1)$$

Each observable variable only depends on its corresponding hidden variable, that is

$$p(\mathbf{x}_n|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N, \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N) = p(\mathbf{x}_n|\mathbf{z}_n) \quad (2)$$

Figure 1 shows a graphical model of a HMM illustrating these dependencies.

For harmonising we let the hidden variables represent chords and the observable variables represent a melody line over a given chord. Thus we see that each chord is dependent on the previous chord due to the Markov property which makes the model able to learn chord progressions. Having a training set of tunes in a given genre, we can train the HMM to fit the given genre by

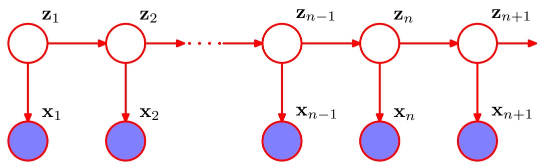

Figure 1. A graphical model of the hidden Markov model

computing the transition matrix for the Markov chain and the emission matrix for the observable variables. Since we have both chords and melody in the training set we can compute these directly. Having learned the parameters of the HMM we can now use it to generate a new harmonisation to a given input melody using the Viterbi algorithm (see [1] and [3]).

Because we need to treat chords and melody lines as different variables, we need to store them separately in our database, since extracting chords and melody in a mixed environment is not a trivial task. The melody line consists of single notes whereas the chord line consists of parallel notes of the chords usually placed at the first or third beat of each bar. Storing the data in the MIDI format makes the task of splitting up very easy. Figure 2 shows an example snippet of the chorus of the Mercury Rev song *Chains* in which the melody is stored in channel 1 and the chords are stored in channel 2.

III. Harmonic Features

The basic harmonic units used in rock and jazz are triads and seventh chords and can be build from diatonic scales. This skeleton is usually extended with additional notes for colouring the chords where these implied extensions are chosen with respect to the harmonic function of the chords. By only considering the functional harmony of the chords (tonic, subdominant etc.) we can extract the important notes of the chords and discard the extensions and therefore reduce the quantity of chords in our training data. This reduces the dimensionality of the problem and helps Cremo choosing the right chords for a given melody sequence. When categorising the chords by harmonic function, the important notes are the third and the seventh scale degrees. The fifth degree is only important when augmented. Though, for dominant chords we retain the augmented fifth and the extension $\flat 9$ used for an altered sound.

The final set of extracted chords can be seen in Table I and II. The chords depicted in Table II are so-called *slash chords* in major. Slash chords are chords where the bass note of the chord is not the actual root of the chord (e.g. C/E or Cmaj7/D).

A. Implementation of harmonic features

We represent the chords as the absolute pitch of the bass note together with the relative distance from that note to each of the chord notes. We denote the interval by integer values from 0 to 11, where 1 is a minor second, 2 is a major second, and so on. This representation is unambiguous but the original voicing can not be reconstructed (but

Triads
Major
Minor
sus4
sus2
5 (power chord)
$^\circ$ (diminished)

Sixth/Seventh chords
maj7
maj7 $\sharp$11
maj7 $\sharp$5
m7
mmaj7
mmaj7 $\sharp$5
7
7, $\sharp$5
7$\flat$9
7 sus4
7$\flat$9 sus4
7sus2
$^{\varnothing}$7
$^\circ$7

Table I
Basic chord types with only the most important extensions present. The 9 sus4 chord is not depicted since it corresponds to the slash chord I/II depicted in Table II.

Slash chord	**Chord with respect to the root**	**Relative representation**
I/$\flat$II	mmaj7 $\flat$5	-
I/II	9sus4	$\{2, 5, 10\}$
I/$\flat$III	6 $\flat$9 $\flat$5	-
I/III	m$\flat$6	$\{3, 8\}$
I/IV	maj7 sus2	$\{2, 7, 11\}$
I/$\flat$V	$\flat$7 $\sharp$11 omit 3	$\{6, 10\}$
I/V	6sus4	$\{5, 9\}$
I/VI	m7	-
I/$\flat$VII	6 $\sharp$4 add9	$\{2, 6, 9\}$
I/VII	6 $\flat$9 sus4	{1,5,8}

Table II
Basic major slash chords.

the chord function can). The root of the chord is explicitly added when reconstructing the chord by adding a 0 to the chord.

For extracting the chords depicted in Table I and II, we distinguish between chords with a third and chords with no third present. The chords depicted in boldface in Table II are the slash chords we will support, since the chords in plain text can be interpreted as normal chords with the third present. Sus2 and sus4 chords will also be treated like slash chords in our implementation, since no third is present. For the non-slash-chords, we will distinguish between dominant and non-dominant chords. This partitioning can be seen in Table III.

The implementation is done by considering chord notes as sets and is done as follows: For chords with a third, let

Chords without third	**Chords with third**	
	Dominant	Non-dominant
I/II, I/III, I/IV, I/$\flat$V, I/V, I/$\flat$VII, I/VII, sus2, sus4, 7sus4, 7sus2, 7$\flat$9sus4, power chord	7, 7$\sharp$5, 7$\flat$9, 7$\sharp$5$\flat$9	major, minor, $^\circ$, maj7, maj7$\sharp$4, maj7$\sharp$5, m7, mmaj7, mmaj7$\sharp$5, $^{\varnothing}$7, $^\circ$7

Table III
Basic chords after feature extraction.

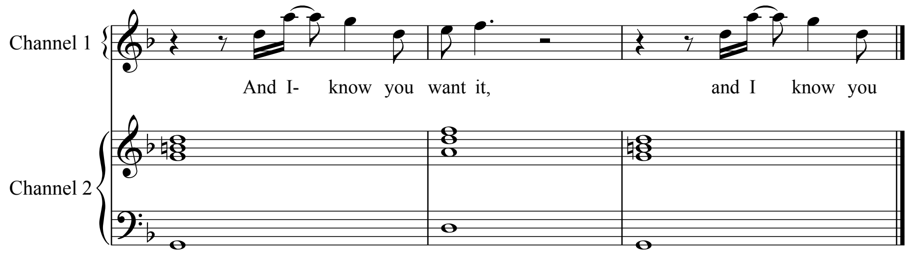

Figure 2. This snippet of the chorus of the Mercury Rev song *Chains* shows the structure of the MIDI files. Channel 1 contains the main melody and channel 2 contains the chords.

H_k be a set of relative distances to the bass note of the k'th chord, and let E be the set containing the notes we want to extract. Then we can get the simplified chord S_k as

$$S_k = H_k \cap E \tag{3}$$

For non-dominant chords, we will use

$$E_{\text{non-dom}} = \{3, 4, 6, 8, 10, 11\} \tag{4}$$

and for dominant chords we will use

$$E_{\text{dom}} = \{1, 4, 8, 10\} \tag{5}$$

Using these sets, all the important notes from the chords depicted in column 2 and column 3 in Table III can be extracted.

Example: From the chord $\{2, 4, 7, 10\}$, corresponding to a major 9 chord, the important notes are the third and the seventh:

$$S_k = \{2, 4, 7, 10\} \cap \{1, 4, 8, 10\} \tag{6}$$
$$= \{4, 10\} \tag{7}$$

$$H_{\text{C9}} = \{2, 4, 7, 10\} \tag{8}$$

$$H_{\text{C7}\flat 9} = \{1, 4, 7, 10\} \tag{9}$$

$$H_{\text{CmMaj9}} = \{2, 3, 7, 11\} \tag{10}$$

$$S_k = \{2, 3, 7, 11\} \cap \{3, 4, 6, 8, 10, 11\} \tag{11}$$
$$= \{3, 11\} \tag{12}$$

Example: From the chord $\{1, 4, 7, 10\}$, corresponding to a major 7♭9 chord, the important notes are the third, the seventh and the ♭9:

$$S_k = \{1, 4, 7, 10\} \cap \{1, 4, 8, 10\} \tag{13}$$
$$= \{1, 4, 10\} \tag{14}$$

By this method we get a simpler chord without the ninth and fifth as depicted in Figure 3, which are not important for the harmony.

For chords without a third, we would like to simplify extended slash chords to the simple forms in column 1 in Table III. We use the relative representation given in Table IV and the representation for sus and power chords in the third column in Table II.

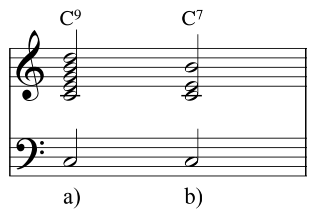

Figure 3. Feature extraction for the chords. a) Original C9 chord. b) Chord after feature extraction, which resolves to C7 without the fifth.

Power and sus chords	Relative representation
sus4	{5,7}
7sus4	{5,10}
7♭9sus4	{5,10}
sus2	{2,7}
7sus2	{2,10}
5 (power chord)	{7}

Table IV
RELATIVE REPRESENTATION FOR SUS2, SUS4 AND POWER CHORDS.

To find the simplified representation, we perform the following nearest distance set difference: Let again H_k be a set of relative distances to the bass note of the k'th chord and let C_m be the set of relative distances to the bass note for the chords $m = 1, \ldots, M$, where $M = 13$ is the number of different chords in column 1 in Table III. Then the simplified slash chord C_m, for which the set difference with H_k is smallest, is given by

$$\{C_I | I = \underset{m=1,2,\ldots,M}{\operatorname{argmin}} \|H_k \setminus C_m \cup C_m \setminus H_k\|\} \tag{15}$$

If two or more of these set differences contain the same number of elements, we give first priority to the chord containing a minor seventh. This ensures, that the 7sus4 with a fifth present and a 7sus4 13 chord will map to {5,10} instead of {5,7} and {5,9} (sus4 and I/V), respectively.

Example: For simplicity, we will only consider the major slash chords from Table II in this example. Given a chord Cmaj7/D with the representation $\{2, 5, 9, 10\}$

respective to bass note ‘C’, we get

$$\begin{array}{rcl}
\{2,5,9,10\} \setminus \{2,5,10\} & \cup & \\
\{2,5,10\} \setminus \{2,5,9,10\} & = & \{9\} \\
\{2,5,9,10\} \setminus \{3,8\} & \cup & \\
\{3,8\} \setminus \{2,5,9,10\} & = & \{2,3,5,8,9,10\} \\
\{2,5,9,10\} \setminus \{2,7,11\} & \cup & \\
\{2,7,11\} \setminus \{2,5,9,10\} & = & \{5,7,9,10,11\} \\
\{2,5,9,10\} \setminus \{6,10\} & \cup & \\
\{6,10\} \setminus \{2,5,9,10\} & = & \{2,5,9\} \\
\{2,5,9,10\} \setminus \{5,9\} & \cup & \\
\{5,9\} \setminus \{2,5,9,10\} & = & \{2,10\} \\
\{2,5,9,10\} \setminus \{2,6,9\} & \cup & \\
\{2,6,9\} \setminus \{2,5,9,10\} & = & \{5,10\} \\
\{2,5,9,10\} \setminus \{1,5,8\} & \cup & \\
\{1,5,8\} \setminus \{2,5,9,10\} & = & \{1,2,8,9,10\}
\end{array}$$

and we see that the nearest chord is $\{2,5,10\}$, corresponding to the slash chord I/II, which is resolved to the absolute chord C/D.

IV. Melodic features

The observable variables in the hidden Markov model correspond to the melody line above each chord, so we need a way to extract the important melody notes. A time window is placed above each chord indicating which notes to be considered as important. This time window can be chosen in different ways as seen in Figure 5. Here we see three possible options, two of which are constant: A half measure (shown as a solid line) and a whole measure (shown as a dashed line). Since we place chords with constant duration (either a half or a whole measure) when generating harmonisations, having a constant time window with equal length when extracting importance notes should keep a good correspondence between the input features and the database features. As seen in Figure 5, using a window that is larger than the length of the chord results in extracted melody notes that belongs to the next chord. On the other hand, using a window that is too small, we may miss important notes. Using a varying time window is problematic, since we use a constant time window in the input melody and hence the foundation for comparing the melody sequences is not fair.

We have chosen to use a fixed time window, where chords in the database with a duration smaller than the desired chord duration are disregarded. For chords with a greater duration, the corresponding melody line is cut of at the chosen duration and the rest of the melody is discarded.

Having chosen a proper time window, we have to choose which of the notes to extract. We could extract all notes, but often some notes are more important than others for determining the harmonic context. For melody lines with chromatic motifs, the notes on the offbeats will often be notes not determining the underlying chord, and therefore it can be favourable only to extract the notes on the downbeats (e.g. 4/4 notes). On the contrary, we may be too rough and throw away the characteristics of a composer for choosing a specific chord and therefore extracting all eight notes could capture these details. Whether to use quarter note or eighth note quantisation is a parameter that is dependent on the input melody. An example of an eighth note quantisation is seen in Figure 4. We see that while some details are lost due to the quantisation, the overall idea of the melody is retained.

After quantising the melody the remaining notes placed within the time window are extracted. Any information about octave for each note is disregarded before feeding to the HMM. This ensures that to notes lying one octave apart are still measured as the same note. Thus we end up with features being a list of relative note values corresponding the the notes placed on the downbeats (or the eighths) within the time window above each chord.

A. Nearest neighbour

When extracting features for the input melody we have a constant chord duration for all the generated chords, so we can quantise and extract notes exactly as described above with the given duration. But before feeding the features to the HMM we perform one final step: A nearest neighbour transformation. The distance measure is simply the length of the set difference between the input melody feature and the features from the database. Let $M_{I,k}$ denote the melody features of the input melody for the k'th chord and let $M_{db,l}$ denote the melody features for the l'th chord in the database. Thus the distance between between $M_{I,k}$ and $M_{db,l}$ is

$$\delta_{k,l} = \|(M_{I,k} \setminus M_{db,l}) \cup (M_{db,l} \setminus M_{I,k})\| \tag{16}$$

This distance is calculated between each sequence in the melody corresponding to the chord l to insert, and each sequence in the database corresponding to the chord k. The input melody feature is replaced with the nearest neighbour from the database. This ensures that the feature given to the HMM is something that is actually present in the database. In case of a tie we choose randomly among the melody parts of equal distance. An example of the nearest neighbour calculation is shown below.

Example: Let $M_{I,k} = \{A,D,G\}$ and $M_{db,k} = \{A,G,C,F\}$, then $\delta_{k,l}$ is given by

$$\begin{array}{rcl}
\delta_{k,l} & = & \|(\{A,D,G\} \setminus \{A,G,C,F\}) \\
 & & \cup\ (\{A,G,C,F\} \setminus \{A,D,G\})\| \\
 & = & \|\{D\} \cup \{C,F\}\| \\
 & = & \|\{D,C,F\}\| \\
 & = & 3
\end{array}$$

V. Results

Judging the performance of the Cremo harmoniser is not a trivial task since harmonisations in rhythmic music are not bound by strict rules that we can validate. Also, what sounds harsh and challenging for some might sound

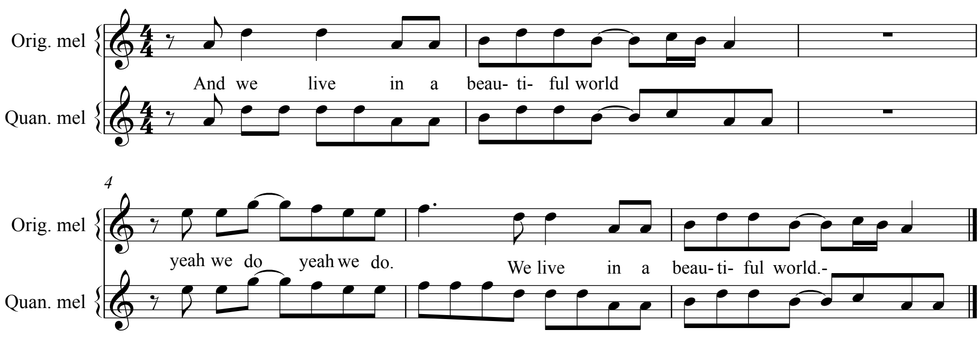

Figure 4. An example of the melody quantisation into eighth notes. Notice that quarter notes are split into two eighth notes and sixteenth notes are removed.

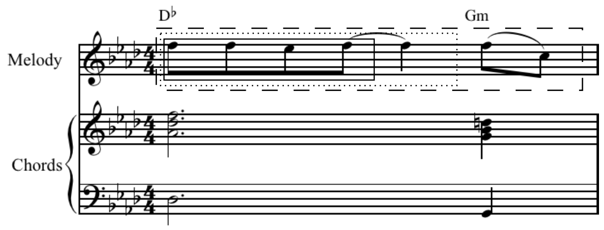

Figure 5. Three possibilities for extracting the melody corresponding to the chord in beat 1 is shown. The solid line shows the notes extracted for a half measure, the dashed line shows the the notes extracted for a whole measure, where the dotted line shows the extracted notes corresponding to the length of the chord. We see that extracting a whole measure results in the melody for another chord in beat 4 to be included for the chord in beat 1, which may not be an optimal choice.

interesting and intriguing to others, especially in the jazz genre. So in order to test Cremo we look at the effect of the harmonic feature extraction and perform a simple music theoretical analysis on three harmonisations.

A. Feature extraction for the chords

In Figure 6 we see all the different chords for the melodic rock and standard jazz database plotted against the number of occurrences for each chord. The left column shows the chords as they are in the database and the right column shows the chords after being simplified by the feature extraction method described in Section III. We clearly see a reduction in the number of different chords around 60 %, which gives the HMM method more possibilities for making more rational harmonisations due to the reduced dimensionality.

B. Harmonisations

Our training data[1] is derived from tunes in the two genres melodic rock and standard jazz. The melodic rock genre consists of 86 tunes composed by Coldplay, Radiohead, Muse, Jeff Buckley and Mercury Rev, whereas the standard jazz genre consists of 43 tunes composed and/or performed by Bill Evans, Charlie Parker, John Coltrane, Miles Davis, Thelonious Monk and Wayne Shorter.

We have generated three harmonisations on three different melodies, two in the melodic rock genre and one in the standard jazz genre[2].

1) Harmonisation of "Burma": In Figure 7 the harmonisation of the tune *Burma* composed by Börsenfieber can be seen. The parameters used are 2 chords per bar, 8th note quantisation and a fixed time window of 1 bar with feature extraction on the chords. All artists in the melodic rock genre are used.

In bar 1-6 we see that the harmonisation has made use of many tonic, subdominant and dominant chords, where the first dominant chord in the 6th bar is suspended and resolved musically in the following chord. In bar 7-12 we modulate to the parallel key, namely Am, where the subdominant chord Dm and the dominant chord E is used. This is an interesting choice, since the original harmonisation is clearly in the key of C major, but since the dominant chord E does not clash with the melody, this is a completely valid choice. In bar 13-20 the harmonisation modulates back to the original key of C, although the chords D and A are major chords, but again, since no notes F and C are present in the melody for the chord D and A, respectively, this is a valid choice. Notice that the bass line is very coherent with few big intervals, giving smooth transitions between the chords.

2) Harmonisation of "KTAS Blues": In Figure 8 the harmonisation of the tune *KTAS Blues* composed by Christian Munch can be seen. The parameters used are 1 chords per bar, 8th note quantisation and a fixed time window of 1 bar with feature extraction on the chords. All artists in the standard jazz genre are used.

This is a bluesy tune with the characteristics of using both minor and major thirds on the tonic chord. Normally, a major seventh chord is used on tonic, subdominant and dominant chords, but here we see that Cremo has chosen to use a completely different approach possibly because of the sparse number of jazz blues in the database and that the tune is not in the typical style of blues. In general there are many harsh notes clashing with the chords, but

[1]The database is constructed by scanning music sheets and adjusting them to the format of Cremo.

[2]The harmonisations in MIDI format can be downloaded from http://cremo.nikolasborrel.com.

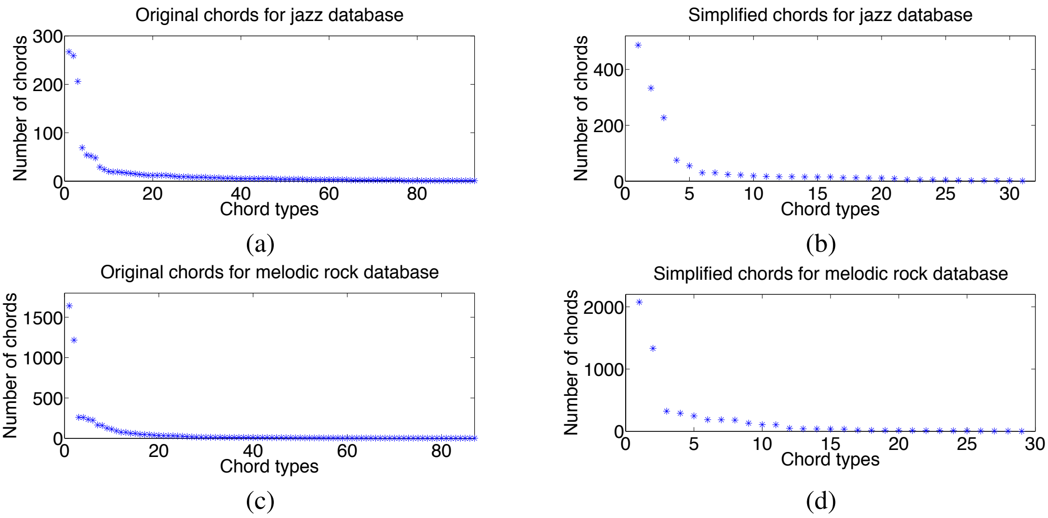

Figure 6. Number of chords for each chord type. a) The original chords for the tunes in the jazz genre consisting of 93 different chords. b) The simplified chords for the tunes in the jazz genre after feature extraction consisting of 31 different chords. c) The original chords for the tunes in the melodic rock genre consisting of 87 different chords. d) The simplified chords for the tunes in the jazz genre after feature extraction consisting of 29 different chords.

Figure 7. The harmonisation of the tune "Burma" composed by Börsenfieber using all artists in the melodic rock genre.

Figure 8. The harmonisation of the tune "KTAS Blues" composed by Christian Munch using all artists in the standard jazz genre. Be aware that the E13sus4(omit9) chord corresponds to the chord A/E.

Figure 9. The harmonisation of the tune "Prins Na Na" composed by Nils Bouchet using only the artist Jeff Buckley.

in many cases these notes are placed on the offbeat and therefore it sounds fine with a jazzy vibe. In bar 1-3 we have a whole tone descending movement where the important thirds correspond to the chosen chord and a nice ♯11 melody note has been chosen for the E♭maj7 chord. A characteristic II-V-I progression in minor can be seen in bar 5-8, where the chord FmMaj7 fits perfectly to the melody notes. In bar 9 and 10 we have a V-I progression to minor, though a minor 7 is used for the tonic chord clashing with the melody note E. This is also the case in the 14th bar, where also the third of the melody is clashing with the chord. The melody in the last bar makes heavy use of leading-notes to the important notes E, A and C♯, but Cremo sees through this by choosing a chord with a prim, fourth and seventh note corresponding to these notes.

3) Harmonisation of "Prins Na Na": In Figure 9 the harmonisation of the tune *Prins Na Na* composed by Nils Bouchet can be seen. The parameters used are 1 chord per measure, 8th note quantisation and a fixed time window of 1 bar without feature extraction on the chords. Only the artist Jeff Buckley is used.

In the first 11 bars only tonic and subdominant chords are used – which fits the melody – except for a misplaced E♭7 dominant chord in the first bar. Also a suspended chord in the 11th bar is well-resolved in the following bar reaching its climax on the dominant chord in the 12th bar leading to a tonic chord with the third in the bass. In bar 13 and 14 Cremo modulates to the parallel key Cm and introduces the progression Im-IIm7b5. Surprisingly, Cremo changes the minor key to major and finalise with an augmented chord that never resolves.

VI. Conclusion and further work

Cremo, an automatic harmoniser, has been implemented and tested. Harmonisations of a given input melody in the style of a specific genre can be generated by doing probabilistic analysis on a database using hidden Markov models in which we model a piece of music as a series of chords forming a Markov chain, where each chord emits a melody line. In order to do this, we perform a series of feature extraction steps on both the chords and melody of the database pieces and input melody. The feature extraction on the chords cuts the number of possible chords down about 60 % thereby making it easier for the HMM to choose the proper chords.

The feature extraction for the melody lines makes it possible for us to measure similarity between two melody parts without sacrificing too much of the uniqueness of the individual parts. This gives us a way to compare melody parts of the input melody with parts in the database and substitute the input part with a similar part from the database thereby helping the HMM to choose a good chord.

Three different input melodies have been harmonised by Cremo and analysed. Cremo is able to generate chords that fits the melody with no or very few clashing tones and it even recreates common chord progressions from the different genres. For example, the jazz harmonisation contains a classic minor II-V-I progression. The method is not perfect, though, as Cremo also makes some challenging choices from time to time, e.g. changing from minor to major in the final two chords of the tune "Prins Na Na" or by placing single out-of-tune chords at different places. In general, Cremo is able to generate harmonisations that fit the given genre and that match the input melodies.

Since the chords are assumed to form a first-order Markov chain the algorithm will only consider the previous chord along with the melody when choosing the current chord. By placing dependence on chords to, say, the two or three previous chords should make chord progressions more consistent and true to the given genres. Also, instead of finding the best fitting chord sequence using Viterbi, a Monte-Carlo sampling could be performed instead. This would make the method able to generate several different harmonisations with the same parameters.

The harmonisation method is constructed in such a way that it will be able to harmonise melodies in any genre, as long as a training set in the given genre is present in the database. Therefore more testing on the performance of different genres would be interesting.

References

[1] M. Allan and C. K. I. Williams, "Harmonising chorales by probabilistic inference," 2005.

[2] D. Hörnel and W. Menzel, "Learning musical structure and style with neural networks," *Computer Music Journal*, vol. 22, no. 4, pp. 44–62, Winter 1998.

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer Science, 2006.