Impact of Distance in Pitch Class Profile Computation

Giordano Cabral[1], Jean-Pierre Briot[1], François Pachet[2]

[1]Laboratoire d'Informatique de Paris 6 – Université Pierre et Marie Curie
8 Rue du Capitaine Scott 75018 Paris – France

[2]Sony Computer Science Lab Paris
6 Rue Amyot 75005 Paris – France

{Giordano.CABRAL,Jean-Pierre.BRIOT}@lip6.fr, pachet@csl.sony.fr

Abstract. *Pitch Class Profiles (PCP) [Fujishima 1999] are largely used for all applications involving harmonic content. Although the main steps of the PCP calculation are generally equal over all the scientific literature, some implementation details may vary, specially the impact of the distance between the frequency of the FFT bins and their closest note. This paper compares 6 ways of using this distance to weight the contribution of each FFT bin in the final PCP vector. We present the results of the 6 computed PCPs when used in a chord recognizer, a tonality estimator and a key detector.*

1. Introduction

Pitch Class Profiles (PCP) are vectors of low-level instantaneous features, representing the intensity of each of the twelve semitones of the tonal scale. They are largely used in all applications involving harmonic content, especially chord recognizers [Yoshioka at al. (2004)], tonality estimators [Gómez and Herrera 2004a] and key detectors [Pauws 2004]. The main advantages of the PCPs are the simplicity of calculation, the concision of the harmonic information and the power to unify various dispositions of a single chord class. Although the main steps of the PCP calculation are very similar over all the scientific literature, some implementation details may vary. Among them, is especially imprecise how the distance from each FFT bin to its closest note may contribute to the quality of PCP. This paper intends to contribute to the discussion on the subject, comparing 6 functions (uniform, discrete, linear, anti-quadratic, exponential, and gaussian) using the above mentioned distance to change the PCP computation. To test the PCPs, we have implemented a chord recognizer, a tonality estimator, and a key detection system, as described in main publications: given a labeled database of sound recordings, the respective PCPs are computed, and a classification algorithm, such as k-nearest neighbors [Mitchell 1997] or hidden markov models [Sheh and Ellis 2003] is ran. We assume that the higher the quality of a classifier using a specific PCP calculation method, the greater the precision of the PCP information. In other words, the number of correctly classified instances may serve as a comparative measure of precision for the PCP vectors, as illustrated in the end of this paper.

Next section describes how to compute the PCP vectors. Section 3 exposes the 6 proposed functions for weighting by distance. Section 4 explains the experiment in more details, the used datasets and the applied methodology. Section 5 presents the absolute and relative results, and section 6 draws some conclusions and future work.

2. PCP Computation

PCP vectors are computed by mapping each frequency bin of the spectrum to a pitch class (one of the 12 notes of the tonal scale). Figure 1 illustrates the main steps in the PCP computation. The sound in Figure 1a is converted to the frequency domain by means of some Fourier-Transform (FFT) [Orfanidis, S. 1995]. Each FFT bin is mapped to its closest note (e.g. FFT bins corresponding to frequencies like 433 Hz, 438 Hz, or 443 Hz are mapped to the A at 440 Hz). One may see such mapping as the division of the spectrum into regions, as shown in Figure 1c. Then, the amplitudes inside each region are summed up and divided by the number of bins inside the region, resulting in a histogram as in Figure 1d. Finally, the histogram is folded, collapsing pure tones of the same pitch class, despite the octave, to the same chroma bin, resulting in a 12-sized vector, where each index represents the intensity of one note.

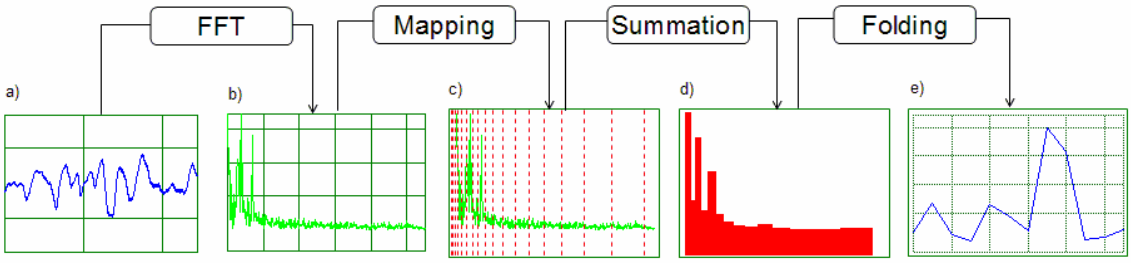

Figure 1. PCP computation steps.

Formally, we can formulate the PCP as in equation 1. N is the number of samples in the sound (or the number of bins the FFT), k is a bin in the FFT (where $0 \leq k \leq N-1$), f_{ref} is the reference frequency corresponding to PCP[0], and f_{sr} is the sampling rate. Normally, the value of each PCP element is calculated by summing the magnitude of all frequency bins that correspond to a particular pitch class (i.e. p = 0, 1, …, 12), as shown in Equation 2.

$$p(k) = \lfloor 12 \cdot log_2(k/N \cdot f_{sr}/f_{ref}) \rfloor \; mod \; 12$$

Equation 1. Mapping from frequency bins to PCP bins.

$$PCP_{[p]} = \sum_{k:p(k)=p} |X[k]|^2$$

Equation 2. Calculation of the values of the PCP elements.

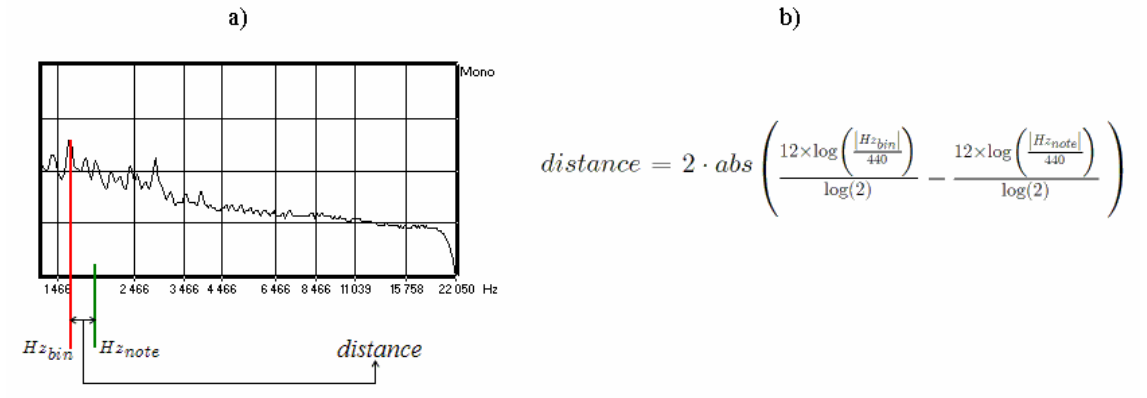

Figure 2. Distance between the frequency of the FFT bin (Hz_{bin}) and the frequency of its closest note (Hz_{note}).

However, as in most cases the frequencies of the FFT bins do not match the frequency of a note, one can imagine that this summation may take into account the distance between them (Figure 2). Figure 2b shows the formula of such distance, which is proportional to the size of the region. It varies from 0 (the FFT bin has exactly the same frequency of a note) to 1 (exactly in the middle of 2 notes). This work intends to evaluate the different methods using this distance to weight the summation, in order to improve the quality of the classifiers over such PCPs, with regards to precision and robustness.

3. Weighting functions

We are interested in the possible functions *f*(distance) (Equation 3) that could be used to weight the summation of values inside each region. These functions must be applicable for all values in the range 0:1, and must also return values in the range 0:1. For example, f(x) = 1/x or f(x) = log(x) would not be allowed since they are not defined for x=0.

$$PCP_{[p]} = \frac{\sum_{k:p(k)=p} |X[k]|^2 \cdot f(distance)}{\sum_{k:p(k)=p} f(distance)}$$

Equation 3. Weighted summation using the distance to the closest note.

We took into consideration 6 functions: uniform, discrete, linear, anti-quadratic, exponential, and gaussian. The uniform function returns always 1. It means that the distance does not affect the result. This is the simpler and most used method. Discrete weighting will only consider the frequencies inside a narrower region (in Figure 3, distance must be smaller than 0.2). In the other 4 functions, the weight of the element will gradually decrease as the frequency goes farther, but the respective curves have different shapes. The graphs of these functions are shown in Figure 3, as well as their formula.

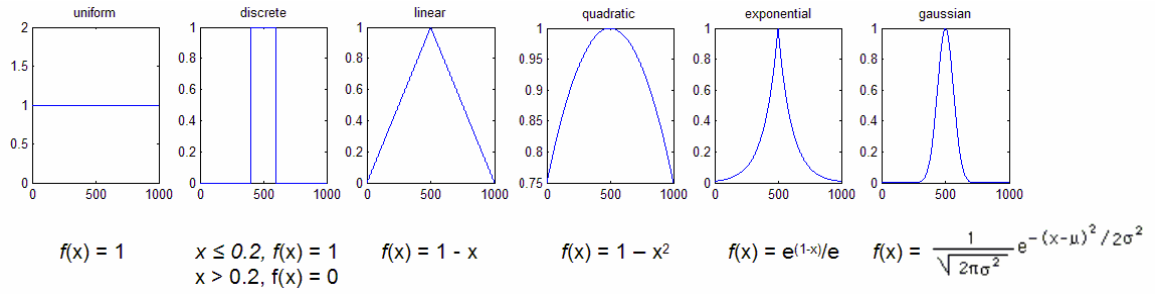

Figure 3. Weighting functions (uniform, discrete, linear, anti-quadratic, exponential, and gaussian).

4. Chord Recognition, Tonality Estimation, Root Detection

In order to evaluate the different weighting functions, we re-implemented some applications using PCP: a chord recognizer, a tonality estimator, and a key detector. All of them share the same architecture: a database of sound recordings is labeled with the expected class for each example. In the case of the chord recognizer, the class is the expected chord (key and type); for the tonality estimator, the expected tonality (key and

mode); for the key detector, just the key. Then, the PCP of each example is calculated, and used as input to some machine learning algorithm [Mitchell 1997] in order to automatically generalize a classifier. In other words, these algorithms try to automatically learn from the examples the patterns of the PCPs of each chord, so it will be capable to give answers to new examples.

As suggested by [Gómez and Herrera 2004b], we used a KNN classifier over the PCPs of the sound recordings. The data was taken from D'accord Guitar Chord Database [Cabral et al. 2001], a guitar midi based chord database. Each midi chord was rendered into a wav file using Timidity++ and a free nylon guitar patch. The richness of the symbolic information present (chord root, type, set of notes, key, position, fingers, etc.) allowed us to automatically label the data and create some databases. The first one (MajMin) is a tonality estimator-like database, restricted to deal only with Amaj and Amin chords. The second one (ChordC) constrained the root to be C, trying to automatically learn to classify the type (Maj, Min, Dom7, Min7, Dim). The third one (Root) tried to determine the root of a chord. The forth one (Chord) is the chord recognizer. It tries to learn what is the chord (i.e. which root and type simultaneously). The fifth one (RealTest) is the same chord recognizer, but tested with real sound recordings, instead of synthetic sounds. 80% of each database was settled on as the training dataset and 20% as the testing dataset.

Table 1. The databases corresponding to the 5 experiments.

Database	Root	Types
MajMin	Fix (A)	Maj, Min
ChordC	Fix(C)	Maj, Min, Dom7, Min7, Dim
Root	Variable (C, C#, …, B)	Fix
Chord	Variable (C, C#, …, B)	Maj, Min, Dom7, Min7, Dim
RealTest	Variable (C, C#, …, B)	Maj, Min, Dom7, Min7, Dim

5. Results and Discussion

Table 2 and Figure 4 show the results of the classifiers by weighting function and by experiment. For the first, trivial, problem of separating the major and minor chords (with a synthetic database of fix root chords), all weighting functions worked satisfactorily. For the second experiment, such of finding the chord type, the simple discrete weighting surprisingly surpassed all others. For the third experiment, such of finding the root of a chord, regardless its type, the exponential and gaussian worked slightly better than the others. For the forth problem, such of chord recognition (the one we were most interested in), all but the uniform function got close values. Finally, the last experiment, dealing with the same problem of chord recognition but tested with a dataset of recorded audio, showed the most discrepant results. In fact, real recordings may have differences in tuning, which affect significantly the precision of the algorithms, especially those that abruptly increase or decrease, such as the discrete and gaussian functions. These functions leak in robustness, since they are extremely dependent to a good tuning.

On the other hand, a comparative analysis shows that a part from the dependency to the quality of the tuning, the weighting functions do not present significant disparities. Figure 5 shows such an analysis, in which the results of each experiment are proportional (i.e. each value is divided by the maximum). We can see, for example, that

the simpler and normally worst solution (uniform) is always at least 90% as good as any other method.

Table 2. Results of the classifiers using the different weighting functions in each experiment.

Name	MajMin	ChordC	Root	Chord	RealTest
Uniform	100,00 %	84,85 %	73,74 %	71,35 %	65,38 %
Discrete	100,00 %	93,94 %	78,51 %	77,45 %	34,62 %
Linear	100,00 %	84,85 %	77,98 %	76,39 %	69,23 %
Anti-Quadratic	100,00 %	84,85 %	77,72 %	76,39 %	65,38 %
Exponential	100,00 %	87,88 %	80,37 %	79,31 %	61,54 %
Gaussian	100,00 %	87,88 %	80,11 %	79,31 %	42,31 %

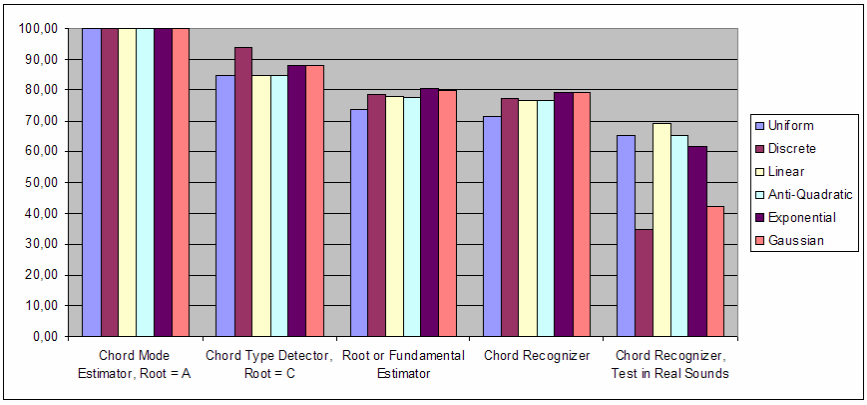

Figure 4. Results of the classifiers using the different weighting functions in each experiment.

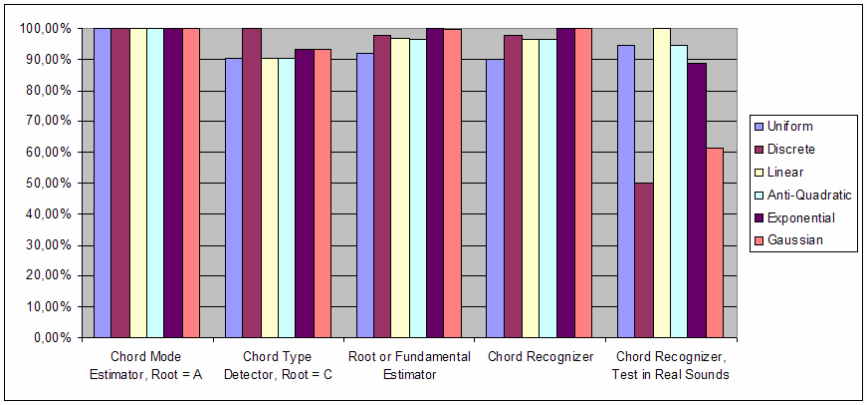

Figure 5. Comparative results (each value is divided by the maximum value in the experiment).

6. Conclusions and Future Work

Given the results, we can draw a few conclusions: 1) the weighting functions do not affect significantly the quality of the PCP; 2) discrete and gaussian weightings are not robust; 3) there is no absolute "winner". Nevertheless, Table 3 suggests the use of some functions, depending on the interest of the developer. If he wants a simple solution, no weighting (uniform) works satisfactorily. Otherwise, the linear, discrete, exponential,

and gaussian works a little better. Additionally, if the tuning of the sound samples is not guaranteed, he should better choose the linear weighting.

Table 3. Best weighting function, given the interests of the developer.

	Robust	**Good tuning guaranteed**
Light, simple	Uniform	Discrete
Efficient	Linear	Discrete/Exponential

Some improvements can be done in future works, specially the conversion of the datasets to real recordings. The addition of hybrid methods, such as the discrete+linear, would also be interesting. Finally, the comparison with other existent variations of PCP extraction algorithms is strongly desirable, in order to delineate a standard, well defined algorithm.

7. Acknowledgements

We would like to thank all the team from CSL Sony in Paris.

This research is supported by CAPES/COFECUB, Brazil/France.

References

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

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

Gómez, E. and Herrera, P. (2004a) "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.

Gómez, E. Herrera, P. (2004b) "Automatic Extraction of Tonal Metadata from Polyphonic Audio Recordings", Proceedings of 25th International AES Conference, London.

Mitchell, T. (1997) "Machine Learning", The McGraw-Hill Companies, Inc.

Orfanidis, S. (1995) "Introduction to Signal Processing", Prentice-hall.

Pauws, S. (2004) "Musical key extraction from audio", Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR04), Barcelona.

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

Yoshioka, T., Kitahara, T., Komatani, K., Ogata, T., and Okuno, H.-G. (2004) "Automatic Chord Transcription with Concurrent Recognition of Chord Symbols and Boundaries", Proceedings of 5th International Conference on Music Information Retrieval (ISMIR 2004), Barcelona.