DYNAMIC CHORD ANALYSIS

Thomas Rocher, Matthias Robine, Pierre Hanna, Robert Strandh

LaBRI
University of Bordeaux 1
351 avenue de la Libration F 33405 Talence, France
simbals@labri.fr

ABSTRACT

In this paper, we present a new method for chord recognition from audio. This method builds a graph of all possible chords and selects the best path in this graph. A rule-based approach is adopted to enumerate chord candidates from groups of notes by considering compatible chords and compatible keys. The distance proposed by Lerdahl is then used to compute costs between different chord candidates. Dynamic programming is also involved to select the best path among chord candidates. Experiments are performed on a MIDI song database, divided in different music styles. Then the proposed system is compared to the *Melisma Music Analyzer* software proposed by Temperley. Results show that our method has a comparable efficiency and provides not only the root of the chord, but also its mode (major or minor). The proposed system is still open and is able to support more chord types if correct rules to handle them are specified.

1. CHORD REPRESENTATION

A chord can be defined in different ways. A chord can simply be defined as the perception of several notes sounding at the same time. However, this first definition is not entirely compatible with the jazz chord notation, where each bar may be labeled with a chord. In this case, the pianist, for example, is not supposed to play a whole note of the specified chord: his only restriction is to play notes or groups of notes belonging *mostly* to the specified chord (some ornaments might be played, which do not belong to the chord). The difference between these two possible definitions of a chord is illustrated in Figure 1. Depending on the definition adopted, it is possible to see either 8 chords or only a single one in this excerpt.

In this paper, we choose the second definition, and we use *chord* to mean the common label a common sequence of notes or groups of notes that could sound at the same time (like the Bb^6 chord label in Figure 1). These consecutive notes or groups of notes within the same chord are called *note segments* of this chord. Each note segment of a given chord may not contain all the notes of the chord, and may contain ornaments (notes which do not belong to the chord).

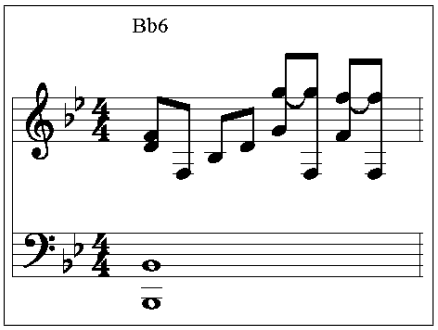

Figure 1. An excerpt of *Bohemian Rhapsody* by Queen. According to a first possible definition, there are 8 different chords in the presented bar (as much as groups of notes sounding at the same time). According to a second possible definition, there is only one single chord in this bar (characterized by the Bb^6 chord label).

For example, both (C,E,Bb) and (C,G) can be note segments of the C^7 chord. In this paper, the 3 parameters of a chord are:

- the root (the note upon which the chord is built),
- the type (the component intervals defining the chord relatively to the root)
- the mode (major, minor or undefined)

2. DYNAMIC CHORD ANALYSIS

In this section, we first propose a new method for time segmentation by using note segments. The graph of chord candidates is then created, before the dynamic process takes place to select the best path in this graph.

2.1. Time segmentation

The first issue in the chord detection process is to find an appropriate time segmentation. Audio based methods [**?**, **?**, **?**] use time frames in this purpose. It is possible to adopt a similar approach in symbolic music, by using MIDI ticks (or milliseconds) as time units to set the length of an analysis window. We choose a different approach, and assume that a new chord could only occur with a new instance, *i.e.* when

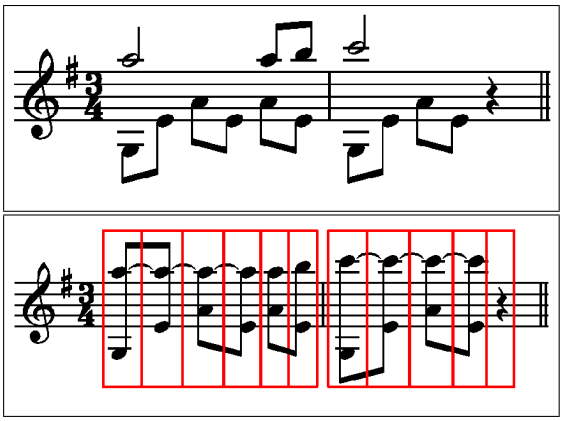

Figure 2. Above: a musical excerpt. Below: the same excerpt with an homorhythmic transformation. Note segments appear bordered in red.

at least one new note starts being played, or stops being played. We thus chose to perform an homorhythmic transformation, as introduced in [**?**], in order to make all the notes sounding at the same time start and end at the same time as well. Therefore no overlapping between notes occurs. This time segmentation defines the different note segments, each of them being formed by notes starting and ending at the same time. Each of these segments potentially starts a new chord. It is important to note that this transformation does not affect the way music is played and heard. An illustration of this homorhythmic transformation can be found in Figure 2. Once this transformation completed, the enumeration of chord candidates can take place for each note segment.

2.2. Graph of chord candidates

The note segment constitutes the observation of the dynamic process. A list of hypotheses is built from each observation, these hypotheses being the chord candidates. In the method proposed, a chord candidate is a pair, composed of a compatible chord and a compatible key. Rule-based algorithms are used to determine which chords and keys are compatible with each note segments. The graph of chord candidates is then built: each chord candidate of a given segment is linked to all the candidates of the next segment, thus forming an directed acyclic graph.

2.2.1. Compatible Keys

The proposed method enumerates which keys are compatible with each note segment, by using a rule-based approach. Different rules may be used for that purpose. In this paper, we choose to define a key as compatible if each note of the note segment is a component pitch of this key, which means that each note of the segment must belong to the scale of the key (melodic scale for the minor mode). For example, if the note segment is (C,E), the compatible keys are C_{Maj}, F_{Maj}, G_{Maj}, A_{min}, D_{min}, and E_{min}, because both C and E belong to the scales of these keys.

2.2.2. Compatible Chords

As for compatible keys, several rules may be defined to determine which chords are compatible with a note segment. We use the following rules, and define different conditions for a chord to be compatible with a note segment, depending on the chord type:

- Maj/min triad chord: such a chord is compatible if each note in the considered segment belongs to the chord. For example, the note segment (C,E) has two compatible triad chords: A_{min} and C_{Maj}.
- Maj/min 7th chord: such a chord is compatible if each note in the considered segment belongs to the chord and if the root and the Maj/min 7th note are present. Such a rule is to avoid having note segment like (E,G,B) or (E,G) be compatible with C^7_{Maj}, for example.
- Maj/min 9th chord: such a chord is compatible if each note in the considered segment belongs to the chord and if the root, the 5th note and the 9th note are present,
- Maj/min 11th chord: such a chord is compatible if each note in the considered segment belongs to the chord and if the root, the 5th note and the 11th note are present.
- other chords may also be compatible, like the min75b chord. For these chords to be compatible, each note is required (for example, a min 75b chord wold be compatible if the considered segment contains the root, the minor 3rd, the flat 5th and the minor 7th).

The proposed system can be modified in order to accept more chord types in the future. Therefore, new rules for any new chord to be compatible with a given note segment must also be specified.

2.2.3. Chord Candidates Enumerated

The chord candidates finally enumerated are all the possible combination of compatible keys and compatible chords. If n chord and m keys are compatible, n x m pairs are enumerated. For example, with C_{Maj} and A_{min} as compatible chords and C_{Maj} and G_{Maj} as compatible keys, the chord candidates enumerated would be (C_{Maj}, C_{Maj}), (C_{Maj}, G_{Maj}), (A_{min}, C_{Maj}) and (A_{min}, G_{Maj}). If no compatible chord can be enumerated for a given note segment, the hypotheses we choose to set are the previous note segment chord candidates combined with the compatible keys. This is to keep taking into account key detection, even if no chord is compatible. If no compatible key can be built, the hypotheses we choose to set are the same ones as for the previous note segment. Both these choices may be justified by the high probability

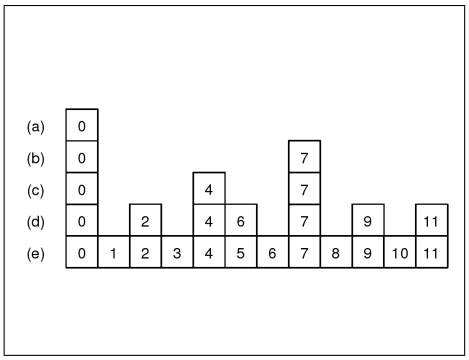

Figure 3. The basic space of the C_{Maj} chord in the C_{Maj} key. Levels (a) to (e) are respectively chromatic, diatonic, triadic, fifths and root levels.

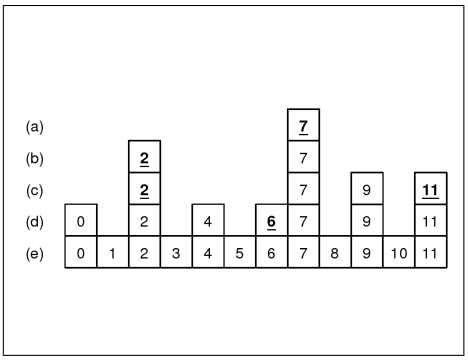

Figure 4. $\delta((C_{Maj},C_{Maj}) \rightarrow (G_{Maj},G_{Maj})) = i+j+k = 1+1+5 = 7$. The underlined pitches are the non-common pitches.

of having two consecutive note segments being part of the same chord. Finally, the chord output only mentions a root and a mode (major or minor). In other words, even if the type of a chord is supported by our system, only the corresponding mode is taken into account. For example, if a chord is detected as a C^7, it is handled by our system as C_{Maj}. We thus choose to focus on the root and the mode for now, the evaluation of chord types being part of a future work.

2.2.4. Chord Transition Cost

Once the chord candidates are enumerated for two consecutive note segments, an edge is built from each of the first segment's chord candidates to each of the second segment's. This edge is weighted by a transition cost between the two chord candidates. This transition cost must take into account both the different compatible chords, and the different compatible keys.

We thus choose to use Lerdahl's distance [**?**] as transition cost. This distance is based on the notion of basic space. Lerdahl defines the basic space of a given chord in a given key as the geometrical superposition of:

a the chromatic pitches of the given key (chromatic level),

b the diatonic pitches of the given key (diatonic level),

c the triad pitches of the given chord (triadic level),

d the root and dominant of the given chord (fifths level),

e the root of the given chord (root level).

Figure 3 shows the basic space of the C_{Maj} chord in the C_{Maj} key. If (C_x,K_x) represents the chord C_x in the key K_x, Lerdahl defines the transition cost from $x = (C_x,K_x)$ to $y = (C_y,K_y)$ as follows:

$$\delta(x \rightarrow y) = i+j+k$$

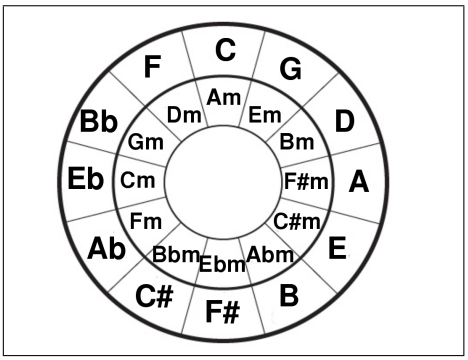

Figure 5. The circle of fifths.

where i is the distance between K_x and K_y in the circle of fifths (Figure 5), j is the distance between C_x and C_y in the circle of fifths and k is the number of non-common pitch classes in the basic space of y compared to those in the basic space of x.

The distance thus provides a integer cost from 0 to 13, and is completely adequate for a transition cost in the proposed method, since both compatible chords and keys are involved in the cost computation. A calculation of chord transition is illustrated in Figure 4, from $x = (C_{Maj},C_{Maj})$ to $y = (G_{Maj},G_{Maj})$. Here, i=j=1 because 1 step is needed to go from C_{Maj} to G_{Maj} in the circle of fifths. k=5 is the number of non-common pitches belonging to the basic space of of y compared to those in the basic space of x (underlined in the Figure). The distance is therefore 1+1+5=7.

2.3. Dynamic process

Once the graph between all the chord candidates is formed, the best path has to be found. This task is achieved by dynamic programming [**?**]. In the graph, from left to right, only one edge to each chord candidate is preserved. Several ways to select this edge can be considered. After experiments, we choose to preserve the edge minimizing the cost

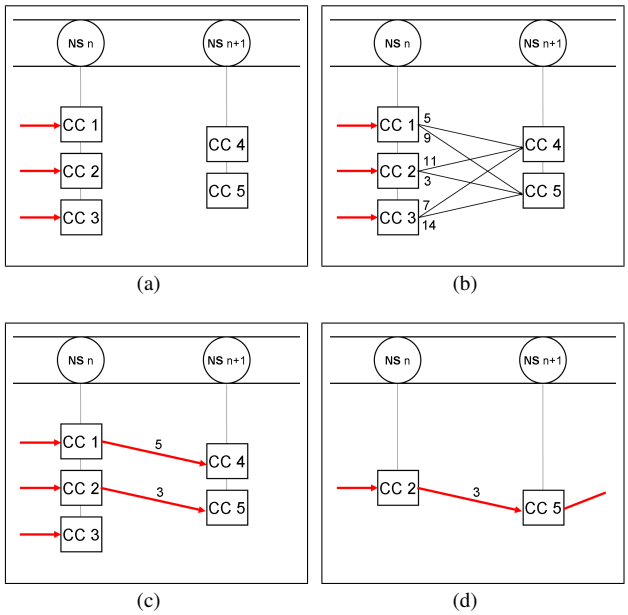

Figure 6. Illustration of the overall process. Graph construction between two consecutive note segments (NS): the possible chord candidates (CC) are listed (**a**), and linked with weighted edges (**b**). Dynamic process: only one edge to each chord candidate is preserved (**c**). In the end, the best path is selected (**d**).

to each candidate, as illustrated in Figure 6.c. Other possibilities must be explored in a future work, like preserving the edge minimizing the total sum of costs along the path to the candidate.

The number of final paths is the number of chord candidates for the last note segment. The final path minimizing its total cost sum is then outputted by the program.

2.4. Global computation

The overall process is illustrated in Figure 6.The construction of the graph is processed in two steps. First, chord candidates for each note segments are determined (Figure 6.a). Costs are then computed between each candidates for the latest note segments, and each chord candidates for the first one (Figure 6.b). After that, the dynamic process takes place: only one edge to a given chord candidate is preserved (Figure 6.c). Finally, the path minimizing its total cost sum is selected (Figure 6.d).

An example of the overall process computation on a musical excerpt is detailed in Figure 7. For each note segment, a chord is computed and the consecutive segments having the same chord identification are then merged, in order to form the boundaries of each chord. In this example, the detected chords are Eb_{Maj} for the first two beats, and Ab_{Maj} for the last two.

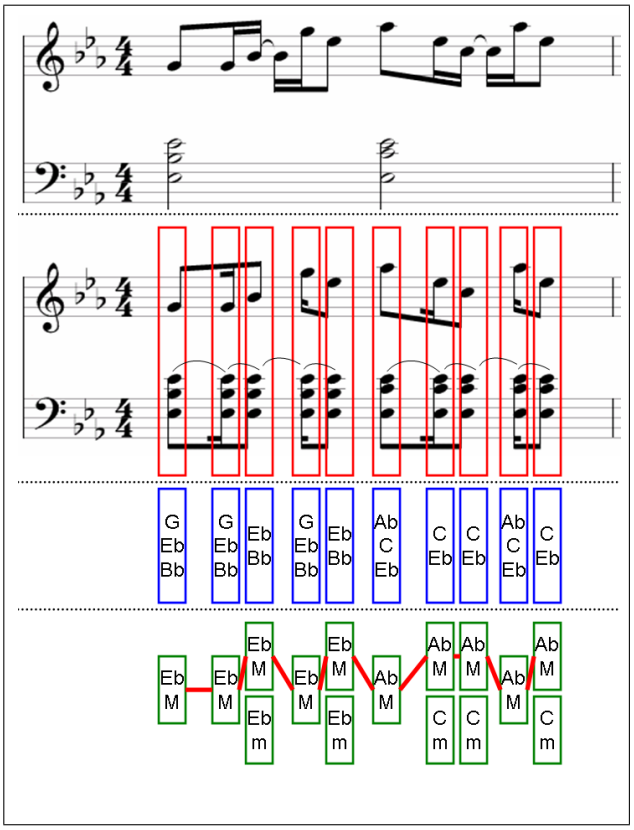

Figure 7. The first bar of *Your Song*, by Elton John, analyzed by our system. Above is the original score. One step below, the score with the homorhythmic transformation. One step below, the note segments. One step below, the chord candidates. For reading purposes, only the chord candidates sharing the same key (here, Eb_{Maj}) are shown. The best path is formed by the chord linked to each others.