A Framework for Real-time Instrumental Sound Segmentation and Labeling

Adriano Monteiro
Interdisciplinary Nucleus of Sound Communication, University of Campinas
Campinas, Brazil
monteiro.adc@gmail.com

Jônatas Manzolli
Interdisciplinary Nucleus of Sound Communication, University of Campinas
Campinas, Brazil
jotamanzo@hotmail.com

Abstract

This paper presents a collection of Pure Data abstractions for real-time transcription of audio signals produced by musical instruments. Initially we describe the adopted methodology follow by its implementation. Finally we show an application in a computer-assisted improvisation system.

Keywords

Automatic Transcription, Musical Information Retrieval, Assisted-Improvisation

1. Introduction

The aim of a musical transcription process is to extract symbolic information from an audio signal in order to find high-level musical structures such as those in a traditional music scores [1]. Further in western noted music, a complete transcription requires solving pitch, timing, and instrumentation of all sound events [2]. The first step in this process is called segmentation that determines the time boundaries of each note-event in a continuous digital signal by locating onsets and durations. Next, a labeling task is required to extract the perceptual musical content from each event such as pitches and timbre. Finally, the task of the highest symbolic level is to extract relationships among labeled events, such as, melodic profile, rhythmic patterns, harmonies and textures.

According to [2] [3] a music transcription system can be applied to: a) music information retrieval; b) music processing - parameters of music and audio effects controlled in an adaptive fashion; c) music-related equipment - e.g. light and video effects controlled in real time; d) musicological analysis; e) transcription tools for amateur musicians; and f) human-computer interaction - such as a system for co-improvisation, generative composition, score-following and accompaniment.

Here we are going to focus topic (f). We describe an application in the context of Computer-Assisted Improvisation Systems (CAI) as defined in [4]. The principle of CAI is to develop computer models of human musical cognition to assist musician's decision (computer player) during a musical improvisation. The cognitive models might include: memory and its retrieval, sound analysis for multi-sensorial reinforcement of the acoustic information, musical solution/continuation/response suggestion.

In this article we concentrate on tools for musical transcription of audio signals produced by musical instruments. Our main interest is to study free-improvisation between an acoustic instrument player(s) and a computer player. In the next section we review Pure Data implementations for automatic transcription: a) onset detection and note segmentation [3],[5]; b) harmonic feature extraction [6]; c) recognition of percussion instruments [7] and d) algorithms to handle the musical data obtained by (a),(b) (c). In the following section we present our application for CAI [8] as part of a memory/information-retrieval structure in real time. All implementation presented here is included in the PDescriptors library[1]. This library is a sub-product of our research and it is a set of Pure Data abstractions for audio feature extraction developed using the BSP technique [9].

2. Instrumental Sound Transcription

This section presents the methods we adopted for music transcription.

2.1 Segmentation

A commonly used approach to segmenting musical events from an audio signal is to detect the onset and offset time. The onset detection method adopted here is mainly based on [3]. It is divided into two steps: 1) the extraction of a

[1] https://sites.google.com/site/pdescriptors/

detection function which reduces signal information and highlights onset moments and 2) a peak picking algorithm applied to the detection function.

According to the proceedings of MIREX[2], detection functions extracted from the spectral domain of an audio signal have successful results and are the most widely used functions in onset detection methods. Based on [5] we chose the detection function based on the spectral difference with half-wave rectification (spectral flux) for our Pure Data implementation. We use this solution due to its simplicity, good results and low computational cost. In order to achieve good temporal resolution, it is usually computed on FFT windows varying from 512 to 2048 samples

The peak picking process is based on an adaptive threshold described in [3]. It is composed of three steps.

First a low-pass filter is applied to smooth the detection function and prevent some spurious peaks. Next, a dynamic threshold is computed over a small number of samples, including past and future samples, to compensate pronounced amplitude changes in the function profile. It is composed of: a moving median filter that reduces noise and selects the larger amplitude peaks; and a moving mean filter that follows the amplitude variation and smoothes the function profile. The moving mean compensates the absence of DC-removal and normalization processes present in the off-line version of the algorithm [10]. Finally, the local maximum over the threshold is computed from at least three consecutive samples also taking into account past and future data.

To detect offsets we used an RMS function which is defined by the energy mean of a signal window in the time domain. After an onset has been detected, the offset position is selected when the value of the signal amplitude decreases below a threshold determined by the user. The algorithm schema is shown in Figure 1.

The onset and offset time position returned by the above algorithms are related to an analysis

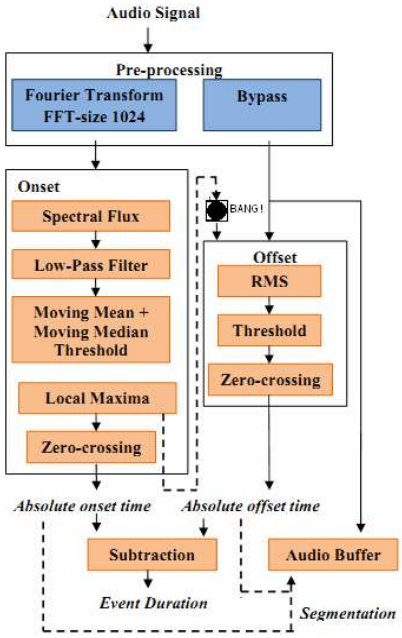

Figure 1- Segmentation algorithm schema

window. Since it has no information in regards to the sample amplitude levels in a current window, it can cause clicks if used as a parameter of an event player algorithm. To solve this problem and to improve the accuracy of the segmentation point, we implement an algorithm that searches for a zero-crossing point in an audio buffer. The search is done in a decremental fashion starting from the last sample of the onset/offset window detected and stopping when the first zero-crossing is found.

Finally, the event duration is given by the subtraction of the offset time by the onset time.

2.2 Harmonic Features

For harmonic analysis we used the extraction of the *chroma* feature, also commonly referred to as the *Pitch Class Profile*. It is a vector in which the values represent the spectral energy distribution in the pitch classes according to the octave division of Western music. We chose a quarter-tone octave division that corresponds to a 24-dimensional *chroma* vector.

The conception behind *chroma* feature extraction is based on observations related to the *circularity* of the human pitch perception. Two pitches with different *tone heights* (i.e., in different octaves) share the same pitch class. The *tone height* attribute is related to the octave number and the *chroma* attribute is associated to the pitch spelling of musical notes (C, C#, D… B).

A *chroma* vector is obtained from the spectral domain of a sound signal by summing up the

[2]http://www.music-ir.org/mirex/wiki/MIREX_HOME

energy contained in the sub-bands that correspond to the same pitch class. Better results are found when it is computed using large FFT windows with 8192 samples, for example. The spectral division in sub-bands is applied in the power spectrum where the most significant partials can be found. We adopted the Jehan [6] solution that proposed six octaves ranging from C1 = 65Hz to B7 = 7902 Hz. Next, a Hanning windowing is applied to the bins included in each sub-band with the window centered within each respective pitch frequency. The sub-bands are summed and the resultant 24-element vector is normalized by dividing each of its elements by the maximum element value. A *chromogram* is constructed by a normalized histogram of successive *chroma* vectors. Finally, these values are exponentiated and the values above the mean of the total energy are selected. The algorithm returns a 24-element binary vector that represents the presence or absence of the pitch-classes. The algorithm schema is shown in Figure 2.

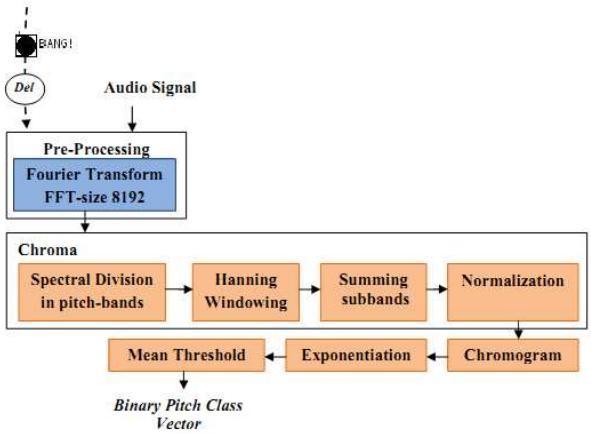

Figure 2 - Data flux representation of the harmonic classification algorithm.

The *chroma* extraction is triggered by the segmentation algorithm. It can be delayed for some signal blocks to avoid possible noise insertion to the pitch content caused by analyses of attack transients. Analysis parameters are set by the user, such as, the frequency range of the spectrum to analyze, the number of successive *chroma* vectors to consider and the number of analysis windows to wait after onset detection.

2.3 Percussion Instrument Recognition

Brent [7] presents an efficient approach to recognize percussion musical instruments. This process requires an off-line training step. A data set is collected from a sequence of percussion sound events. The sound events are the strikes of the percussion instruments and the data set consists of one or many spectral features extracted from the analysis frame (or frames) after the attack peak. Brent's criteria for instrumental choice were: diversity of material, diversity of spectrum and relatively short decay.

A delay after the attack is introduced to reduce the presence of any pre-onset resonance in the analysis window. The matching between the training data set and incoming data is computed by the Euclidian Distance.

Brent [7] conducted experiments to test the performances of several features. The best scores were found in cases where the BFCC feature (Bark Frequency Cepstral Coeficients) was used. One hundred percent accuracy was achieved by combining BFCC with other features for all delay-time tests (ranging from 0ms to 18.87 ms in steps of 1.45 ms) and in cases of some successive-frames analyses.

Considering the low computer cost, we implemented a BFCC algorithm. In Brent's experiments it also achieved 100% of accuracy for 13.06 ms, 15. 96 ms and 17.42 ms delay times in the single-frame analysis case and for 9 of 10 delay times below 13.06 ms, in the successive-frames analysis case. The algorithm is described in Figure 3.

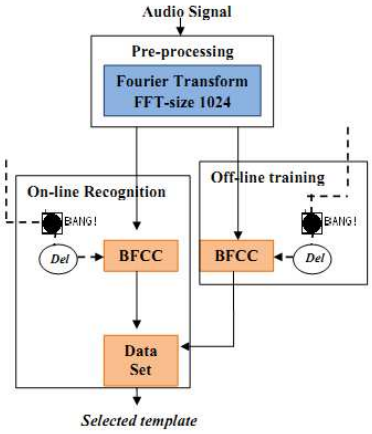

Figure 3- Timbre recognition algorithm data flux representation

The bonk~ object used by Brent for onset detection and BFCC triggering was replaced by the algorithm described in section 2.1.

2.4 Implementation

The algorithms above were implemented as Pure Data abstractions. There are common types of arguments for the three objects: FFT-size, Hop-size (like in Pd switch~ and block~ objects) and an object name. The objects must share the same name in order to onset trigger command (bang) to be sent by the segmentation algorithm to the harmonic and timbre classifiers. A graphical

interface version was also implemented (Figure 4) that provide icons for parametrical interface as well icons for data display.

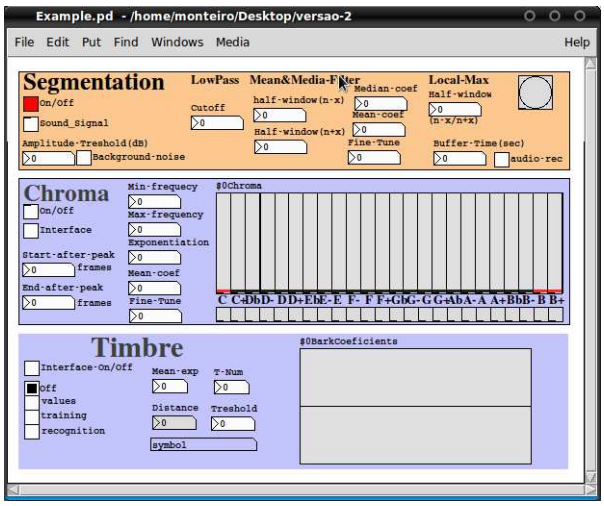

Figure 4 - Transcription objects interface

2.5 Data Handling

This is a collection of seven Pure Data abstractions that provides memory and selection of the data extracted by the transcription algorithms:

- a dynamic resizing buffer for symbolic data;
- an object which accesses slices of the symbolic data buffer
- a silence filter;
- a duration filter with two operation modes:
    - filtering duration values above or below a threshold determined by the user;
    - Selecting specific duration values determined by the user.
- a harmonic filter with two operation modes:
    - filtering *chroma* vectors containing the pitches set by the user.
    - selecting the exact *chroma* vectors set by the user.
- A percussion timbre selection algorithm that selects similar events compared to a target event.
- A simple player with four operation modes related to the event permutation: sequence player, Brownian movement player, serial player, and a random player. It plays slices of an audio buffer according to the onset-offset times recorded in the symbolic data buffer.

The data buffer must be connected with the segmentation algorithm shown in section 2.3. The other objects send and receive lists of data indexes in the buffer and can be connected in a sequence where the slicer algorithm is first and the player is last.

3. Application on CAI

In this section we show a musical application in computer-assisted improvisation (CAI) of the tools described above.

3.1 System Overview

The compositional system was conceived as an *algorithmic composition* and a *computer-assisted improvisation system* for a piece including computer and electric bass. The piece was called “As Duas Criaturas que Estavam a Mesa de Chá não Tiveram esta Conversa” (The Two Creatures on the Tea Table Didn’t Have this Conversation) and was presented during the 3rd PDcon, July 2009 in São Paulo.

The system is constructed in four algorithmic modules for sound generation by synthesis and/or sound processing of the bass. The modules are:

- Distortion: a wave-shaping distortion is applied to the bass sound using a signal function as a transfer function;
- Pitch-shift: a pitch- shifter effect is applied to the bass sound;
- Phaser: a phaser effect is applied to the bass signal and to clipped sinusoidal sounds.
- Attractor: values resulting from iterations of a non-linear function described in [11] as the Latoocarfian attractor are stored in an array. It is read as a waveform in a synthesis process and applied as a transfer function in a wave-shaping process to the bass sound. The waveforms generated by the attractor function always converge for variations of six main types: simple periodicity, complex periodicity, random (noise), pulse train and constant zero (silence).

In the first version of the algorithm, most of the controlling parameters are directly changed through icons in the graphical interface (GUI) (Figure 4). Serial and random permutations are also applied in the fourth module to control some parametric changes. A deeper view of the algorithm properties can be seen in [8]

3.2 Architecture of the New Version

In the second version of the piece new tools were developed in order to: 1) increase the interaction between bass player and the computer player/system; 2) provide access to performance

memory through a high-level symbolic layer related to musical features and system parameters.

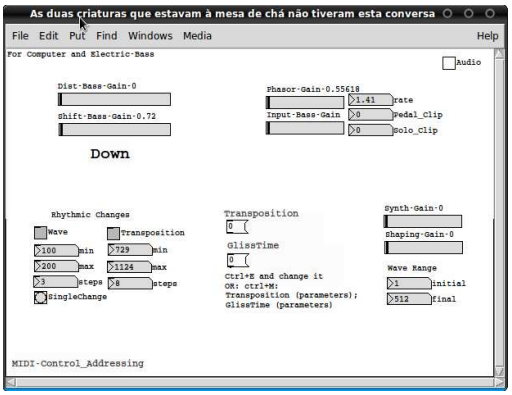

Figure 5 - GUI of "The Two Creatures on the Tea Table Didn't Have this Conversation (v.1)"

In Figure 5 we show the new system architecture. The yellow rectangles represent parts of the system already present in version 1. The blue rectangles represent the parts of the system related to the development presented in this paper. Temporal and pitch data extracted from the bass signal are recorded on-the-fly. It can be accessed and modified by the performer through the data handling objects then applied to the synthesis parameters in the attractor module

The retrieved values, when applied to the synthesizer are feedback to the memory with new values extracted from the bass

The red rectangles represent a method for analysis and visualization of times series related to spectral features on the Poincaré Maps that we also implemented in the new version. We applied it as a system to retrieve information describing recursive dynamics of spectral features within sections of the piece. This approach was used in order to assist the computer player decisions. The discussion of this method is not within the scope of this work. Further information can be seen in [12] and [13].

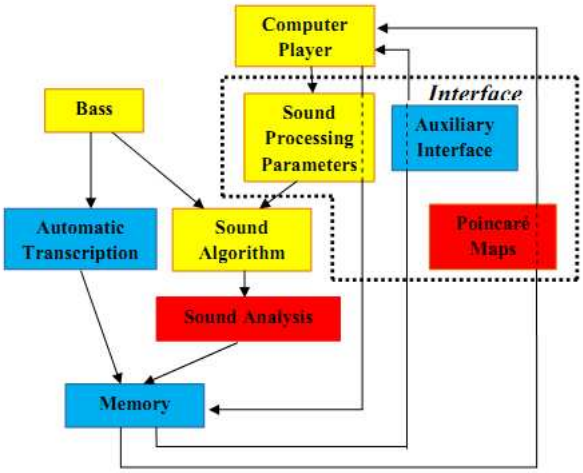

Figure 6 - Data flux representation of the "The Two Creatures on the Tea Table Didn't Have this Conversation (v.2)".

The Figure 6 shows the graphical interface of the new system (version 2) where the central window is the main system interface inherited from the previous version.

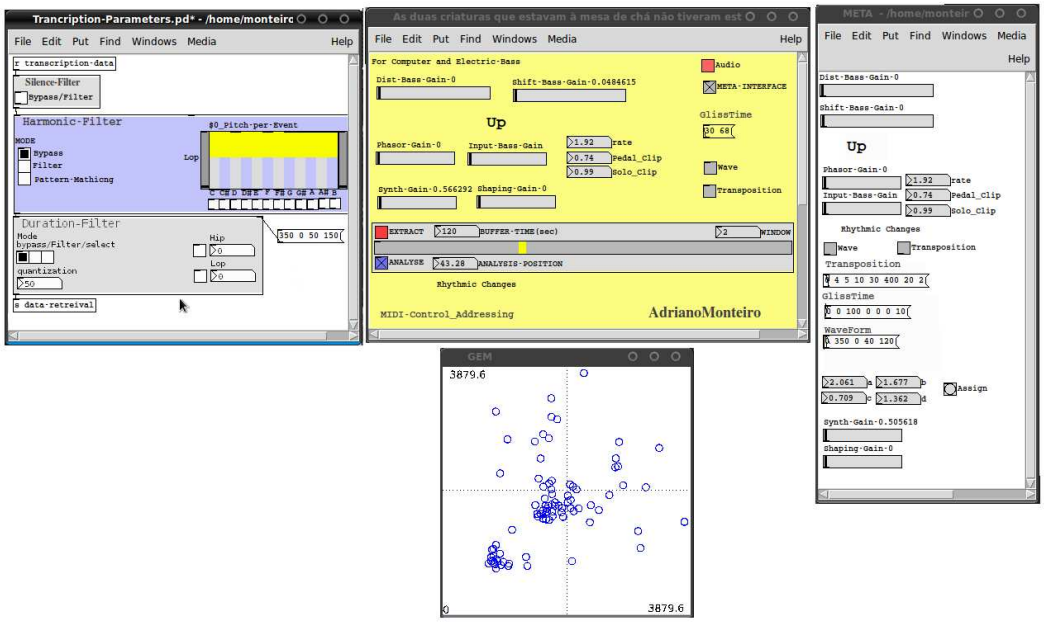

Figure 7 - GUI of "The Two Creatures on the Tea Table Didn't Have this Conversation(v.2)"

The left and right windows are respectively the interface for manipulation of data extracted from the bass sound and a display showing the memory of system parameters. The bottom window is a Poincaré map which plots a recursive spectral analysis of memory slice (section) selected by the user.

4. Conclusion

In this paper we presented Pure Data implementations of methods for music transcription. We showed examples of its application as a structure for memory and retrieval of musical information in a computer-assisted improvisation system. Next steps in this work include implementation of other methods for music transcription and methods related to stylistic music generation to be used for solution/response suggestion mechanisms in our improvisation systems.

Acknowledgments

We thank the Brazilian agencies FAPESP and CNPq for supporting Monteiro and Manzolli.

References

[1] Scheirer, E. “Extracting Expressive Performance Information from Recorded Music”. Master’s Thesis, MIT, 1995.

[2] Klapuri, A., & Davy, M. ed. “Signal Processing Methods for Music Transcription”, Springer Science+Business Media LLC, New York, USA, 2006.

[3] Brossier, P. “Automatic Annotation of Musical Audio for Interactive Applications”, PhD Thesis, Queen Mary University of London, 2006.

[4] Maniatakos, F., Assayag , G., Bevilacqua, F., Agon, C. “On Architecture and Formalism for Computer-Assisted Improvitsation” In: http://articles.ircam.fr/textes/Maniatakos10a/index.pdf

[5] Dixon, S. “Onset Detection Revisited” Proc. of the 9th Int. Conf. on Digital Audio Effects (DAFx-06)

[6] Jehan, T. “Creating Music by Listening” PhD Thesis, MIT, 2006.

[7] Brent, W. “Cepstral Analysis Tools for Percussive Timbre Identification” In Proceedings of the 3rd Pure Data Convention, Sao Paulo, 2009.

[8] Monteiro, A.C. Manzolli J., “Estudo de Performance e Interação Utilizando Processamento em Tempo Real”, Proceedings of Performa’11, Aveiro, 2011

[9] Barknecht, F.. “Applications of Blocked Signal Processing (BSP) in Pd” In Proceedings of Linux Audio Conference 2010 (LAC2010), Ultrecht, the Netherlands, 2010.

[10] Bello, J.P. Daudet, L., Abdallah, S., Duxbury, C., Davies, M., Sandler, M.B. “A Tutorial on Onset Detection in Music Signals” IEEE Transactions on Speech and Audio Processing, 13, 1035–1047, 2005.

[11] Pickover, C.A. “Chaos in Wonderland: Visual Adventures in Fractal World”. St. Martin's Press, New York, USA. (1994)

[12] Monteiro, A.C. Manzolli, J., “Análise Computacional de Texturas Sonoras via Mapas de Poincaré” proceedings of XXI Congress of ANPPOM, Uberlândia, 2011.

[13] Monteiro, A.C., Manzolli, J., “Análise de Áudio e recuperação da Informação Musical em um Ambiente Computacional Voltado a Improvisação, proceedings of 13th Brazilian Symposium on Computer Music (SBCM), Vitória, 2011.