CHORD DETECTION USING CHROMAGRAM OPTIMIZED BY EXTRACTING ADDITIONAL FEATURES

Jean-Baptiste Rolland
Steinberg Media Technologies GmbH
jb.rolland@steinberg.de

ABSTRACT

This paper presents some concepts regarding the optimization of chord detection algorithms based on chromagrams. The main goal of chord detection is to transcribe an audio recording of a piece of harmonic music into a musical score containing chord notations. Within a piece of music, a chord is very rarely alone, and a common failing of chord detection using chromagrams is to consider each chord individually and as isolated information. A piece of music is usually composed of connected and evolving structures, following specific musical rules and conventions: in this context, the issue is not only to detect separate chords, but use the musical context and accepted musical practice to improve the overall result, leading to a reasonable chord progression. Taken alone, each chord detection, structure analysis and appraisal of musical key, can be difficult problems to solve, but this paper will show how the differing analytical methods can interact with one another to optimize a chromagram-based chord detection algorithm.

1. INTRODUCTION

The field of automatic metadata extraction from music poses many different problems of detections. The specific problem of chord detection can be solved in many different ways. The most famous one seems to be the matching of chroma-vectors according to [1]. This simple technique gives good results with simple situations. But when the bands get bigger, when the sound is more processed, we must find a way to improve results keeping that information.

Before researching for completely new methods, we should understand all boundary conditions. That is why the following algorithm will keep the base of chroma-vector extraction, and optimize the results not seeing the problem as a local problem, but as a global problem.

If we consider the problem of chord detection in the specific situation of a song, we have a lot more information than only a small audio segment; we have a list of chords. We can be sure this list is not composed of random chords. It is likely written by humans following rules and fashions. Music theory explains the constraints. So why not use them to help us with our problem?

2. PERFORMANCE EVALUATION

The goal of an optimization is to increase the number of good results. But this optimization can have side effects we have to determine to accept it or not. This is why an evaluation is needed to understand how an optimization will affect the result.

All the results presented in this paper are processed by comparing the generated chord list of our algorithm with a reference annotation dataset. The reference dataset we used is The Beatles 180 songs. This famous dataset follows the annotation is defined by [3] and was introduced by Harte's PhD [4].

To determine the performances results, we have to compare our results with a reference using a metric. Of course, the first metric would be to evaluate the number of good results by exact matching of chords. According to [2] the chord symbol recall (CSR) is a good metric to evaluate performances of good results.

But the mentioned metric does not provide any information about errors. In chord detection, errors do not always have the same importance. That is why others metrics were used, such as the chord match recall (CMR).

The second metric is processed the same way as CSR but is not using the same distance. CSR uses the distance of 0 if two chords are exactly the same, and the distance of 1 if the symbols are different. In the case of CMR, the distance is the matching percentage of chords vectors. A chord vector is composed of 12 binaries values corresponding to basic chroma-vector of a chord. Some examples of template chord vectors are available Table 3 for each mode.

For two chord vectors named u and v, the CMR is defined by the given correlation coefficient formula (1):

$$CMR = \frac{\vec{u} \cdot \vec{v}}{\left\|\vec{u}\right\| \cdot \left\|\vec{v}\right\|} \qquad (1)$$

The distance between two identical chords is still 0, but if the chords are different, the value is between 0 and 1. The result is calculated by processing the normalized correlation of chord templates. Examples of results are a distance of 0.33 for C and Amin; 0.66 for C and F; 0.85 for C and C7.

The CMR provides more information about mistakes and is not directly correlated to CSR. An optimization is judged to be good when the CSR increases without decreasing CMR. All depends on the goal: more good results, or less important mistakes.

3. OVERVIEW OF THE ALGORITHM

The algorithm works in a classic way, related to what we find in the field of computer vision, or biometric identification and authentication. The first step is to extract features from the audio song, in which chromagrams are calculated with various different temporal window sizes and offsets, in order to extract as much information as needed. The next step matches the chroma-vectors with models to calculate chord probabilities. Then we use a statistic analysis to give a meaning to all of the extracted features and probabilities. The final step is to determine what the result is according to the available information.

The presented algorithm uses a chord positioning algorithm to detect the position of chords in a song. This chord positioning algorithm utilizes a beat tracking algorithm and a state-of-the-art method to merge the beats.

The chord detection algorithm can be improved by all the optimization based on other features extraction from a song. Other famous processes such as structure detection, key detection, and beat detection can be seen as related processes in a way. Linking all these processes improves the results of all improve all individuals. We will see how in the following sections.

4. EXTRACTING THE CHROMA FEATURES

Before starting the extraction, a simple audio file is normalized and mixed down to make it as independent as possible from the original source. Then, the feature extraction begins with spectrum processing and chroma-vector extraction.

4.1 Optimization of the Spectrum

Before computing a spectrum, we apply a temporal window to the signal. The quality of final performances will depend on the shape of the window. Some normalized performance results are available in Table 1. We can see that Kaiser or Bartlett gives the best results. All depends on what measure we prefer to use.

Name	CSR	CMR
Nothing	61%	38%
Hamming	81%	90%
Bartlett	86%	100%
Hann	50%	58%
Blackman	0%	0%
Kaiser α=2,5	100%	87%

Table 1. Normalized performance results for windows.

The next step is to calculate the spectrum in a classic way using the Fast Fourier Transform (FFT). The spectrum allows us to see some chords, but it is dangerous to use it directly. A chord is composed of multiple harmonic sounds. This means that the spectrum will contain multiple fundamentals and formants mixed together. More generally, some inharmonic sounds can also be mixed. In pop music, drum and synths that produces such sounds are most common and can really change the spectrum. That is why we considered two ways to improve the spectrum before reading it.

The first process is de-noising. This process cleans the spectrum by removing small and fast variation of frequency magnitude. To do this, we compute a spectral envelope reference with the spectrum using a low-pass filter. Then, we keep only what is over this envelope.

The second processing is whitening. The goal is to make the spectrum looking like a white noise, by improving the magnitude values when needed. The processing simply consists to calculate the same spectral envelope as the Denoiser, but now dividing the entire spectrum by it.

Some normalized performances results are available Table 2.

	With Denoiser	Without Denoiser
With Whitening	87%	100%
Without Whitening	26%	0%

Table 2. Normalized CMR performances results with spectrum post-processing

Some other techniques like the Harmonic Product Spectrum (HPS) can be used too. All depend on what kind of source we have. See the paper [6] for more information about other possible spectrum optimizations.

4.2 Chroma-Vector Extraction

After the spectrum processing, it is time to compute the chroma-vectors. The chroma-vector is a 12 energy values vector corresponding to the 12 possible pitches (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). Each value is computed by doing the sum of all spectrum magnitudes corresponding to all different notes. This way, we are already able to read easily some chords when the sound is clean. See figure 1 for an example of chroma-vectors we can get from the C Major chord.

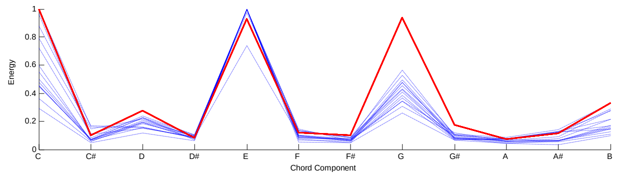

Figure 1. Chroma-vectors calculated for the reference chord of C Major.

4.3 Solving Tuning issues

One problem that has to be solved now is the tuning issue. Indeed, all records do not tune their A4 to 440Hz. In classical music, the tune wasn't always this one. It evolved for centuries between 420 and 450 Hz that can be really close to G # at 415Hz.

The solution here is to consider two possibilities: the normal way of a 440Hz tune, and the other where all is tuned quarter-ton lower. In this case, the result will always be good, or always a ton lower but will not switch between lower and higher during the song.

To determine which one between the tuned and the normal chroma-vector we use, we just compare the global dynamics of the vectors. The more dynamics we have, the more we will be able to determine which chord it is.

5. STATISTIC ANALYSIS

Now we have our clean spectrums, our chroma-vectors, we just need to give them a sense. The first step is to match the chroma-vector with all possible chords to process a matching score for all chords. After that, we determine the bass pitch from the spectrum. This is going to give us a first probability for each chord to be the right one.

5.1 Determine the chord probabilities

To determine the score of matching of each reference chord with the chroma-vector, we process the scalar product between the chroma-vector and each shifted chord mode reference vectors for each fundamental. The results can be stored in a matrix 12*(Number of Modes).

The reference chord vector is like a chroma-vector, but only with the main components of the chords. For example, Major is (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0). For less common chords, the result is multiplied by a coefficient to reduce the score. For example, bests results are get if we multiply the score of Diminished chords by 0.9. The reference used is presented Table 3.

Chord	Template	Coefficient
Major	1 0 0 0 1 0 0 1 0 0 0 0	100%
minor	1 0 0 1 0 0 0 1 0 0 0 0	100%
Sus4	1 0 0 0 0 1 0 1 0 0 0 0	90%
Sus2	1 0 1 0 0 0 0 1 0 0 0 0	90%
Dim	1 0 0 1 0 0 1 0 0 0 0 0	100%
Aug	1 0 0 0 1 0 0 0 1 0 0 0	100%
Maj7	1 0 0 0 1 0 0 1 0 0 0 1	70%
7	1 0 0 0 1 0 0 1 0 0 1 0	70%
min7	1 0 0 1 0 0 0 1 0 0 1 0	70%
min6	1 0 0 1 0 0 0 1 0 1 0 0	70%
min7/b5	1 0 0 1 0 0 1 0 0 0 1 0	70%
minj7	1 0 0 1 0 0 0 1 0 0 0 1	70%
Sus4/7	1 0 0 0 0 1 0 1 0 0 0 1	70%
Sus2/7	1 0 1 0 0 0 0 1 0 0 0 1	70%
Dim6	1 0 0 1 0 0 1 0 0 1 0 0	70%
Maj7/#5	1 0 0 0 1 0 0 0 1 0 0 1	70%

Table 3. Modes templates for chords matching

In our case, we decided to detect 8 or 16 different chord modes. The first eight are the most commons in pop song (Major, minor, Diminished, Augmented, 7th, 7th Major, minor 7th and minor 7th Major). The others chords are less common but can be important in some kinds of music (Suspended 2 or 4 with 7th or not, Minor 6th, Minor 7th/b5, Diminished 6th and Major 7th/#5).

5.2 Neural Networks

Another method to give a meaning of a chroma-vector is to use a neural network. This kind of automatic learning process always requires a reference dataset for the learning period. A training dataset of 4608 audio chords was generated by Halion 4, containing 28 different common instruments playing in 16 different modes. These data were used to train many perceptron corresponding to each different mode.

After the learning process, the network was able to answer with a very low False Rejection Rate (FRR) for solo instruments. The most the chord was specific, the lowest the FRR was. On the other hand, the False Acceptance Rate (FAR) was too high to believe the network in case of acceptance. It can be explained by the small differences between chords chroma-vectors and the close link between some chords.

We concluded that the use of neural networks was good to help us to validate chord detection, especially for uncommon chords. But it does not provide any information about how sure the answer is. In consequence, this improvement could be only one element involved in our analysis to help us for our final decision.

5.3 Determine the bass

The most important data to improve the chord probability is to determine the value of the bass. In pop music, the bass is most of the time related to the chord. For example, the most obvious way would be to think that the bass can be the fundamental. So if we detect a C Major chord and a bass playing a C, the probability of C Major to be the right one is higher. This is not working if the bass is changing all the time (walking bass is a good example). That is why we need a special strategy to determine the main bass of the chord.

Determining the bass with a spectrum looks simple, because it correspond to the first big pitch magnitude. The question is to determine how big it is. Goods results can be gotten with the first magnitude over 66% of the maximum magnitude. But this will not work with all instruments and all songs.

Some instruments have other pitch frequencies before their fundamental. One solution could be to search for harmonics to validate the fundamental value. In our case, we just determined the best practice by combining the bass determination to the chord detection evaluation.

5.4 Sub-windows strategy

To improve the chord probabilities and the bass detection, we can split our chord signal into sub-windows and process 4.1 and 4.2 on it. The goal after that is to merge results to get a more stable answer on chord probabilities, and bass probability.

To split our signal, we simply divide our windows into smaller overlapping sub-windows. In our case, we try to split it in regular sub-windows with the size of 0.2 seconds, and complete them with a minimal zero-padding to have a size corresponding to a power of two if needed.

It is important to determine a weighting strategy to merge the results to increase the importance of sub-windows in the beginning of the chord for example.

There are other possibilities to determine the position of the sub-windows like doing local hit-point detection. The local hit-point detection can be done by processing the difference between a fast and a slow integrator, and searching for local maximums of it. This way, we are able to adapt dynamically the position of sub-windows, removing partially the effect of the drums and other rhythmic noises by concentrating it to fewer sub-windows. But this solution can create wrong results for example in the case of arpeggios.

This statistic analysis gives us a first probability matrix for each chord to be the real one. But it was considering the chord as local information. However, most of the time a chord isn't alone. And the other chords probabilities are other information we can use globally to improve the detection.

6. KEY DETECTION

As explained in the previous part, when we are looking for a chord in a part of the song, we split this part in sub-windows. Each sub-window gives a result of chord probabilities. One good optimization to eliminate some bad estimation would be to use the problem of key detection.

Indeed, the problem of key detection can be solved with some methods like described in [7]. Using this method, we detect the local key around the chord we are processing to get more information about chord probabilities. Indeed, some chords are more or less possible depending to the tonality. A good way to process chord probability is to match the chord template vector with the tonality template vector. The tonality template vector contains all possible notes of the tonality.

For example, the C tonality has the following template: (1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1). A basic normalized scalar product is used to determine the relation between a chord and a tonality. So, in the C tonality, the probability to have a Dm is 100%; D is 66%; C7 is 75% and so on.

7. CHORD PROGRESS PROBABILITY

Two of mains elements in instrumentals pop songs are the melody, and the chord progressions. The chords characterize the song with its evolution. What people usually mainly remain is how chords evolve between them, not the precise note value of the key or the precise speed they evolve. That is why music theory brings some rules about what people could usually do with the chord progression. This way, we are able to compute some references chord progression probabilities, and use them. One other way to compute those probabilities is to learn progression statistics from a dataset. This idea consists in a way to see the problem of chord progression as a Markov chain. That what are doing some studies like [8] using additional features.

In our case, we focused to the most general way that was to use music theory. The system used is a derived version of the Steinberg Chord Assistant present inside Cubase 7.5. Processing the probabilities of chord progression was done using some references full cadence models (for example I-IV-V-I). At first, each cadence defines some progression probabilities considering all possible positions. In a second way, we consider all progressions with the relative substitutes to increase the possibilities to switch between modes. All those rules are extracted from music theory, and succeeded to give us more information about the next chord probabilities.

8. STRUCTURE ANALYSIS

At last but not least, the problem of structure detection is really interesting for chord detection from pop songs. Indeed, most of the errors of chord detection are dues to difficulties like a drum, or voice too present on the record. Most of pop songs are made of repeating parts like verses and refrains. Those parts are mostly made with the same chords. A small mistake of chord detection can be solved this way by associating all parts together, and analyzing what is abnormally different.

8.1 Detecting the structure

The first step in this optimization and the biggest one is to detect the song structure. Some technics like [9] and [10] gives good results, and shows that one of the most important common points between same parts is the spectrogram content. Using this information, a simple structure detection algorithm can be made using the already detected chords. This way, we do not search for structure directly inside the song, but inside the chord list.

The most important in the chord structure detection is to evaluate when a chord part is the same as another. To allow our detection to keep working with some random chord detection mistakes, we have to think about a metric for chord parts comparison. One simple solution is to process the percentage of time when the parts are transcript with the exact same chord symbols. It is an analog way of processing the CSR. One other solution is to use the CMR metric to process the matching score. Others metrics can be used, such as replacing the mode template of CMR by a mean of chords chroma-vectors. The question after that is to fix how combining it, and defining a threshold to determine how permissive we are.

One other choice we have to do concern the size of the parts we want to detect. All depends if we prefer to detect a structure with a few big parts, or a lot of small parts. In our case, we would prefer small parts to have more chord material for our following correction.

8.2 Quality evaluation of song structure

Before using a song structure, it is important to evaluate the quality of it. The reason is that the optimization has to work with all songs, including those without logic structure. The evaluation determines if it is pertinent to use the detected structure to correct the chord list. The decision is depending on the quantity of different parts, the percentage of the song covered by the structure, and the mean of distance between each parts iterations.

8.3 Chords corrections

After processing the structure of the song, it is easy to correct the possible errors of the chords detection with it. The strategy we used is to mean all parts iterations, and to choose the most common chords. If all the structures disagree with one particular chord, it can be better to not correct it and to go back to chord probabilities to determine the right solution.

9. CONCLUSIONS

After processing simple chord detection using classic technique related to chromagrams, we presented some ways to improve the results by using the solution of other related automatic detection problems. We get better results for chord detection by considering not only each individual chord, but by looking at all chords as a given piece of music. The algorithm optimization gives a lot more data that has to be merged in a statistical analysis. This way we can get a very general algorithm working for most situations including solo or multi instrumental songs. We evaluated the algorithm with the Beatles dataset, and get a CSR value of 73%, and a CMR of 89%. Even if the algorithm does not always find the correct chord symbols, the found chord will be coherent with the music.

10. ACKNOWLEDGEMENTS

This research was supported by Steinberg Media Technologies GmbH. Thanks to my supervisor Yvan Grabit (Head of Steinberg's Research Team) for his support and advices and Ralf Kuerschner (Head of Steinberg's Engineering) for his trust and the opportunity to work together.

11. REFERENCES

[1] K. Lee: "Automatic Chord Recognition from Audio Using Enhanced Pitch Class Profile" *Proceedings of the International Computer Music Conference*, New Orleans, USA, 2006.

[2] J. Pauwels and G. Peeters: "Evaluating Automatically Estimated Chord Sequences" *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Vancouver, Canada, 2013.

[3] C. Harte, M. Sandler, S. A. Abdallah, and E. Gómez: "Symbolic representation of musical chords: A proposed syntax for text annotations." *Proceedings of the 6th International Conference on Music Information Retrieval, ISMIR 2005*, London, UK, pages 66–71, 2005.

[4] C. Harte: "Towards Automatic Extraction of Harmony Information from Music Signals" PhD from the Department of Electronic Engineering, Queen Mary, University of London, UK, 2010.

[5] M. Goto, Y. Muraoka: "Real-time beat tracking for drumless audio signals: Chord change detection for musical decisions", *Speech Communication*, Vol. 27 No. 3-4, 311-335, 1999.

[6] A. Savard: "Overview of homophonic pitch detection algorithms" Schulich School of Music, McGill University, Montreal, Technical Report MUMT-612, 2006.

[7] T. Rocher, M. Robine, P. Hanna and L. Oudre: "Concurrent Estimation of Chords and Keys from Audio", *11th International Society for Music Information Retrieval Conference, ISMIR 2010*, LaBRI - University of Bordeaux, France, 2010.

[8] R. Chen, W. Shen, A. Srinivasamurthy, and P. Chordia: "Chord Recognition Using Duration-explicit Hidden Markov Models", *Proceedings of the 13th International Society for Music Information Retrieval Conference*, Porto, Portugal, 2012.

[9] J. Paulus, M. Müller and A. Klapuri: "State of the Art Report: Audio-Based Music Structure Analysis", *11th International Society for Music Information Retrieval Conference, ISMIR 2010*, Fraunhofer Institute for Integrated Circuits IIS, Germany, 2010.

[10] F. Kaiser and T. Sikora: "Music Structure Discovery in Popular Music using Non-negative Matrix Factorization", *11th International Society for Music Information Retrieval Conference, ISMIR* 2010, Communication Systems Group - Technische Universität Berlin, Germany, 2010.