Multimodal Guitar: A Toolbox For Augmented Guitar Performances

Loïc Reboursière [2], Christian Frisson [1], Otso Lähdeoja [3],
John Anderson Mills III [2], Cécile Picard [4], Todor Todoroff [2]
[1] TCTS Lab, Université de Mons, 31 Boulevard Dolez, 7000 Mons, Belgium;
[2] TELE Lab, Université catholique de Louvain, Place du Levant 2, 1348 Louvain-la-Neuve, Belgium;
[3] CICM, University of Paris 8, MSH Paris-Nord, 4, rue de la croix faron, 93210 St Denis, France;
[4] REVES / INRIA, 2004 route des lucioles, BP 93, 06902 Sophia Antipolis, France
loicreboursiere@gmail.com, christian.frisson@uclouvain.be, otso.lahdeoja@free.fr,
nodog@konfuzo.net, cecile.picard@sophia.inria.fr, todor.todoroff@skynet.be

ABSTRACT

This project aims at studying how recent interactive and interactions technologies would help extend how we play the guitar, thus defining the "*multimodal guitar*". Our contributions target three main axes: audio analysis, gestural control and audio synthesis. For this purpose, we designed and developed a freely-available toolbox for augmented guitar performances, compliant with the PureData and Max/MSP environments, gathering tools for: polyphonic pitch estimation, fretboard visualization and grouping, pressure sensing, modal synthesis, infinite sustain, rearranging looping and "smart" harmonizing.

Keywords

Augmented guitar, audio synthesis, digital audio effects, multimodal interaction, gestural sensing, polyphonic transcription, hexaphonic guitar

1. INTRODUCTION

The evolution of the guitar as a musical instrument has benefitted from advances in communication technologies: vacuum tube amplification, electronic diodes and chips, magnetic, piezoelectric or optical sensing, wireless linking, and so on [3].

The "guitar synthesizer", which extends the palette of guitar sounds with synthesis algorithms and effect processing, is composed of a guitar, a monophonic or hexaphonic pickup (the latter allowing signal analysis of individual strings, with either magnetic [9], optical [5], or piezoelectric sensing) and an analog or digital processing device (the latter utilizing a microprocessor).

Recently, people have been extending the real time processing and sound possibilities of their monophonic or hexaphonic guitar (e.g, [10]) by processing their sound using modular environments such as PureData[1] and Max/MSP[2].

[1]http://www.puredata.info
[2]http://www.cycling74.com

NIME2010, 15-18th June 2010, Sydney, Australia

Additionally, the multiple sensing methods available at low cost nowadays, from remote cameras [6, 2] to built-in accelerometers and other sensors [4] enable guitarists to emphasize their musical expression and to use gestures as a way of controlling and producing sound, making the guitar an "augmented guitar" [4].

Our purpose, with this toolbox, was to start making and gathering modules to enhance performances. The tools we worked on range from the sound analysis, synthesis and processing of the monophonic or hexaphonic guitar to the use of gestural expression. Tests still need to be done to have users feedback.

We are providing a collection of open-source patches and objects for both Max/MSP and PureData environments, available for download on the numediart[3] website. The developed externals (fretboard visualization and grouping, modal synthesis) have been written using either the java language using pdj[4] developed by Pascal Gauthier or the C++ language with a flext[5] layer by Thomas Grill. Both of pdj and flext enable one to develop objects working on Max/MSP and PureData with the same code.

For longer description, please have a look at the numediart report of the eNTERFACE'09 summer workshop during which most of this toolbox was designed.

2. METHOD AND IMPLEMENTATIONS

Building an augmented guitar can be considered through several parts:

- Audio analysis : how to use features of the guitar sound to detect events or to control parameters
- Gestural control : how to use movements made by the guitarist to add control on the sound produce by the computer
- Audio synthesis : how can we enhance guitar performance with relevant effects (monophonic and hexaphonic)

To develop and test these tools we used a Fender Stratocaster guitar with a Roland GK3 hexaphonic pickup[1] mounted on it and a StringPort interface made by Keith McMillen[2]. The FSR pressure sensors and the MIDI sensors interface we used were designed by interface-Z[6].

[3]http://www.numediart.org/download
[4]http://www.le-son666.com/software/pdj/
[5]http://puredata.info/Members/thomas/flext/
[1]http://www.roland.com
[2]http://www.keithmcmillen.com
[6]http://interface-z.com

2.1 Audio Analysis

2.1.1 Fretboard Visualization

Before starting the audio analysis, we needed a quick and easy integrated tool to visualize the guitar fretboard including its physical parameters (number of strings, number of frets and tuning), the groups the user is working with and the played notes detected by the polyphonic pitch estimation. Several parameters are therefore customizable for the fretboard's display:

- the guitar parameters (as mentioned above) : number of strings and number of frets. The tuning parameter doesn't have a direct influence on the display, but enables notes to be displayed accurately.
- the display parameters : strings spacing and fret spacing factor. These factors can easily lead to a non realistic representation of the fretboard, but can be helpful to see notes clearly, especially in the highest frets.

2.1.2 Polyphonic Pitch Estimation

One of the great interests of using a hexaphonic pickup in conjunction with Max/MSP or PureData is that one can have an individual control (processing and analysis) on several or all of the strings of a guitar at the same time. Detection and processing of chords then become possible with a set of accurate monophonic pitch estimators.

A first version of our polyphonic pitch estimation tool was made using both Puckette's `sigmund~`(which he suggested to use rather than `fiddle~` [10]) and IRCAM's `yin~` implementation for Max/MSP, to compare a frequency-based and a time-based algorithm respectively. The final version uses only `sigmund~` as it is freely available for both Max/MSP and PureData and because the two algorithms gave quite similar results for our purpose. After having run several tests, it appears that choosing the `sigmund~` window size closest to two periods of each of the open strings frequencies gives good results in terms of accuracy and latency. The overlap parameters were set to a third of the window size values.

During our research, it appeared that detecting the played note correctly was not the only issue of our algorithm. The other one was to detect the end of a note (e.g. if you want to add a specific effect until the end of the played note). We decided to define the end of the note as going under a adjustable threshold level so that when the played note is considered by the player to be over, then it is considered over for the tool as well.

2.1.3 Fretboard Grouping

The polyphonic pitch estimation presented in section 2.1.2 allows one to go further in the segmentation of the guitar fretboard by giving a per fret control. The idea of managing groups of frets, and not only strings then quickly emerges.

We can find a similar idea in the software packaged with the hexaphonic-to-MIDI hardware converter Axon 50 USB[7]. In this software, one can create zones by splitting the fretboard by string, fret and/or pickup. The created zones can then only be rectangular and seem, after few tests, to be difficult to use.

The tool we made goes further with this idea of making zones on the fretboard. Instead of talking about zone, we would rather use the more general idea of "group" : a group is defined by any set of notes in space and time. Each time a note is played, the external checks if it belongs to a group and reports it. If a group is detected as having been entirely played, the externals report it, as well, and resets the memorized previous incoming notes. Therefore, the term group can handle : a scale, a chord, an arpeggio or even the region including the first two frets of the first two strings. This tool can be used in two major methods. The first can be seen as defining group's behavior by routing the notes of a specific group to a given effect, processor, etc. The second is event detection, e.g, the whole group was played or a note of that or this group has been played.

[7]http://www.axon-technologies.net

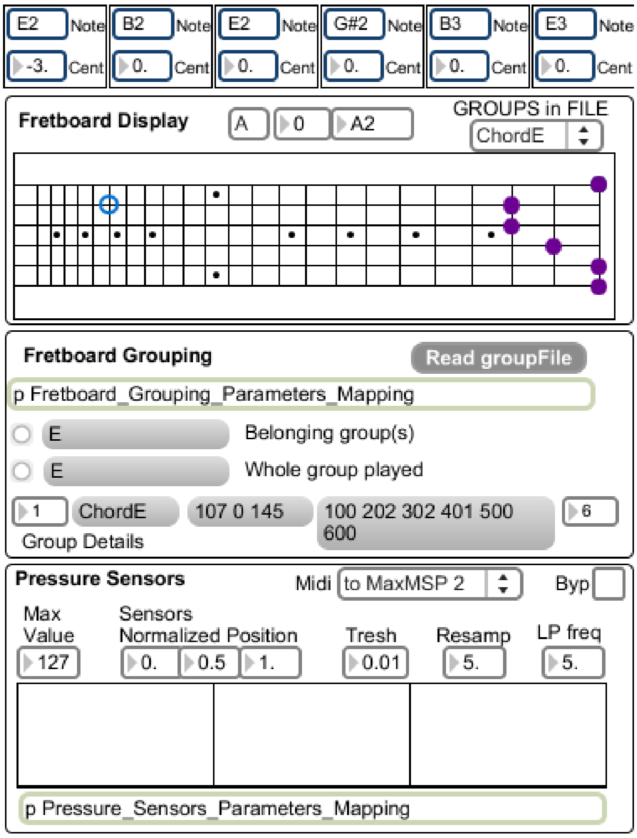

Figure 1: Multimodal Guitar Audio Analysis and Gestural Control Modules: Recognition of the E chord.

2.2 Gestural Control: Rear-Mounted Pressure Sensors

The basic premise of this gestural control research was to add pressure sensors to the back of an electric guitar in order to add an expressive control to the sound using the natural movement of the guitarist.

Due to familiarity, and to amount of covered surface, we chose the Interface-Z 4 cm square FSR pressure sensors to use for this implementation. After some testing, it was determined that these sensors may have been more sensitive than necessary for this application, and the 1.5 cm round FSR pressure sensors might have been a better choice for both sensitivity and placement on the guitar body. This will need further testing in the future.

An array of pressure sensors was chosen as the interface to use instead of a single position sensor, because the array could also provide total pressure as a second control value. After several players tested the system, we decided that three sensors would be more appropriate than two. A four sensor array was never tested.

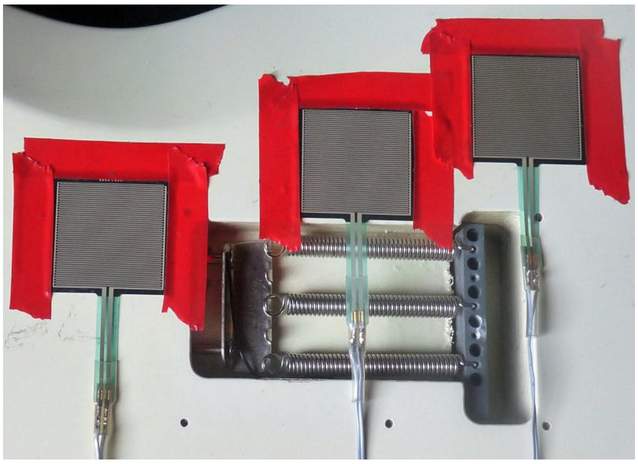

Figure 2: Rear of the guitar with the three pressure sensors (backplate removal was optional)

By having several guitarists test the system, we made two important discoveries: the system was extremely sensitive to the body shape of the player, and the playing position of some players often caused one of the pressure sensors to be favored disproportionately. Our solution was to put a block of foam along the array between the guitar body and the player's body. This solution was excellent from a technical standpoint, but something more durable would need to be used for a longterm solution.

The data which are extracted from the array of sensors and which can be used as control values are: the total pressure, the position of the center of the pressure (weighted average) and the velocity with which the movement is executed. After trying several different effects parameters, we decided to map the center of pressure and the total pressure respectively to the frequency and the Q parameter of a bandpass filter. A velocity trigger was then mapped to a gate to a delay line which adds feedback to the signal. These mappings provide an effective demonstration of the capabilities of the pressure sensor array as a sound/effects controller.

2.3 Audio Synthesis

2.3.1 *Modal Synthesis*

The purpose of this tool is to modify the output sound by modifying physical parameters of the guitar like its size and its material. For this purpose, we bring a physics-based approach, so as to have parameters that can easily be controlled, e.g, by the sensors. We chose modal analysis which consists of modeling the resonator, here the guitar body, with its vibration modes. The advantage of the approach, in contrast to more signal-based approach using a simple filter, is that it preserves sound variety when hitting the surface at different locations. In a pre-processing phase, the modal parameters of the guitar, body without the strings, are computed for each point on the surface. We chose the method described in [7] due to its robustness and multiscale structure. It uses the SOFA Framework[8] to get the mass and stiffness matrices based on the geometry (from a mesh `.obj` file), the material and the size of the object.

The resulting sounds are synthesized setting an excitation force. Modal sounds can also be creating using outputs of sensors on the fly, giving the user extended flexibility for interactive performance. We used a collection of reson filter (similar to [11]) for more expressiveness in the sound rendering. We implemented a flext object based on the source code for modal synthesis of bell sounds [11]. It gives one the opportunity to switch between different predefined models of guitar shape (i.e different precomputed set of modal parameters). A large variety of the sound performance can be obtain when changing the shape of the resonator. Geometries can be chosen from available 3D model libraries such as the Princeton 3D model search engine[9] or created with modeler such a Blender[10].

[8] http://www.sofa-framework.org

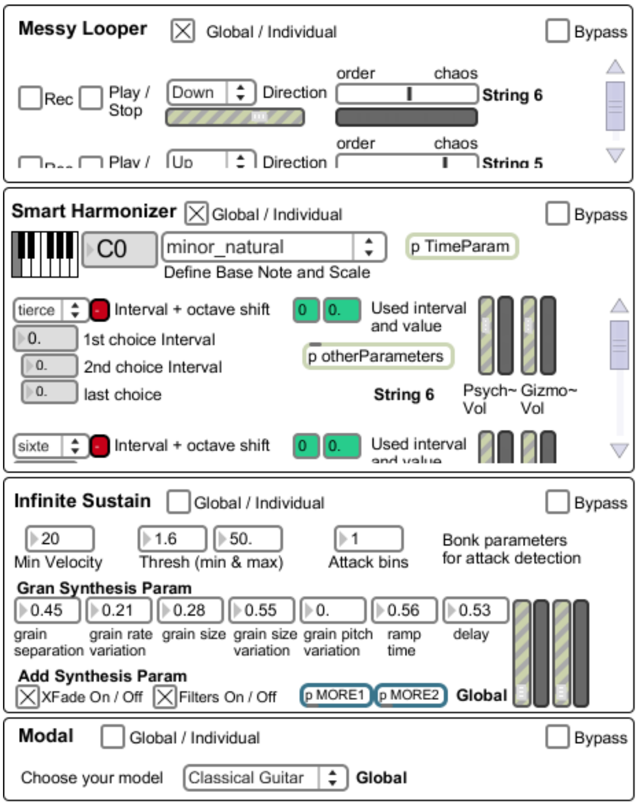

Figure 3: Multimodal Guitar Audio Synthesis Modules: the looper and harmonizer are set to a hexaphonic behavior as the sustain and modal synthesis are set to a monophonic behaviour

2.3.2 *Infinite Sustain*

The guitar is an instrument with a relatively short sustain (e.g compared to wind instruments). The electric guitar has addressed this problem with various methods: overdrive, compression and feedback. In this application, we use additive and granular synthesis to create a continous sound from a detected note or chord.

The Infinite Sustain tool goes through several steps :

- Attack detection with the `bonk~` object
- Spectral composition analysis of the detected note at 2 points (attack time + 100ms and 120ms)
- A mixture of additive (`add_synthw~`[11]) and granular (with `munger~`[1]) synthesis methods to generate the sustained note

[9] http://shape.cs.princeton.edu

[10] http://www.blender.org

[11] http://www.numediart.org/download

- A tilt sensor to control the sustain amplitude

The mix of the two synthesis techniques and the two different analysis time (100 ms and 120 ms) allow to create a lively sustained tone, with lots of timbral variation abilities. The mix between the synthesis techniques is made by tuning the volume of each one of them.

2.3.3 *Rearranging Looper*

Loop pedals are often used in a performance context to create sound textures and grooves. One can be frustrated with the static quality of the looped audio; the same loop is played over and over again, leading to boredom and to aesthetic similarity in mixed music performances. We wanted to create a looper which could rearrange the recorded audio. The program works like a beat slicer: once the recording is on, the incoming audio is analysed looking for attacks. An "event map" is created according to the attack times. The events may then be played back in any order. In this first version of the tool, the playback options are straight, backwards, and a specific random factor. With randomness set to 0, the playback stays true to the recorded audio, on the other hand with randomness set to 100, the audio events are played back totally randomly.

Setting this randomness to 100, creates a highly inspiring sonic mess in an improvisation context. This tool is in its first version and will evolve towards more interesting playback behavior, controlled by the player via sensors and playing.

2.3.4 *Smart Harmonizer*

As previously mentioned in the section 2.1.2, the hexaphonic pickup enables one to have, an individual control on each strings of the guitar. Every usual guitar effects (e.g overdrive, reverb, etc.), and in our case harmonizer, can then be tuned specifically to each string. What hexaphony brings is the possibility to harmonize properly and easily several notes at the same time, e.g, a chord, as we use one harmonizer per string.

A "smart" quality was included in our tool; by defining the root note and the scale the player is going to play in, the harmonizer can produce notes which only belong to that scale. We have implemented some common scales like major, minor natural, minor harmonic, minor melodic ascending and descending. The set of intervals defined for each scale is totally tunable.

In the Max/MSP version of the harmonizer, the user has the possibility to choose between the IRCAM's `psych~` and the `gizmo~` objects, to perform the pitch transposition. We kept both objects, even if the first one is not free, as they do sound quite differently (the first one is a time domain algorithm and the latter a frequency domain algorithm).

3. CONCLUSION AND PERSPECTIVES

We worked on all the parts of the signal chain of an augmented instrument, from extraction and audio signal features to digital audio effects manipulated by gestural control with sensors. We achieved a usable toolbox for hexaphonic and monophonic guitar. Most of these tools (i.e, Fretboard Visualization, Polyphonic Pitch Extraction, Fretboard Grouping, Rear-Mounted Pressure Sensors, Modal Synthesis, Infinite Sustain, Rearranging Looper, Smart Harmonizer) are available for both the Max/MSP and PureData (extended) environments. Efforts will be put on the availability of the entire toolbox on both platforms. Documentation and tutorials will be provided so that the complete package, freely downloadable on the eNTERFACE'09 and numediart websites, will be directly usable. We are looking forward to user feedback.

Physical sound synthesis sometimes lacks realism. One interesting approach can be to use pre-recorded sounds relevant to specific playing techniques on guitar such as sliding, slapping, etc, in order to add more texture to the sounds. Since we proposed granular synthesis to enrich the guitar sustain, we could collect specific audio grains for enhancement of modal sounds and sustains, separately. By using the approach from Picard et al. [8], audio grains could be automatically extracted from recordings. In addition, the audio grains could be mapped to specific sensors outputs during runtime.

4. ACKNOWLEDGMENTS

Christian Frisson, John Anderson Mills III, Loïc Reboursière and Todor Todoroff are supported by numediart[12], a long-term research program centered on Digital Media Arts, funded by Région Wallonne, Belgium (grant N°716631).

Otso Lähdeoja's work is partially funded by Anne Sedes from CICM, University of Paris 8.

We would like to thank all the organizers from the eNTERFACE'09 Summer Workshop on Multimodal Interfaces, hosted at InfoMus lab, Casa Paganini, Genova, Italy, from July 13th to August 8th, where this project was initiated[13].

5. REFERENCES

[1] I. I. Bukvic, J.-S. Kim, D. Trueman, and T. Grill. munger1˜ : towards a cross-platform swiss-army knife of real-time granular synthesis. In *Proc. ICMC*, 2007. http://ico.bukvic.net/Max/.

[2] A.-M. Burns. Computer vision methods for guitarist left-hand fingering recognition. Master's thesis, McGill University, 2006.

[3] R. M. French. *Engineering the Guitar: Theory and Practice*. Springer, 2008.

[4] O. Lähdeoja. An approach to instrument augmentation : the electric guitar. In *Proc. of the 2008 Conf. on New Interfaces for Musical Expression (NIME08)*, 2008.

[5] N. Leroy, E. Fléty, and F. Bevilacqua. Reflective optical pickup for violin. In *Proc. of the 2006 Intl. Conf. on New Interfaces for Musical Expression (NIME06)*, 2006.

[6] A. Loscos and T. Aussenac. The wahwactor: A voice controlled wah-wah pedal. In *Proc. of NIME05*, 2005.

[7] C. Picard, F. Faure, G. Drettakis, and P. G. Kry. A robust and multi-scale modal analysis for sound synthesis. In *Proc. of the Intl. Conf. on Digital Audio Effects*, 2009.

[8] C. Picard, N. Tsingos, and F. Faure. Retargetting example sounds to interactive physics-driven animations. In *AES 35th Intl. Conf. on Audio for Games*, 2009.

[9] C. Poepel. Synthesized strings for string players. In *Proc. of the 2004 Conf. on New Interfaces for Musical Expression (NIME04)*, 2004.

[10] M. Puckette. Patch for guitar. In *Pd-convention*, 2007. Pd patches available here http://crca.ucsd.edu/~msp/smeck/latest/.

[11] K. van den Doel, P. Kry, and D. K. Pai. Foleyautomatic: Physically-based sound effects for interactive simulation and animations. In *ACM SIGGRAPH 01 Conf. Proc.*, 2001.

[12] http://www.numediart.org

[13] http://www.infomus.org/enterface09/