AUTHOR GUIDELINES FOR ICASSP 2018 PROCEEDINGS MANUSCRIPTS

Mattia Lecci, Federico Mason, Victor Cercos Llombart

UPC ETSETB - MET

ABSTRACT

***Index Terms*—** One, two, three, four, five

1. INTRODUCTION

Citation test: [**?**], [**?**], [**?**]

2. TECHNIQUES USED

2.1. Features Extraction

Feature extraction is performed using *Chroma Toolbox* [**?**], a MATLAB implementation of many different features under GPL license.
At first, the audio signal is decomposed into 88 frequency bands centered in the frequencies corresponding to the pitches of A0 up to C8. A multirate filter bank using elliptic filters is used and finally *Short-Time Mean-Square Power* (STMSP) is used to extract the useful information. In order to ignore the the different harmonicities given by different instruments, the energies referred to the same notes are pooled together resulting in 12-dimensional vectors.
The different features are then obtained operating in different ways on the extracted pitches. In this project we used three of them:

- CLP (*Chroma Features with Logarithmic Compression*): bands from different octaves are summed together, a logarithmic function is applied (to emulate our logarithmic perception of sound intensity) and then the vector is normalized to L_2-norm equal to 1.
- CENS (*Chroma Energy Normalized Statistics*): in order to account for dynamics, timbre, articulation, execution of note groups, and temporal micro-deviations, smart logarithmic quantization, temporal smoothing, downsampling and normalizations are performed.
- CRP (*Chroma DCT-Reduced log Pitch*): they include logarithmic compression, smoothing and timbre invariance by using a DCT-based technique.

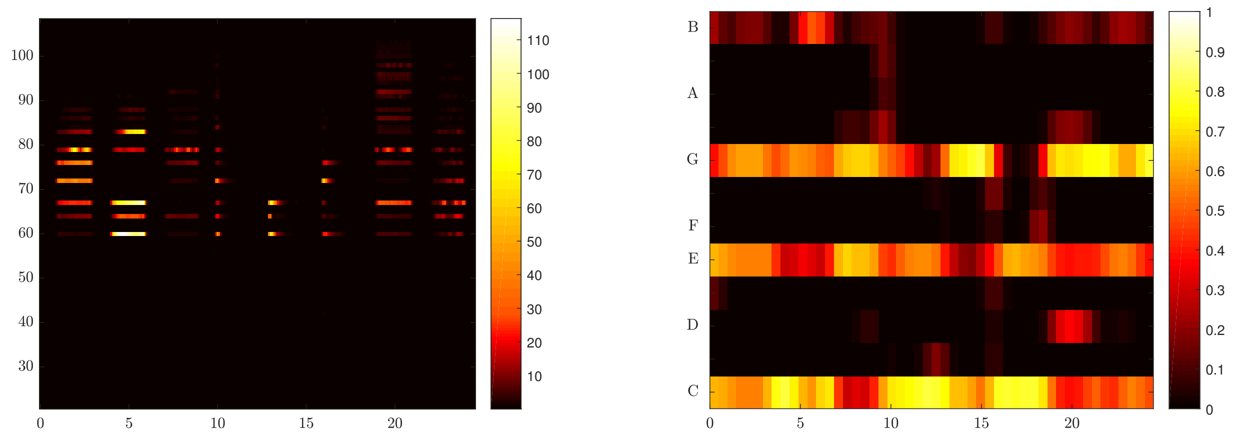
2.2. Template-based chord recognition

2.3. Gaussian Mixture Models (GMMs)

Gaussian Mixture Model represents a statistic technique used to individue subpopolations within an overall population. GMMs can be used in a supervised or not-supervised manner and in both the case they describe each subpopulation with a Multivariate Gaussian Distribution (MGD). MGD represents the best known and used form of multivariate distribution and is defined only by a mean vector and a covariance matrix. We notice that given a MGD it is very simple determine the probability of that an object is included by the population described by the distribution. This can be done using the so-called *Mahalanobis distance* which measures how many standard deviations the considered object is distant from the center of the distribution.
MATLAB defines a class called *gmdistribution* which allows to construct an object that contain one or more MGDs. This object can be built either by adding every distributions by ourself (supervised method) or fitting the entire population to a chosen number of distributions (not-supervised method). MATLAB offers also the method *mahal* to compute the Mahalanobis distance between an object X and a GMM.
GMMs represent a very simple and effective technique, which can be easily implemented with the pre-established MATLAB methods. In our work we focus on the supervised approach of GMM, creating the singles subpopulations one at time during the so-called *training phase*.

2.4. Multiclass Support Vector Machines (MC-SVM)

Support Vector Machines (SVM) are a well known and widely used *Machine Learning* algorithm for binary classification. Before Neural Networks became so popular, they generally outperformed most of the other standard classification (and also regression) algorithms. Some good advantages over them is the possibility of choosing non-linear kernels (the most famous ones are the *Gaussian/RBF* and the *Polynomial*). Also, they are far easier to program and optimize than Neural Networks (less hyperparameters) and they tend to overfit less.
Their main problem, though, is that they were conceived for binary classification. Their extension to multiclass classification is not trivial but there are some standards ways to do it. One of them is already implemented in Matlab and it's called



(a) Example of pitch extraction (b) Example of CENS features

Fig. 1: Example of feature extraction from a track containing the same chord played by 8 different instrument. You can see in (a) the different harmonic content while in (b) the instruments are basically indistinguishable

Error-Correcting Output Codes (ECOC). It works by creating multiple classifiers and assigning to each of them some classes to be considered positive and some other to be negative and by assigning weights to these binary classifiers, the final decision is taken.

For our tests we used the *One-vs-One* method, meaning that we create a classifier for every couple of classes ($O(K^2)$) assigning one to the positive and one to the negative, while ignoring everything else. This is usually a good tradeoff between complexity and precision, but other options could be explored in future works.

2.5. Hidden Markov Models (HMMs)

Hidden Markov Model represents an extremely powerful technique that has been widely used in speech recognition application for twenty years. A HMM consistis in an extension of the classical Markov Model scenario, in which observations are considered probabilistic functions of states. The final result is a pair of stochastic processes connected to each other; one process is directly observable, the other process is not and therefore is called "hidden". A more detailed theory about HMM working can be found in [**?**].

In the speech recognition field HMMs are used as statistical method in order to recover specific sequences of sound. The validity of HMMs in the the particular case of chord transcription has already been demostrated in numerous papers including [**?**] and [**?**]. Both these work utilize the HMM approach proposed in [**?**], training HMM with the *Expectation-Maximization Algorithm*. As this work is not limited to implementing HMM, we choose to take a different approach. In particular we combine HMM to the SVM technique previously presented.

As suggested in [**?**] the working of HMM can be supported by the *Viterbi Algorithm*. This algorithm aims to find the most likely sequence of states given the sequence of observables in a HMM scenario. It works in a recursive mode, first finding the probability of each HMM state for each observable, then identyfing the most probable states sequence. Given a total number of states equal to N and a sequence of observables T long, the Viterbi algorithm shows a complexity equal to $O(T \times N^2)$. The implementation of Vierbi algorithm represents the most difficult challenge in the HMM approach. The codes already existing present problems and there fore need to be carefully adapted to our scenario.

3. EXPERIMENT SETUP

Our experimentation was split in two parts: the first was about recognizing a single chord from a single chord track, the second one was about recognizing the chords of full songs.

For the first part, we used the dataset `jim2012Chords` [**?**] created for their paper [**?**]. It's composed of a total of over 2.000 recordings of 10 guitar chords (both major and minor triads). Four different Guitars are used, as well as Piano, Violin and Accordion. Some tracks were recorded in an anechoic chamber, some other in a noisy environment. For each one of the tracks we obtain a 12-dimensional feature vector using *Chroma Toolbox* and taking the maximum value for each note. Looking at Fig. 1(b), the idea is that we are interested in high values of specific groups of notes in order to guess the chord and the maximum value is a simple way of doing this. It's biggest disadvantage is that impulsive noisy environment can easily mask the useful information introducing high values in the wrong notes. Finally, we run the different methods on these obtained 12D vectors.

Fo the second part, we used *The Beatles*' discography, which has been professionally transcribed. . . .

4. RESULTS

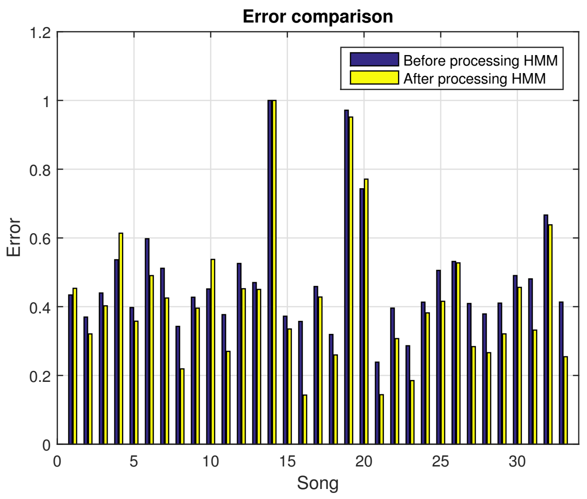


Fig. 2: Error comparison using CENS features (train = 0.8; test = 0.2)

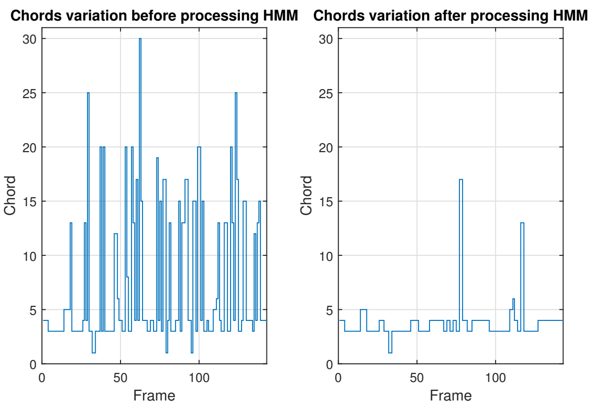


Fig. 4: Chord variation in a single song using CENS features (train = 0.9; test = 0.1)

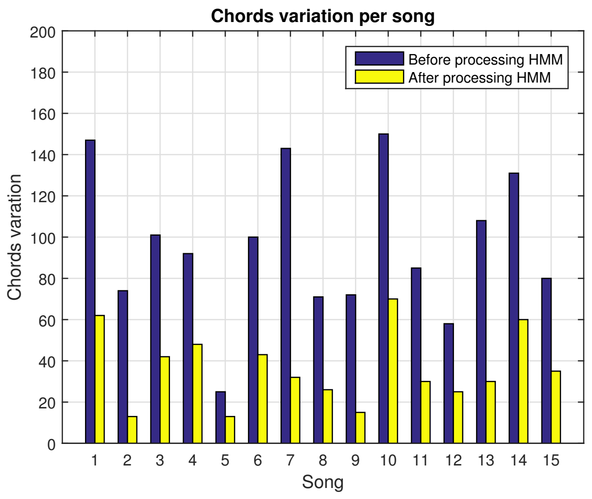


Fig. 3: Chords variation per song using CENS features (train = 0.9; test = 0.1)

5. CONCLUSIONS