AUTOMATIC X TRADITIONAL DESCRIPTOR EXTRACTION: THE CASE OF CHORD RECOGNITION

Giordano Cabral
LIP6 – Paris 6
8 Rue du Capitaine Scott
75018 Paris
Giordano.CABRAL@lip6.fr

François Pachet
Sony CSL Paris
6 Rue Amyot
75005 Paris
pachet@csl.sony.fr

Jean-Pierre Briot
LIP6 – Paris 6
8 Rue du Capitaine Scott
75018 Paris
Jean-Pierre.BRIOT@lip6.fr

ABSTRACT

Audio descriptor extraction is the activity of finding mathematical models which describe properties of the sound, requiring signal processing skills. The scientific literature presents a vast collection of descriptors (e.g. energy, tempo, tonality) each one representing a significant effort of research in finding an appropriate descriptor for a particular application. The Extractor Discovery System (EDS) [1] is a recent approach for the discovery of such descriptors, which aim is to extract them automatically. This system can be useful for both non experts – who can let the system work fully automatically – and experts – who can start the system with an initial solution expecting it to enhance their results. Nevertheless, EDS still needs to be massively tested. We consider that its comparison with the results of problems already studied would be very useful to validate it as an effective tool. This work intends to perform the first part of this validation, comparing the results from classic approaches with EDS results when operated by a completely naïve user building a guitar chord recognizer.

Keywords: descriptor extraction, chord recognition.

1 INTRODUCTION

Audio descriptors express by a mathematical formula a particular property of the sound. Such a property may be for example the tonality of a musical piece, the amount of energy in a given moment, or whether a song is instrumental or sung. Although the creation of each descriptor means a different study, the design of a descriptor extractor normally follows the process of combining the relevant characteristics of acoustic signals (features) using machine learning algorithms. These features are often low-level descriptors (LLD), and the task usually requires important signal processing knowledge.

Since last year, a heuristic-based approach became available through the Computer Science Lab of Sony in Paris, which developed the Extractor Discovery System (EDS). The system is based on genetic programming, and machine learning algorithms employed to automatically generate a descriptor from a database of sound files examples and their respective perceptive values. EDS can be used by non experts or expert users. Non experts can use it as a tool to extract descriptors, even with minimal or no knowledge at all in signal processing. For example, movie makers have created classifiers of sound samples to be used in their films (explosions, car breaks, etc.). Experts can use the system to improve their results, starting from their solution and then controlling and guiding EDS. For instance, the perceived intensity of music titles can be more precisely detected, taking as a starting point the mpeg7 audio features [2].

We are currently designing a guitar accompanier for "bossa nova" style. During the application development process, we ran into the problem of recognising a chord, which turns out to be a good opportunity to compare classical and EDS approaches. On the one hand, chord recognition is a well studied domain, with solid results that can be considered as reference. On the other hand, current techniques use background knowledge that EDS (initially) does not have (pitches, harmony). Good EDS results would indicate the capacity of the system to deal with real world musical description cases.

This paper presents a comparison between a standard technique to chord recognition (knn learner over pitch class profiles) and an EDS solution performed by an inexperienced, naïve user. In the next section, we introduce the chord recognition problem. In section 3 we explain the most widely used technique. In section 4 we examine EDS, how it works and how to use it. Section 5 details the experiment. Section 6 shows and discuss the results. Finally, we draw some conclusions and point future works.

2 CHORD RECOGNITION

The ability of recognizing chords is important for many applications, such as interactive musical systems, content-based musical information retrieval (finding particular examples, or themes, in large audio databases), and educational software. Chord recognition means the transcription of a sound into a chord, which can be classified according to different levels of precision, from a simple distinction between maj and min chords to a complex set of chord types (maj, min, 7th, dim, aug, etc).

Many works can be mentioned here as the state of the art. [4] and [5] automatically transcribes chords from a CD recorded song. [3] deals with a similar problem: estimating the tonality of a piece (which is analogous to the maj/min). In these cases and in most part of the literature the same core technique is used, even if some variations may appear during the implementation phase. This technique has been applied to our problem. We explain it in the next section.

3 PITCH CLASS PROFILE

Most part of the works involving harmonic content (chord recognition, chord segmentation, tonality estimation) uses a feature called Pitch Class Profile (PCP) [6]. PCPs are vectors of low-level instantaneous features, representing the intensity of each pitch of the tonal scale mapped to a single octave. This intensity can be calculated by the magnitude of the spectral peaks, or by summing the magnitudes of all frequency bins that are located within a certain frequency band. Each frequency band corresponds to a pitch, and may change to deal with differences in tuning and/or to gain in performance. The equivalent pitches from different octaves are summed, producing a vector of 12 values, consequentially unifying various dispositions of a single chord class.

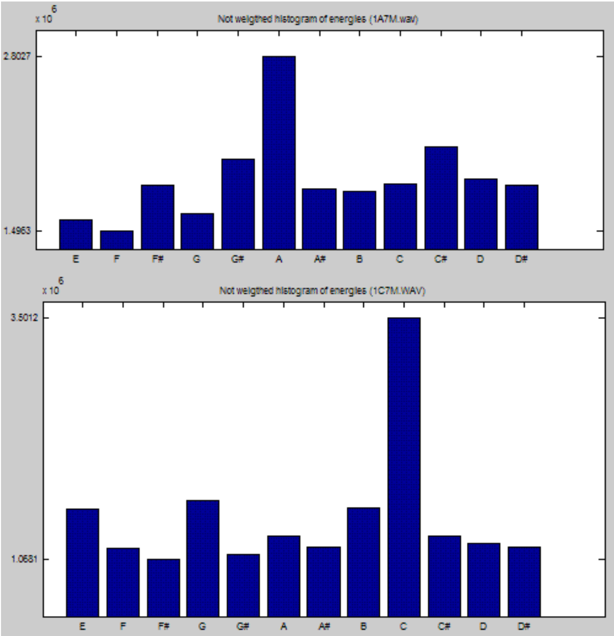
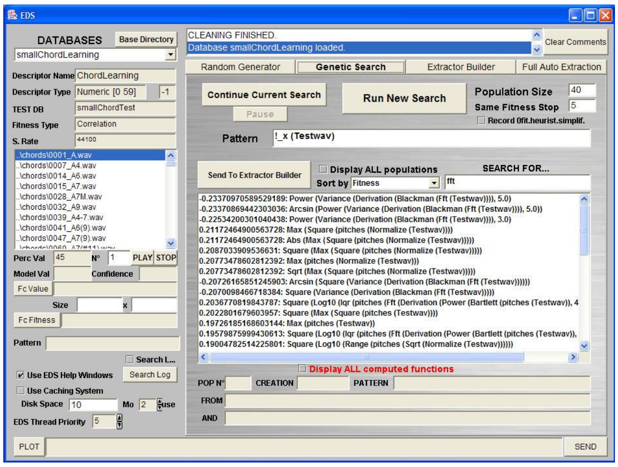


Figure 1. Examples of PCPs for a Amaj7 (above) and a Cmaj7 (below).

The idea is that the PCPs of a chord follow a pattern, and that pattern can be learned from examples. Machine learning (ML) [9] techniques are used to generalize a classification model from a given database of labelled examples, and then new examples can be automatically classified. The original PCP implementation from Fujishima used a KNN learner [9], and more recent works [3] successfully used other machine learning algorithms.

4 EDS

EDS (Extractor Discovery System), developed at Sony CSL, is a heuristic-based generic approach for automatically extracting high-level music descriptors from acoustic signals. EDS approach is based on Genetic Programming [11], used to build extraction functions as compositions of basic mathematical and signal processing operators. Given a database of audio signals with their associated perceptive values, EDS is capable to generalize a descriptor. Such descriptor is built by running a genetic search to find relevant signal processing features to match the description problem, and then machine learning algorithms to combine those features into a general descriptor model.



smallChordLearning

Descriptor Name: ChordLearning
Descriptor Type: Numeric [0 59] -1
TEST DB: smallChordTest
Fitness Type: Correlation
S. Rate: 44100

..\chords\0001_A.wav
..\chords\0007_A4.wav
..\chords\0014_A6.wav
..\chords\0015_A7.wav
..\chords\0028_A7M.wav
..\chords\0032_A9.wav
..\chords\0039_A4-7.wav
..\chords\0041_A6(9).wav
..\chords\0047_A7(9).wav

Perc Val: 45 N° 1
Model Val:
Confidence:
Size: x
Pattern:
□ Search L...
☑ Use EDS Help Windows
□ Use Caching System
Disk Space: 10 Mo 2 use
EDS Thread Priority: 5

CLEANING FINISHED.
Database smallChordLearning loaded.

Population Size: 40
Same Fitness Stop: 5
□ Record 0fit.heurist.simplif.
Pattern: !_x (Testwav)
□ Display ALL populations
Sort by: Fitness
SEARCH FOR...: fft

```
-0.23370970589529189: Power (Variance (Derivation (Blackman (Fft (Testwav)))), 5.0)
-0.23370869442303036: Arcsin (Power (Variance (Derivation (Blackman (Fft (Testwav)))), 5.0))
-0.22534200301040438: Power (Variance (Derivation (Blackman (Fft (Testwav)))), 3.0)
0.21172464900563728: Max (Square (pitches (Normalize (Testwav))))
0.21172464900563728: Abs (Max (Square (pitches (Normalize (Testwav)))))
0.2087033909536631: Square (Max (Square (pitches (Normalize (Testwav)))))
0.20773478602812392: Max (pitches (Normalize (Testwav)))
0.20773478602812392: Sqrt (Max (Square (pitches (Normalize (Testwav)))))
-0.20726165851245903: Arcsin (Square (Variance (Derivation (Blackman (Fft (Testwav))))))
-0.2070098466718384: Square (Variance (Derivation (Blackman (Fft (Testwav)))))
0.2036770819843787: Square (Log10 (Iqr (pitches (Fft (Derivation (Power (Bartlett (pitches (Testwav)), 4
0.2022801679603957: Square (Max (Square (pitches (Testwav))))
0.19726185168603144: Max (pitches (Testwav))
0.19579875999430613: Square (Log10 (Iqr (pitches (Fft (Derivation (Power (Bartlett (pitches (Testwav)),
0.19004782514225801: Square (Log10 (Range (pitches (Sqrt (Normalize (Testwav))))))
```

□ Display ALL computed functions
POP N°:
CREATION:
PATTERN:
FROM:
AND:

Figure 2. EDS main interface.

The genetic search performed by the system is intended to generate functions that may eventually be relevant to the problem. The best functions in a population are selected and iteratively transformed (by means of reproduction, i.e., constant variations, mutations, and/or crossover), always respecting the pattern chosen by the user. The default pattern is *!_x(Testwav)*, which means a function presenting any number of operations but a single value as result. The populations of functions reproduce until no improvement is found.

At this point, the best functions are selected to be combined. This selection can be made both manually or automatically. For example, given a database of audio files labeled as 'voice'/'instrumental', kept the default pattern, these are some possible functions that might be selected by the system:

```
Log10    (Range  (Derivation    (Sqrt    (Blackman
(MelBands  (Testwav, 24.0))))))

Square   (Log10   (Mean   (Min   (Fft
(Split (Testwav, 4009))))))
```

Figure 3. Some possible EDS functions for the default pattern.

The final step in the extraction process is to choose and compute a model (linear regression, model trees, knn, locally weighted regression, neural networks, etc.). Alternatively, the user can choose the option *test and optimize all classification methods*. As the output, EDS creates an executable file, which classifies an audio file passed as argument.

5 BOSSA NOVA GUITAR CHORDS

Our final goal is to create a guitar accompanier in Brazilian "bossa nova" style; consequently our chord recogniser has examples of chords played with nylon guitar. The data was taken from D'accord Guitar Chord Database [10], a guitar midi based chord database. The purpose of using it was the richness of the symbolic information present (chord root, type, set of notes, position, fingers, etc.), which was very useful for labelling the data and validating the results. Each midi chord was rendered into a wav file using Timidity++ [13] and a free nylon guitar patch, and the EDS database was created according to the information found in D'accord Guitar database. Even though a midi-based database may lead to distortions in the results, we judge that the comparison between approaches is still valid.

5.1 Chord Classes

We tested the solutions with some different datasets, reflecting the variety of nuances that chord recognition may show:

AMaj/Min –classifies between major and minor chords, given the root is A. 101 samples, 2 classes.

Chord Type, fixed root – classifies among major, minor, seventh, minor seventh and diminished chords, given it is a fixed root (A or C). 262 samples, 5 classes,

Chord Recognition – classifies major, minor, seventh, minor seventh and diminished chords, in any root. 1885 samples, 60 classes.

80% of each database is settled on as the training dataset and 20% as the testing dataset.

5.2 Pitch Class Profile

In our implementation of the pitch class profile, frequency to pitch mapping is achieved using the logarithmic characteristic of the equal temperament scale.

The intensity of each pitch is computed by summing the magnitude of all frequency bins that correspond to a particular pitch class. The same computation is applied to a white noise and the result is used to normalize the other PCPs.

For the *chord recognition* database, PCPs were rotated, meaning that each PCP was computed 12 times, one time for each possible rotation (for instance, a Bm is equivalent to a Am rotated twice).

After the PCP extraction, several machine learning algorithms could be applied. We implemented 2 simple solutions. The first one calculates a default, or a template PCP to each chord class. Then, the PCP of a new example can be matched up to the template PCP, and the most similar one is retrieved as the chord.

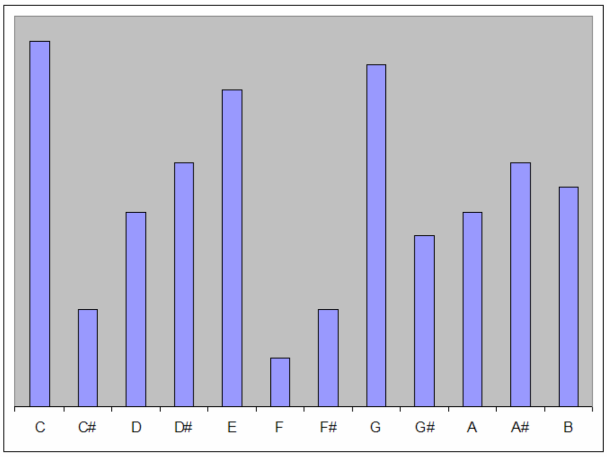


Figure 4. Example of a template PCP for a C chord class.

The second one uses the k-nearest neighbours algorithm (KNN), with maximum of 3 neighbours. KNNs have been used since the original PCP implementation and have proved to be at least one of the best learning algorithms for this case [3].

5.3 EDS

The same databases were loaded in EDS. We ran a fully automated extraction, keeping all default values. The system generated the descriptor without any help from the user, obtaining the results we call EDS Naïve, because they correspond to the results that a naïve user would achieve.

6 RESULTS AND DISCUSSION

The results achieved by us are presented in the table above. Rows represent the different databases. Columns represent the different learning techniques. The percent values indicate the number of correctly classified instances over the total number of examples in the testing database.

Table 1. Percentage of correctly classified instances for the different databases using the studied approaches.

Approach Database	PCP Template	KNN	EDS
Maj/Min (fixed root)	100%	100%	90.91%
Chord Type (fixed root)	89%	90.62%	87.5%
Chord Recognition	53.85%	63.93%	40.31%

As we can see, EDS gets really close to classical approaches when the root is known, but disappoints when the whole problem is presented. It seems that a combination of low level functions is capable of finding different patterns in the same root, but the current palette of signal processing functions in EDS is not sufficient to generalize harmonic information.

6.1 Case 1: Major/Minor classifier, fixed root.

Figure 5 shows the selected features for the Amaj/min database. The best model obtained was a KNN of 1 nearest neighbour, equally weighted, absolute error (see [9] for details). The descriptor reached 90.91% of the performance of the best traditional classifier.

```
  EDS1: Power (Log10 (Abs (Range (Integration
(Square (Mean (FilterBank (Normalize (Testwav),
5.0))))))), -1.0)

  EDS2: Power (Log10 (Abs (Range (Sqrt (Bart-
lett (Mean (FilterBank (Normalize (Testwav),
9.0))))))), -1.0)

  EDS3: Sqrt (Range (Integration (Hanning
(Square (Mean (Split (Testwav, 3736.0)))))))

  EDS4: Arcsin (Sqrt (Range (Integration (Mean
(Split (Normalize (Testwav), 5862.0))))))

  EDS5: Log10 (Variance (Integration (Bartlett
(Mean (FilterBank (Normalize (Testwav),
5.0))))))

  EDS6: Power (Log10 (Abs (Range (Integration
(Square (Sum (FilterBank (Normalize (Testwav),
9.0))))))), -1.0)

  EDS7: Square (Log10 (Abs (Mean (Normalize
(Integration (Normalize (Testwav)))))))

  EDS8: Arcsin (Sqrt (Range (Integration (Mean
(Split (Normalize (Testwav), 8913.0))))))

  EDS9: Power (Log10 (Abs (Range (Sqrt (Bart-
lett (Mean (FilterBank (Normalize (Testwav),
3.0))))))), -1.0)
```

Figure 5. Selected features for the Amaj/min chord recogniser.

6.2 Case 2: Chord Type Recognition, fixed root.

Figure 6 shows the selected features for the chord type database. The best model obtained was a GMM of 14 gaussians and 500 iterations (see [9] for details). The descriptor reached 96,56% of the performance of the best traditional classifier.

```
  EDS1: Log10 (Abs (RHF (Sqrt (Integration
(Integration (Normalize (Testwav)))))))

  EDS2: Mean (Sum (SplitOverlap (Sum (Bartlett
(Split (Testwav, 1394.0))), 4451.0,
0.5379660839449434)))

  EDS3: Power (Log10 (Abs (RHF (Normalize (In-
tegration (Integration (Normalize (Test-
wav))))))), 6.0)

  EDS4: Power (Log10 (RHF (Testwav)), 3.0)

  EDS5: Power (Mean (Sum (SplitOverlap (Sum
(Bartlett (Split (Testwav, 4451.0))), 4451.0,
0.5379660839449434))), 3.0)
```

Figure 6. Selected features for the Chord Type recogniser.

6.3 Case 3: Chord Recognition.

Figure 7 shows some of the selected features for the chord recognition database. The best model obtained was a KNN of 4 nearest neighbours, weighted by the inverse of the distance (see [9] for details). The descriptor reached 63,05% of the performance of the best traditional classifier. It is important to notice that 40,31 % is not necessarily a bad result, since we have 60 possible classes. In fact, 27,63% of the wrongly classified instances were due to mistakes between relative majors and minors (e.g; C and Am); 40,78% due to other usual mistakes (e.g. C and C7; C° and Eb°; C and G); only 31,57% were caused by unexpected mistakes. Despite these remarks, the comparative results are significantly worse than the previous ones.

```
  EDS1: Square (Log10 (Abs (Sum (SpectralFlat-
ness (Integration (Split (Testwav, 291.0)))))))

  EDS4: Power (Log10 (Abs (Iqr (SpectralFlat-
ness (Integration (Split (Testwav, 424.0)))))),
-1.0)

  EDS9: Sum (SpectralRolloff (Integration
(Hamming (Split (Testwav, 4525.0)))))

  EDS10: Power (Log10 (Abs (Median (Spec-
tralFlatness (Integration (SplitOverlap (Test-
wav, 5638.0, 0.7366433546185794)))))), -1.0)

  EDS12: Log10 (Sum (MelBands (Normalize
(Testwav), 7.0)))

  EDS13: Power (Median (Normalize (Testwav)),
5.0)

  EDS14: Rms (Range (Hann (Split (Testwav,
9336.0))))

  EDS15: Power (Median (Median (Split (Sqrt
(Iqr (Hamming (Split (Testwav, 2558.0)))),
4352.0))), 1.5)

  EDS17: Power (HFC (Power (Correlation (Nor-
malize (Testwav), Testwav), 4.0)), -2.0)

  EDS18: Square (Log10 (Variance (Square
(Range (Mfcc (Square (Hamming (Split (Testwav,
9415.0))), 2.0))))))

  EDS19: Variance (Abs (Median (Hann (Filter-
Bank (Peaks (Normalize (Testwav)), 5.0)))))

  EDS21: MaxPos (Sqrt (Normalize (Testwav)))

  EDS22: Power (Log10 (Abs (Iqr (SpectralFlat-
ness (Integration (Split (Testwav,
4542.0)))))), -1.0)
```

Figure 7. Some of the selected features for the chord recogniser.

6.4 Other cases

We also compared the three approaches on other databases, as we can see in the table 2. *MajMinA* is the major/minor classifier, root fixed to A. *ChordA* is the chord type recogniser, root fixed to A. *ChordC* is the chord type recogniser, root fixed to C. *RealChordC* is the same chord type recogniser in C, but the testing dataset is composed by real audio recordings (samples of less than 1 second of chords played in a nylon guitar), instead of midi rendered audio. Curiously, in this case, the EDS solution worked better than the traditional one (probably due to an alteration in tuning in the recorded audio). *Chord* is the chord recognition database. SmallChord is a smaller dataset (300 examples) for the same problem. Notice that in this case EDS outperformed KNN and

PCP Template. In fact, the EDS solution does not improve very much when passing from 300 to 1885 examples (from 38,64% to 40,31%), while the KNN solution goes from 44% to 63,93%. Finally, *RealChord* has the same training set from the *Chord* database, but is tested with real recorded audio.

Table 2. Comparison between the performance of the EDS and the best traditional classifier for a larger group of databases. Comparative performance = eds performance / traditional technique performance.

DB NAME	Comparative Performance
MajMinA	91,00%
ChordA	94,38%
ChordC	96,56%
RealChordC	116,66%
Chord	63,05%
SmallChord	87,82%
RealChord	55,16%

The results from these databases confirm the trend of the previous scenario. The reading of the results indicates that the effectiveness of the EDS fully automated descriptor extraction depends on the domain it is applied to. Even admitting that EDS (in its current state) is only partially suited to non expert users, we must take into account that EDS currently uses a limited palette of signal processing functions, which is being progressively enhanced. Since EDS didn't have any information about tonal harmony, it was already expected that it would not reach the best results. Even though, the results obtained by the chord recogniser with a fixed root show the power of the tool.

7 CONCLUSION AND FUTURE WORKS

In this paper we compared the performance of a standard chord recognition technique and the EDS approach. The chord recognition was specifically related to nylon guitar samples, since we intend to apply the solution to a Brazilian style guitar accompanier. The standard technique was the Pitch Class Profiles, in which frequency intensities are mapped to the twelve semitone pitch classes, and then uses KNN classification to chord templates. EDS is an automatic descriptor extractor system that can be employed even if the user does not have knowledge about signal processing. It was operated in a completely naïve way so that the solution and the results would be similar to those obtained by a non expert user.

The statistical results reveal a slight deficit of EDS for a fixed root, and a greater gap when the root is not known a priori, showing its dependency on primary operators. An initial improvement is logically the increase of the palette of functions. Currently, we are implementing tonal harmony operators such as chroma and pitchBands, which we suppose will provide much better results. Additionally, as the genetic search in EDS is indeed an optimisation algorithm, if the user starts from a good solution, it will be expected that the algorithm makes it even better. The user can also guide the function generation process, via more specific patterns and heuristics.

With these actions, we intend to perform the second part of the comparison we started in this paper – between the traditional techniques and EDS operated by a signal processing expert.

ACKNOWLEDGEMENTS

We would like to thank all the team at Sony CSL Paris, particularly Anthony Beurivé, Jean-Julien Aucouturier and Aymeric Zils for their support and assistance with EDS; and a special thanks to Tristan Jehan for his help in the conception and implementation of the algorithms.

REFERENCES

[1] Pachet, F. and Zils, A. "Automatic Extraction of Music Descriptors from Acoustic Signals", Proceedings of Fifth International Conference on Music Information Retrieval (ISMIR04), Barcelona, 2004.

[2] Zils, A. & Pachet, F. "Extracting Automatically the Perceived Intensity of Music Titles", Proceedings of the 6th COST-G6 Conference on Digital Audio Effects (DAFX03), 2003.

[3] Gómez, E. and Herrera, P. "Estimating the tonality of polyphonic audio files: cognitive versus machine learning modelling strategies", Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR04), Barcelona, 2004.

[4] Sheh, A. and Ellis, D. "Chord Segmentation and Recognition using EM-Trained Hidden Markov Models", Proceedings of the 4th International Symposium on Music Information Retrieval (ISMIR03), Baltimore, USA, 2003.

[5] Yoshioka, T., Kitahara, T., Komatani, K., Ogata, T. and Okuno, H. "Automatic chord transcription with concurrent recognition of chord symbols and boundaries", Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR04), Barcelona, 2004.

[6] Fujishima, T. "Real-time chord recognition of musical sound: a system using Common Lisp Music", Proceedings of International Computer Music Conference (ICMC99), Beijing, 1999.

[7] Bartsch, M. A. and Wakefield, G. H. "To catch a chorus: Using chromabased representation for audio thumbnailing", Proceedings of International.

Workshop on Applications of Signal Processing to Audio and Acoustics, Mohonk, USA, 2001.

[8] Pardo, B., Birmingham, W. P. “The Chordal Analysis of Tonal Music”, The University of Michigan, Department of Electrical Engineering and Computer Science Technical Report CSE-TR-439-01, 2001.

[9] Mitchell, T. “Machine Learning”, The McGraw-Hill Companies, Inc. 1997.

[10] Cabral, G., Zanforlin, I., Santana, H., Lima, R., & Ramalho, G. “D'accord Guitar: An Innovative Guitar Performance System”, in Proceedings of Journées d'Informatique Musicale (JIM01), Bourges, 2001.

[11] Koza, J. R. "Genetic Programming: on the programming of computers by means of natural selection", Cambridge, USA, The MIT Press.

[12] Gómez, E. Herrera, P. “Automatic Extraction of Tonal Metadata from Polyphonic Audio Recordings”, Proceedings of 25th International AES Conference, London, 2004.

[13] Website: http:// timidity.sourceforge.net/