Some Case Studies in Automatic Descriptor Extraction

Giordano Cabral[1], Jean-Pierre Briot[1], Sergio Krakowski[2], Luiz Velho [2],
François Pachet[3], Pierre Roy[3]

[1]Laboratoire d'Informatique de Paris 6 (LIP6) – Université Pierre et Marie Curie
8, Rue du Capitaine Scott, 75015 Paris, France

[2]Instituto Nacional de Matemática Pura e Aplicada (IMPA)
Estrada Dona Castorina, 110, 22460-320 Rio de Janeiro, RJ, Brazil

[3] SONY Computer Science Lab
6 Rue Amyot, 75005 Paris, France

{Giordano.CABRAL,Jean-Pierre.BRIOT}@lip6.fr,
{skrako,lvelho}@visgraf.impa.br, {pachet,roy}@csl.sony.fr

Abstract. *This work aims to evaluate the effectiveness of EDS as a tool to automatically extract descriptors for real-world problems, such as melody extraction, chord recognition, and sound classification, comparing its performance and development time to traditional approaches. Each of these problems constitutes a case study, and along with the comparative results we present some remarks about the descriptor extraction procedure.*

1. Introduction

The last few years have witnessed an effort in rendering the descriptor extraction process automatic, in order to 1) improve (current) features efficiency; 2) create new features in an easier and faster way; and 3) allow non-signal processing experts to create sound descriptors. In this scenario, the Extractor Discovery System (EDS) [Pachet and Zils 2004] a very promising option, addressing the automation of the whole process. It started to be developed in 2003, and has been continuously improved ever since.

In a previous work [Cabral et al. 2005] we explored the power of EDS to find harmonic descriptors (chord recognizers) in a completely automated way. That work tried to simulate the use of the system by a non-expert user. The current work extends the previous one, by evaluating EDS capacity to find good descriptors for some well-known and usual problems, namely the f0 estimation, chord recognition, and percussive sound classification whereas EDS is operated by an expert.

The experiments presented in this paper intend to provide some examples of employing EDS in real-world situations. These examples should be indicative of the possibility of automatically extracting features, revealing its strengths and weaknesses. In a simple way, we are trying to answer the following question "supposing someone wants to develop an application such as a chord recognizer, a melody extractor, an accompaniment system, or whatever other tool that needs to classify sound fragments, would it be interesting to use an automatic descriptor extraction tool like EDS instead of traditional techniques"?

We try to answer this question by presenting three case studies, which actually make part of broader systems being developed by Sergio Krakowski at the VISGRAF/IMPA group in Rio de Janeiro, Brazil, and Giordano Cabral at LIP6 and Sony CSL in Paris, both advised by Francois Pachet. More particularly, we are interested in transcribing melody, harmony, and rhythm of songs, with the purpose of building interactive systems.

The transcription of musical information usually relies on a general approach: to perform a short-term analysis on a sliding window, and track the result over time using some kind of dynamic modeling such as HMM's or GMM's [Pachet and Briot 2004] (Figure 1). This analysis typically means the computation of a feature which is pertinent to the problem (e.g. the autocorrelation of the signal, or the pitch class profile).

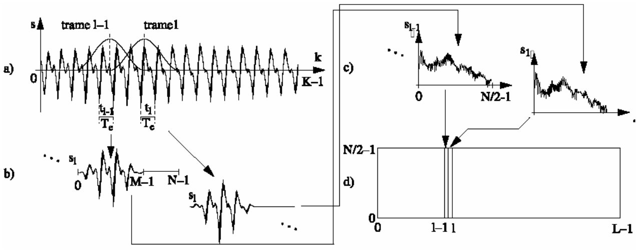


Figure 1 – short-term analysis of a signal.

Nevertheless, those features could hypothetically be automatically discovered by a descriptor extractor, such as EDS. That's precisely the goal of the present work, which compares the performance and development time of EDS to those of traditional approaches.

The next 3 sections respectively describe the 3 problems we are investigating, and the main features used for their analysis. Section 5 gives further detail on how EDS works. Sections 6, 7, and 8 relate the 3 case studies: f0 estimation, chord recognition, and percussive sound classification, presenting the methodology adopted in both approaches as well as the results that were found. Finally, section 9 makes some overall comments and remarks, and section 10 presents the conclusion and future work.

2. F0 Estimation

The problem of estimating the fundamental frequency of a sound is well known by the scientific community. Among the applications of this problem we can cite automatic melody transcription and real time accompaniment. There are many f0 estimation techniques (and even different taxonomies) available in the literature. However, one general classification is clearly observed: the distinction between approaches dealing with the information on the time domain from others based on the frequency domain.

Time-based techniques rely on the high correlation of a signal with this same signal shifted by particular values. These values are usually called *lags*, each one referring to a respective frequency. The correlation is computed for a set of candidate *lags*, and the highest correlation is taken as 1/f0 [Klapuri 2004]. One variation called AMDF [Pachet and Briot 2004] is claimed to be more robust once it redefines the correlation as the difference between the original signal and the shifted one, instead of their multiplication. Alternatively, the *cepstrum* can be used as a replacement for the correlation function [Klapuri 2004].

Frequency-based techniques analyses the signal transformed into the frequency-domain. Whenever a quasi-periodic signal have period T, it can be considered periodic with period nT. This behavior can be seen in the frequency domain by a regularity of the peaks found in the DFT. Thus, a probability function of the fundamental candidates is made by convolving the resultant DFT with comb functions aligned to the multiples of the candidate frequency. An interesting variation of this technique was suggested by [Kunieda 1996]. It is considered as of spectral-interval type and seeks for periods in the frequency domain by performing a sort of autocorrelation of the spectrum. Other variations exist, notably the ones based on the human auditory model [Cheveigné and Kawahara 1999].

In general lines, the frequency-based approach is usually more appropriate to higher register signals, since the DFT has better resolution around higher frequencies, while time-based estimators are more appropriate to lower ones, since more distant periods provide greater precision in the correlation [Klapuri 2004]. Ideally, different methods could be merged in order to achieve robustness, maintaining a good performance independently on the register.

3. 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 by its turn can also be subdivided, for example in a root note and a type. Most part of the works involving harmonic content (chord recognition, chord segmentation, tonality estimation) [Sheh and Ellis 2003][Yoshioka 2004] use the same core technique (even though slight variations may appear in the implementation): to compute an harmonic feature, such as the *Pitch Class Profile* (PCP) [Fujishima 1999], or the *chromagram* [Bartsch and Wakefield 2001], and a subsequent machine learning algorithm to find patterns for each chord class.

PCPs are vectors of low-level instantaneous features, representing the intensity of each pitch of the tonal scale mapped to a single octave. These vectors are calculated as follows: 1) a music recording is converted to a Fourier Transform representation (Figure 2a to Figure 2b); 2) the intensity of a pitch is calculated (Figure 2b to Figure 2d) by the magnitude of the spectral peaks, or by summing the magnitudes of all frequency bins that are located within the respective frequency band (Figure 2c); 3) The equivalent pitches from different octaves are summed, producing a vector of 12 values (eventually 24 to deal with differences in tuning and/or to gain in performance), consequentially unifying various dispositions of a single chord class (Figure 2e and Figure 2f). For example, one can expect that the intensities of the frequencies corresponding to the notes C, E and G in the spectrum of a Cmaj would be greater than the others, independently on the particular voicing of the chord. The *chromagrams* follow a different method to be computed, but are conceptually the same.

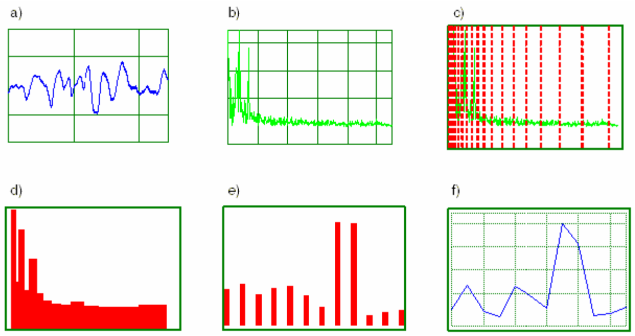


Figure 2 - Steps to compute a PCP. The signal (a) is converted to Fast-Fourier representation (b); the FFT is divided into regions (c); the energy of each region is computed (d); these energies are folded into a 12-note vector (e); the final vector is normalized (f).

The idea behind the use of *PCPs* for chord recognition is that the PCPs of a chord follow a pattern, and patterns can be learned from examples. Thus, machine learning techniques [Mitchell 1997] can be used to generalize a classification model from a given database of labeled examples, in order to automatically classify new ones. So, for the *PCP* of a chord, the system will respond the most probable (or closest) chord class, given the previously learned examples. The original *PCP* implementation from Fujishima used a KNN learner [Fujishima 1999], and more recent works [Gomez and Herrera 2004] successfully used other machine learning algorithms.

4. Percussive Sound Classification

Percussive sound classification is a vast field in which many techniques have been suggested but until now no one can be considered standard nor universal. The general problem is to automatically classify percussive sounds in order to subsequently retrieve the rhythmic structure or transcribe the rhythmic score. In that way, it can be seen as analogous to the melody extraction problem, but instead of notes, we are interested in sounds with different timbres. The set of possible timbres can be incredibly high due to the diversity of instruments, and then solutions vary according to particular drum/percussion sets and the information one wishes to extract.

Thus, a number of different strategies have been applied to specific problems. In [Gouyon et al. 2000] the authors addressed the bass/snare drum discrimination, and eventually considered the automatic extraction of rhythmic structures. For that, they observed the zero-crossing rate estimation on the decay part of the sound. Another approach is the source separation as found in [FitzGerald 2004]. The idea is to consider each sound as made by one or more sources and, by subspace analysis, find the best sources that model the sounds. Finally we can find an extensive work done by Herrera, comparing many feature selection methods and classification techniques applied first to drum kit [Herrera et al. 2002] and afterwards to percussion instruments [Herrera et al. 2003].

5. EDS

The Extractor Discovery System, developed at Sony CSL, is a heuristic-based generic approach for automatically extracting high-level music descriptors from acoustic signals. EDS is based on Genetic Programming [Koza 1992], used to build extraction functions as compositions of basic mathematical and signal processing operators, such as Log, Variance, FFT, HanningWindow, etc. A specific composition of such operators is called feature (e.g. *Log (Variance (Min (FFT (Hanning (Signal))))))*, and a combination of features forms a descriptor.

Given a database of audio signals with their associated perceptive values, EDS is capable of generalizing a descriptor. Such descriptor is built by running a genetic search to find relevant signal processing features matching the description problem, and then machine learning algorithms to combine those features into a general descriptor model.

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 cross-overs), respecting a pattern chosen by the user. The default pattern is *!_x(Signal)*, which means a function presenting any number of operations but a single value as result (for more information about EDS syntax, look at [Zils and Pachet 2004]). The populations of functions keep reproducing until no improvement is achieved, or until the user intervenes. At this point, the best functions are available to be combined. A selection can be made both manually or automatically. The final step is to choose and compute a model (linear regression, model trees, knn, locally weighted regression, neural networks, etc.) that combines all features. As an output, EDS creates an executable file, which classifies an audio file passed as argument.

In short, the user needs to 1) create the database, in which each recording is labeled with the correspondent class; 2) write one or more general patterns for the features; 3) launch the genetic search; 4) select the appropriate features; 5) choose a model to combine the features. Some of the choices taken in these steps are crucial to the process. They delimit how the user can interfere in the search for features, as explained next.

5.1. Pattern Choice

The pattern encapsulates the architecture of the feature. They are represented by the output of the chain of functions to be found. As single values, this output can be a frequency (*f*), amplitude (*a*), time (*t*), or any of them (*x*). Additionally, the output can be a relation, such as time and amplitude (*t:a*), or frequency and amplitude (*f:a*). Moreover, a recent improvement of the system allowed outputs to be a vector of any of the previous types. The patterns may include intermediate outputs, which will be inputs of the outer function. Along with these symbols, a *_ or a !_ is placed to express if the user wants to search a single operator or a sequence of them. At last, the patterns may include specific operators. For example, *!_f(f:a(Signal))* means that the signal is initially converted into the frequency domain (*f:a*), then some operation is applied to get a frequency as a result (*!_f*).

5.2. Genetic Search

Given a set of patterns, a genetic search is launched. It means that a population of features is created, and the capacity of each one to separate the examples in the database is evaluated. The best features are then selected as seeds to a new population. This process evolves the features until no improvement is found.

Although the genetic search can be performed fully automatically, the user can supervise and interfere in the search. This intervention is even desired, since the space of possibilities is enormous, and heuristics are hard to express in most cases. Therefore, the user can lead the system through some specific paths by 1) stopping and restarting the search if it is following a bad path; 2) selecting specific features for future populations; 3) removing ineffective features from the search. Additionally, the stop condition itself is an important factor frequently left to the user.

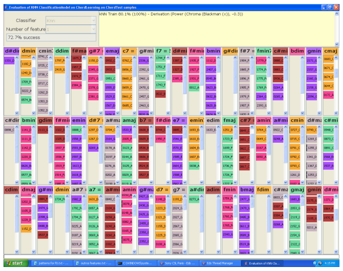
The choice of the population size may also influence the search, since larger populations may hold a bigger variety of features (which will converge slower), whereas smaller populations will perform a more in depth (faster) search, (which will be most likely to terminate at local maxima). At last, the user can optimize features, finding the values for their arguments which maximize the class separation. For example, the split function (which divides a signal in sub-signals) has the size of the sub-signals as a parameter. Depending on the case, a tiny value can be notably better than large values, for example.

5.3. Feature Selection

After many features were found, possibly in different genetic searches, they can be combined to create the final descriptor (eventually with a single feature). The selection of which features to combine is left to the user, even if one useful tool is available: the *expert selection* picks up the features that are better than a customizable threshold and less correlated than another customizable threshold. In fact, as [Herrera et al. 2002] shows, choosing features in a list is as complicated as finding the features themselves, so that is maybe the point at which the quality of the result is more dependent on the user.

5.4. Descriptor Creation and Evaluation

Finally, in order to create the descriptor, the learning method that will combine the features must be chosen (normally KNN or GMM). The resultant descriptor is then evaluated on a test database. The results are presented class by class, along with the precision rates.



Evaluation of KNN ClassificationModel on ChordLearning on ChordTest samples

Classifier: Knn

Number of feature: 1

72.7% success

kNN Train 80.1% (100%) - Derivation (Power (Chroma (Blackman (x)), -0.3))

d#dir dmin cmin7 ddim f#ma g#7 = emaj c7 = g#mi f7 = d#mi f#mi bmin g#dir f#7 = fmin7 c#mi bdim gmin cmaj
c#dir bmin7 gdim f#mi emin d#7 = a#ma amaj b7 = f#dir e7 = emin7 edim fmaj c#7 = amin7 a#mi cmin d#ma c#mi
cdim dmaj g#mi dmin7 a#7 = a7 = a#mi amin g#ma d7 = g7 = a#dir adim fmin bmaj fdim c#ma gmaj gmin7 d#mi
start
patterns for f0.txt - ...
outros features.txt - ...
C:\WINDOWS\syste...
Sony CSL Paris - Eds ...
Eds Thread Manager
Evaluation of KNN Cla...
4:15 PM

Figure 3 – Snapshot of EDS screen givin g the results of a chord recognition descriptor classified with KNN, and evaluated on the test database.

6. Case Study One: F0 Estimation

Our first experiment compares the results of EDS with those of two of the most widely used techniques for the f0 estimation, one in the time-domain, other in the frequency-domain. In order to evaluate the algorithms, we created a database of 1570 wave files, each one containing the sound of a note, ranging from A0 to C9. The wave files were rendered from midi files using SoundFonts [Timidity 2006], where each one encapsulated one note played by one specific melodic instrument.

The two techniques examined were the autocorrelation via *AMDF* and the filtering of the spectrum at specific frequencies, both explained in section 2. The autocorrelation feature was implemented as in [Pachet and Briot 2004], with 100 candidate frequencies starting from 27.5. The correlation function was defined as the difference between samples, as shown in the formula below.

$$FP_{AMDF,l}(\tau) = \frac{1}{Norme} \sum_{k=0}^{M-1} |s_l(k) - s_l(k+\tau)|$$

Figure 4 – formula of the AMDF autocorrelation function

The second one is called here *FPl*, and uses a comb filter on the spectrum of the DFT of the sound (see formula in Figure 5), this filter using the same candidate frequencies as in the previous example.

$$FP_l(f_0) = \int_f ⧢_{f_0}(f) |\widetilde{s_l}(f)|\, df = \sum_i |\widetilde{s_l}(i f_0)|$$

Figure 5 – formula of frequency-based FPl function

The result of these functions can be interpreted as probabilities of each candidate to be the fundamental. As a further step, machine learning algorithms can be used to map the patterns of these density curves to specific pitch classes. We have implemented these 4 possibilities, called here *Pure AMDF*, *Pure FPl*, *AMDF+Knn*, and *FPl+Knn*. Machine learning algorithms could be used as well to combine more than one of these features into a single solution. We have not implemented this combination, but still calculated its upper limit, defined as follows: if one of the algorithms gives the good solution, the combined algorithm will also do it. The results are presented in the Table below:

Table 1. Results of Traditional Techniques for F0 Estimation

Method	Precision
Pure AMDF	45.10%
Pure FPl	43.64%
AMDF+Knn	71.51%
FPl+Knn	44.18%
Upper Limit	77.90%

Noticeably the results reflect the deficiency of the techniques in specific ranges. While the correlation-based works better for low-frequencies, the frequency-based works better for high-frequencies. The learning phase corrected some misclassifications but still did not work for the frequency-based solution.

One must notice that, including the research and reading of specialized papers and books, we took 2 weeks to implement the first solution (Pure AMDF). 1 more day was necessary for the second solution, and 2 more days for the Knn versions. Altogether, the experiment took 2 weeks and 4 days (14 working days). Even if this information may not be rigorously scientific, given that other people with different background, programming skills, and dedication might perform differently, we find it useful to give an idea of the order of the time which is needed for its implementation. One must not forget that this is a part of major systems, and not a problem by itself. The developers are experienced programmers, with medium level sound processing skills, that devoted 6 hours a day to this particular problem.

The goal of the experiment is to verify if the automatic extraction can pass the 71.51% rate of correctly classified instances, hopefully approximate the 77.90%, and ideally pass the 77.90%, as well as to monitor how much time it takes.

All images and illustrations should be in black-and-white, or gray tones. The image resolution on paper should be about 600 dpi for black-and-white images, and 150-200 dpi for grayscale images. Do not include images with excessive resolution, as they may take hours to print, without any visible difference in the result.

6.1. Automatic Extraction of F0

The system was taken from scratch, and we performed some searches using the general patterns: "*_a(x)", "*_f(x)", "*_Va(x)", and "*_Va(*_Va(t:a(x)))", the more specific patterns: "PitchBands(*_t:a(x), 120.0)", "*_Va (PitchBands(*_t:a(x), 120.0)", "*_a(Autocorrelation (*_t:a(x)))", "*_f(Autocorrelation(*_t:a(x)))", "*_Va (Autocorrelation(*_t:a(x)))", "BarkBands(x, 25.0)", "Chroma(x)", "*_Va(Chroma(*_t:a(x)))", "*_Va (SplitOverlap(Autocorrelation(x), 441, 0))", and the very specific (optimization) patterns "PitchBands(x, 120.0)" and "Correlation(x, t:a(x))".

After 3 days of 12 genetic searches, some of them long and intensive, EDS found over 20 features superior to 70%, even though most of them were extremely correlated. On the other hand, they were found in different paths (from different patterns), needing more or less time to be found. We must mention that the correlation-based features scored very badly (19% at best, against 75% from the frequency-based), indicating some malfunctioning in the system. Until the present moment, we are not aware if this is a problem with the genetic search or an error in the operator itself.

We pre-selected 11 features for further evaluation. Many permutations of these features were tried. We hoped that some features would be complementary to the best one(s), improving the overall result. Strangely, the more features in the descriptor the worst was its performance. In fact, weaker features can bring down the quality of the descriptor depending on the method chosen to combine them, and the best descriptor in fact used only 1 feature.

```
1.  Derivation (PitchBands (x, 120.0))
2.  BarkBands (x, 120.0)
3.  Chroma (Derivation (BpFilter (x, 488.0, 26.0)))
4.  Integration (Integration (Hamming (Zcr (Split (Autocorrelation (x),
    882.0)))))
5.  PitchBands (x, 120.0)
6.  Derivation (BarkBands (x, 100.0))
7.  BarkBands (x, 150.0)
8.  Chroma (Hann (BpFilter (x, 488.0, 26.0)))
9.  Triangle (SpectralRolloff (Split (Autocorrelation (x), 3307.0)))
10. BarkBands (Abs (Autocorrelation (Autocorrelation (x))), 5.0)
11. SpectralCentroid (Autocorrelation (x))
```

The Figure 6 shows the final results, comparing them to those from the traditional algorithms. It is interesting to notice that the expert selection facility actually selected just the best feature, based on a minimum quality and a maximum correlation among all the features found. The best descriptor scored 75.08%, using a frequency-based approach. The experiment lasted 4 ½ days.

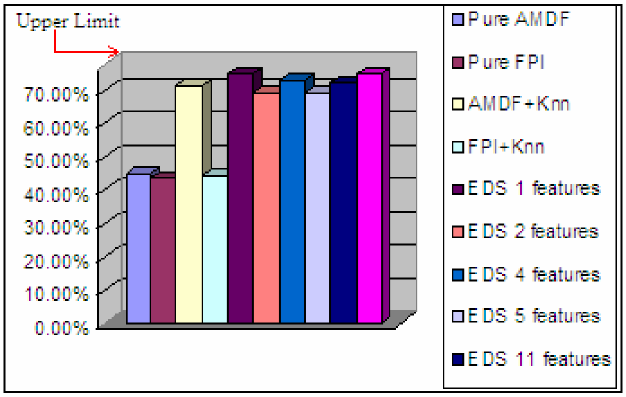


Figure 6 – Results of the F0 Estimators.

7. Case Study Two: Chord Recognition

The final goal of our chord recognizer is to create a guitar accompanier in Brazilian “bossa nova” style. Consequently, our database has examples of chords played with nylon guitar. The data was taken from D’accord Guitar Chord Database [Cabral et al. 2001], a guitar midi based chord dataset. The purpose of using it was the richness of its symbolic information (chord root, type, set of notes, position, fingers, etc.), which was very useful for labeling the data and validating the results. Each midi chord was rendered into a wav file using SoundFonts [Timidity 2006] and a free nylon guitar patch. The EDS database was created according to the information found in D’accord Guitar database. The database divided the chords into 60 classes, 5 types per root note: major, minor, seventh, minor seventh and diminished. From 1885 samples, 80% was settled on as the training dataset and 20% as the testing dataset. We implemented the traditional KNN over Pitch Class Profile algorithm in order to make a comparison. More details about it can be found at [Cabral et al. 2005]. Considering research and implementation, we took almost 4 weeks to implement it.

7.1. Automatic Chord Recognizer

The same databases were loaded in EDS. In our work from 2005, cited above, we found some middling features after having run the system in a fully automated way. Our strategy here is to merge new specialized features with the previous ones. In order to find new and better features, we used specific patterns, appropriate to the problem, mainly: “*_*Va(*_t:a(x))*”, “*chroma(x)*”, “*_Va(PitchBands (*_*t:a(x)*, 120)”, “*_*Va(BarkBands(*_t:a(x), 120))*” and insistently the pattern: “*_*Va(chroma (*_t:a(x)))*”. Grosso modo, we intended to find features which firstly transformed the signal into meaningful information, like the *chroma* and the *pitchbands* do. The *chroma* EDS operator must not be confounded with the *chromagram* or the *PCP* features. These are ready-to-use features, more or less complex, comprehending many processing tasks, including pre and post-processing, while the *chroma* simply folds each bin from the DFT into a single octave, storing the average value for each note.

The search was launched over 40 times, but as well as for the F0 Estimation case, the majority of searches converged to similar results. Notably, the chroma-based features surpassed their concurrent in part due to their adequacy to the problem, in part due to the fact that it does not have extra (internal) parameters, such as the PitchBands or the

BarkBands does[1]. In fact, an internal variable can lead the feature to poor results, even if they are potentially good, renouncing their persistent evolution.

Finally, we selected 14 features (11 from the previous work plus 3 just discovered). The 3 new features are:

```
1. Derivation (Power (Chroma (Blackman (x)), -0.3))
2. Log10 (Chroma (Hamming (x)))
3. Hann (PitchBands (x, 120.0))
```

As in the previous experiment, combining many features did not work better than using a single one. However, the single best one worked significatively better than the traditional solution (72.68% against 63.93%), as illustrated in Figure 7. The time needed to finish the experiment was 11 days.

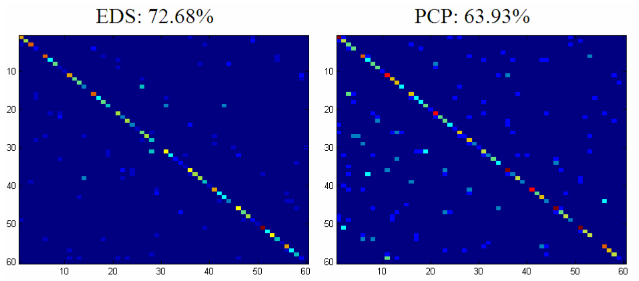


Figure 7 – confusion matrix and precision rate of EDS (in the left) and the traditional PCP method (in the right).

8. Case Study Three: Percussive Sound Classification

The *pandeiro*, a Brazilian variant of the tambourine, is a very important instrument in the musical tradition of the country. Our aim is to automatically classify the different types of strokes played on the *pandeiro* in order to build a reactive system. It is possible to distinguish among three main categories of sounds: low strokes, slap strokes and jingles strokes, although they are not completely mutually exclusive (the jingles are usually played along with the other ones). We have not found any specific work about this instrument in the scientific literature. However, the strong parallel between this instrument and a drum kit conducts us to adapt algorithms initially conceived for the last one. The low strokes can be related to the kick drum due to its low frequency content, around 100 to 250Hz, whereas the slap strokes have the same strong loudness attack and fast decay characteristics of the snare drum. Also, the jingles strokes have its spectral information located at higher frequencies (around 10 to 15 KHz) similar to the hi-hat.

The examples in our database are divided in six classes: two types of low sounds, named '*tung'* and *'ting'*; two types of slap sounds, named *'pa'* and *'grand pa'*; and two types of jingle sounds, named *'tchi'* and the *'tr'*. Noticeably, it is much harder to distinguish between classes in each pair than among the 3 pairs. We recorded several minutes of *pandeiro* solo containing all the six types of sounds, using different microphones and locations in order to preserve some inherent analysis difficulties, such as variations on the room reverberation and different frequency responses of the equipment. We built our database by automatically segmenting these recordings via a peak detector through the derivative of the convolved loudness curve of the signal, as

[1] Both the pitchbands and the barkbands have the number of bands as parameter.

described in [Pachet and Briot 2004]. The database was split into a training part, with 155 sound samples, and a testing one, with 288.

In this case study, we used EDS not only to automatically find a descriptor, but to assess the traditional solutions as well. In fact, a big part of the traditional features are equivalent to EDS built-in operators, such as the zero-crossing rate (ZCR) proposed by Gouyon [2000]. Thus, EDS showed to be useful as a try-and-test tool, allowing the user to instantaneously evaluate these features. Our first experiment was to evaluate if the ZCR, which Gouyon demonstrated to well discriminate between the kick and the snare drum, is suitable for the pandeiro. The unconvincing result of 47.0% is shown in detail in Table 2.

Our second experiment used the fact that each class has most of its spectral information located in a different region. The sum of band-pass filters, the spectral centroid, or any operator that divides the spectrum in sub-bands seem to be appropriate to capture this aspect. The result of dividing the spectrum in 20 Bark bands was considerably better (90.9%), and is also presented in Table 2.

These experiments spent just a couple of minutes to be done. After some manually created features, we did a first attempt to automatically generate features, by launching the genetic search with the following patterns: "!_a (x)", "!_Va (x)", "!_Va (SplitOverlap (x, 220.0, 0.1))", "!_Va (BarkBands (x, 20.0))", and "!_a (BarkBands (x, 20.0))". This search was stopped after 3 populations of 50 features, which took about 30 minutes to be calculated. We selected the 11 features listed in the next page, resulting in 87.2% of correctly classified instances, as shown in Table 2 under the designation EDS1.

```
SpectralDecrease(SplitOverlap(x,220.0, 0.1))
Rms (SplitOverlap (x,220.0,0.1))
RHF (SplitOverlap (x,220.0,0.1))
Mean (SplitOverlap (x,220.0,0.1))
SpectralSpread(SplitOverlap (x,220.0,0.1))
SpectralDecrease(SplitOverlap(x,220.0,0.3))
Rms (SplitOverlap(x,220.0,0.3))
Chroma (x)
SpectralDecrease (x)
HFC(Mean(SplitOverlap(x,220.0,0.1)))
Range(SplitOverlap(x,220.0,0.1))
```

Finally, we left the genetic search run for twelve hours, reaching 618 populations. Among the great number of features found, we selected the features scoring more than 80% and less correlated than 50%, as listed below. The final rate was 89.2%, showing that the system can converge very quickly, but may take a long time to make slight improvements. The detailed results are also presented in Table 2, under the label EDS2.

```
Rms(SplitOverlap(x,220.0,0.1))
SpectralSpread(SplitOverlap(x,220.0,0.1))
SpectralDecrease(SplitOverlap(x,220.0,0.1))
SpectralKurtosis(SplitOverlap(x,220.0,0.1))
Variance(SplitOverlap(x,220.0,0.1))
Mean(SplitOverlap(x,220.0,0.1))
Sqrt(Sqrt(PitchBands(x,5.0)))
Log10 (PitchBands(x,5.0))
Square (Mfcc0(x,10.0))
Square (Chroma (x))
Power(Hanning (BarkBands (x, 20.0)), -0.5)
```

Table 2. Results for the Percussive Sound Classification

Method	Low		Slap		Jingle		Overall
	Tung	Ting	Pa	Gr Pa	Tchi	Tr	
ZCR	68.2%	38.1%	10.9%	22.5%	72.0%	18.8%	47.0%
SubBands	92.2%	42.9%	93.5%	72.5%	89.2%	81.2%	80.9%
EDS1	84.3%	92.9%	84.3%	80.0%	91.4%	81.2%	87.2%
EDS2	90.2%	61.9%	91.3%	95.0%	97.8%	87.5%	89.2%

Besides the bad cost-benefit relationship between performance and computational time, the data shows that the mixture of features enhanced the robustness of the solution (visibly the '*ting'* class, a kind of low stroke frequently misclassified by the traditional solutions). Additionally, it became clear that a split overlap analysis followed by a post-processing technique such as Rms or SpectralSpread overwhelmed other features, becoming predominant inside the populations. Finally, the addition of other kinds of features such as Bark bands, Chroma, Mfcc and Pitchbands apparently increased the quality of the descriptor.

9. Discussion

EDS revealed itself as a good feature exploration mechanism, and as so it seems especially appropriate to new or scarcely explored problems. The case study three illustrated this exploratory characteristic of EDS, as the user was able to try-and-test many possibilities, some of them fairly complicated, serving to validate ideas and/or build prototypes. The results in sections 6 and 7 pointed out that EDS can achieve satisfactory results for real world well-known problems. It exceeded or improved traditional techniques, either by optimizing existing ones or launching searches from scratch. On the other hand, the system did not find any especially innovative feature. Even when started from scratch, EDS-created features corroborated those from specialists.

For these specialists, EDS appeal relies on being: 1) an automatic tool to investigate interesting possibilities of work, which can afterward be refined by the specialist; as well as 2) a tool to optimize existing features.

We consider the feature selection methods could be improved, for example by using the procedure described in [Herrera et al. 2002], since it lacks precise control of the features and operators (ultimately the impossibility to build new operators).

At last, the quality of the final solution depends highly on the user. For instance, our previous work explored the same chord recognition database, while EDS was operated by a naïve user. The final descriptor scored 40.31%, in contrast to the 72.7% achieved here.

Even with these disadvantages, we were quite satisfied with the performance of the system, especially with its capacity to join the entire feature creation procedure into a single piece of software, including the database creation, the search and selection of good features, and the machine learning algorithms to combine them. A whole set of programs which would be very expensive to implement. For this work, the results of traditional approaches were obtained by re-implementing the algorithms described in the literature, hence the performance could obviously be improved by cutting-edge technicians. However, to remember the initial question, we consider these case studies realistic to mimic someone who would start to develop some of these systems.

One extra asset of EDS is the visualization of the result. The rather simple interface, showing the sound samples colored by their class, easily shows the correctly classified instances, making the evaluation/analysis of the result more intuitive and direct. For example, in the chord recognition problem we noticed that 18.94% of the errors referred to equivalent chords, like C° and A°, F#m7/C# and A6/C#, or C/G and Am7/G. Moreover, the system grouped together some chords (11.57% of the errors) with the same function, such as F7 and B7(#11), or Em7 and A4/7. The system even corrected some faults in the database, placing chords like C/Bb in the right class (C7 instead of Cmaj). This situation represented 6.31% of the errors. In the end, the actual error rate would be of 12.47% (instead of the 27.32% previously mentioned), from which only 0.79% refers to flagrant errors.

10. Conclusion and Future Work

This paper presented 3 case studies illustrating the use of an automatic extraction tool called EDS. The work intended to validate the effectiveness of the tool in creating descriptors sufficiently good to be used in real-world applications. The case studies referred to the f0 estimation, the chord recognition, and the percussive sound classification problems. The results showed that the system was able to achieve and even surpass traditional descriptors.

These descriptors were needed for two systems currently being developed (an accompaniment system and an interactive rhythmic tool), and we found it very useful to share our experience with the community. We commented the usage of the system, and suggested some desirable improvements, which can be envisaged as future work.

11. Acknowledgments

We would like to thank the whole music team at Sony CSL for the support and encouragement.

References

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.

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

Cabral, G., Pachet, F., and Briot, J.-P. “Automatic x traditional descriptor extraction: The case of chord recognition”, Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR'2005), London, U.K., September 2005.

Cheveigné, A. de, and Kawahara, H. “Multiple period estimation and pitch perception model,” Speech Communication 27, 175–185, 1999.

Fitzgerald, D. "Automatic Drum Transcription and Source Separation" PhD Thesis, Conservatory of Music and Dram, Dublin Institute of Technology, 2004.

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

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

Gouyon, F., Pachet, F., and Delerue, O., "On the use of zero-crossing rate for an application of classification of percussive sounds", DAF-00, Verone, Italy, 2000.

Herrera, P., Yeterian, A., and Gouyon, F. "Automatic classification of drum sounds: a comparison of feature selection methods and classification techniques", 2002.

Herrera, P., Dahamel, A., and Gouyon, F., "Automatic labeling of unpitched percussive sounds", AES 2003.

Klapuri, A. “Signal Processing Methods for the Automatic Transcription of Music” Doctoral Dissertation. Tampere University of Technology, Tampere, Finland, 2004.

Koza, J. R. "Genetic Programming: on the programming of computers by means of natural selection", Cambridge, USA, The MIT Press, 1992.

Kunieda, N., Shimamura, T., and Suzuki, J. “Robust method of measurement of fundamental frequency by ACLOS — autocorrelation of log spectrum,” Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, 1996.

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

Pachet, F and Briot, J.-P. Du signal au signe musical. Hermes, 2004.

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.

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.

Timidity. Website: http:// timidity.sourceforge.net/, 2006.

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.

Zils, A. and Pachet, F. Automatic Extraction of Music Descriptors from Acoustic Signals using EDS. Proceedings of the 116th AES Convention, May 2004.