Actes des Journées d'étude sur les Identités de la Guitare Electrique, Maison des Sciences de l'Homme Paris Nord, 18-19 Mai 2009

MULTIMODAL GUITAR: PERFORMANCE TOOLBOX AND STUDY WORKBENCH

Christian Frisson [1,♮], *Loïc Reboursière* [2,♭], *Wen-Yang Chu* [1,♯], *Otso Lähdeoja* [3],
John Anderson Mills III [2], *Cécile Picard* [4], *Ao Shen* [5], *Todor Todoroff* [2]

[1] TELE Lab, Université Catholique de Louvain (UCL), Belgique; [2] TCTS Lab, Université de Mons (FPMs), Belgique;
[1,2] numediart Research Program on Digital Art Technologies;
[3] CICM, University of Paris 8 (Fr); [4] REVES Lab, INRIA Sophia-Antipolis (Fr); [5] EECE Dept., University of Birmingham (UK)
[♮] Project coordinator; [♭] Performance Toolbox coordinator; [♯] Study Workbench coordinator

ABSTRACT

This project aims at studying how recent interactive and interaction technologies would help extend how we play the guitar, thus defining the "*multimodal guitar*". We investigate two axes, 1) "A gestural/polyphonic sensing/processing toolbox to augment guitar performances", and 2) "An interactive guitar score following environment for adaptive learning". These approaches share quite similar technological challenges (sensing, analysis, processing, synthesis and interaction methods) and dissemination intentions (community-based, low-cost, open-source whenever possible), while leading to different applications (respectively artistic and educational), still targeted towards experienced players and beginners.

We designed and developed a toolbox for multimodal guitar performances containing the following tools : Polyphonic Pitch Estimation (see section 3.1.1), Fretboard Grouping (see section 3.1.2), Rear-mounted Pressure Sensors (see section 3.2), Infinite Sustain (see section 3.3.2), Rearranging Looper (see section 3.3.3), Smart Harmonizer (see section 3.3.4). The Modal Synthesis tool (see section 3.3.1) needs be refined before being released.

We designed a low-cost offline system for guitar score following (see section 4.1). An audio modality, polyphonic pitch estimation from a monophonic audio signal, is the main source of the information (see section 4.2), while the visual input modality, finger and headstock tracking using computer vision techniques on two webcams, provides the complementary information (see section 4.3). We built a stable data acquisition approach towards low information loss (see section 4.5). We built a probability-based fusion scheme so as to handle missing data ; and unexpected or misinterpreted results from single modalities so as to have better multi-pitch transcription results (see section 4.4). We designed a visual output modality so as to visualize simultaneously the guitar score and feedback from the score following evaluation (see section 4.6). The audio modality and parts of the visual input modality are already designed to run in realtime, we need to improve the multimodal fusion and visualization so that the whole system can run in realtime.

KEYWORDS

Mono- and polyphonic multi-pitch transcription, audio synthesis, digital audio effects, multimodal interaction and gestural sensing, finger tracking, particle filtering, multimodal fusion, guitar score following

1. INTRODUCTION

1.1. The guitar, a marker of telecommunications technologies

The evolution of the guitar as a musical instrument has been showcasing several milestones in telecommunication engineering technologies discovered in the last centuries, from vacuum tube amplification, effect pedals with built-in electronic diodes and chips, magnetic/piezoelectric/optical sensing, wireless linking, and so on [20]. The "guitar synthesizer", extending 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 magnetic [62] or optical [55, 46] or piezoelectric sensing) and an analog or digital processing device (the latter embedding a microprocessor). However, these processing devices still don't offer today an ergonomic, customizable and freely extendable user interface, similar to the one featured on modular environments for audiovisual analysis and synthesis such as PureData (pd-extended [57]) and EyesWeb [28], that can be run on most laptops.

Additionally, guitarists have been developing a body language vocabulary adding a visual communication to their musical performance [8] (from full-body gestures to facial expressions). The multiple sensing methods available at a low cost nowadays, from remote cameras [5, 6] to built-in accelerometers among other sensors [39], would allow to better understand the gestural intention of the guitarist and emphasize hers/his musical expression, renaming the instrument an "augmented guitar" [40, 41].

1.2. From guitar learning towards educational games

Plenty of methods have been proposed to learn and teach guitar playing : from the academic study and interpretation of classical scores, to the more popular dissemination of guitar tabs providing a simplified notation, or teaching by playing within nomadic communities, playing by ear in an improvised music context, and so on... However, most of these methods require a long training before the user is proficient enough to become autonomous.

After efficiency, usability, ergonomics ; an important factor when designing today's user interfaces is pleasurability. "Guitar Hero" [37], a video game featuring a controller inspired by a "diminished" [2] version of the guitar, consisting in having players challenge each-others in following a "4 on/off" note version of a guitar score, has recently caught a lot attention. Blend pleasure and learnability in educational games [21, 7].

2. TWO SUBPROJECTS, TWO APPLICATIONS

We proposed two sub-projects blended together in this project. While the first is aimed at artists and the second at "standard users", while the first would help create artistic performances and the second would enrich educational learning ; we believed that these two projects share enough similarities among their work packages and deliverables so as to merge them together in a single eNTERFACE'09 project.

2.1. A gestural/polyphonic sensing/processing toolbox to augment guitar performances

The purpose of this subproject is to study and refine the methods employed in the sound analysis, synthesis and processing of the guitar and in the gestural expression of the guitarist, so as to provide a low-cost and opensource, software and hardware, toolbox that can allow guitarists to personalize their performance settings, from beginners to experts.

2.2. An interactive guitar score following environment for adaptive learning

The purpose of this project is to propose an open-source, community-based, standalone application to guitar players, from beginners to experts, for helping them to master musical pieces, with different adaptive layers of difficulty. The application would be used with real guitar to perform and follow a musical piece displayed on a screen, while a real-time polyphonic transcription of the guitarist's playing would allow the evaluation of its performance in an enjoyable way.

3. ARTISTIC SUB-PROJECT : PERFORMANCE TOOLBOX

For this sub-project we decided to work on every part of the chain of the use of an augmented guitar :

- 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 the computer produces
- Audio synthesis : how can we enhance guitar performance with hexaphonic effects or not

As these categories are quite general, we decided to focus on the tools listed below :

- Polyphonic Pitch Estimation (see section 3.1.1)
- Fretboard Grouping (see section 3.1.2)
- Rear-mounted Pressure Sensors (see section 3.2)
- Modal Synthesis (see section 3.3.1)
- Infinite Sustain (see section 3.3.2)
- Rearranging Looper (see section 3.3.3)
- Smart Harmonizer (see section 3.3.4)

To build and test these tools we used a Fender Stratocaster guitar with a Roland GK3 pickup [65] mounted on in and a String Port interface made by Keith McMillen [49]. Concerning the pressure sensors, as well as the wireless sensors interface, we used the ones from Interface-Z [30]. All the tools have been developped for Max/MSP [15] and/or PureData [57] environments.

3.1. Audio Analysis

3.1.1. Polyphonic Pitch Estimation

As mentioned in the introduction, there are several ways to capture the vibrations of the six strings of the guitar separately. Several sources point out that the crosstalk between strings is smaller using piezoelectric pickups than with electromagnetic ones, because the latter sees the magnetic flux in the coil beneath one string being influenced by the movement of adjacent strings. But we only had the possibility to test a GK3 hexaphonic magnetic pickup from Roland [65].

The YIN method by de Cheveigné and Kawahara [10, 9] is recognized as one of the most accurate pitch estimator when there is no background noise. While similar to autocorrelation, YIN uses a difference function that minimizes the difference between the waveform and its delayed duplicate instead of maximizing the product. Changes in amplitudes between successive periods will both yield an increase in the difference function. This compares to the autocorrelation function, known to be quite sensitive to amplitude changes, where an increase in amplitude tends to cause the algorithm to chose a higher-order peak and hence a too low frequency estimate ; while a decrease in amplitude has the opposite effect. The introduction of the cumulative mean normalized difference function removes the need for an upper frequency limit and the normalization allows the use of an absolute threshold, generally fixed at 0.1. This threshold can also be interpreted as the proportion of aperiodic power tolerated within a *periodic* signal. A parabolic interpolation step further reduces fine errors at all F_0 and avoids gross errors at high F_0. We used the `yin~` external Max/MSP object developed at IRCAM.

In a paper dedicated to guitar patches [63] and also using the Roland hexaphonic microphone [65], Puckette, who wrote both `fiddle~` and `sigmund~` pitch extractors, both widely used within Max/MSP and PureData, suggests using the later. We found it was interesting to combine an time- with a frequency-domain pitch estimator. `yin~` gave overall better results, but we kept the possibility to combine it with `sigmund~` as they yielded close results during stable sections.

The pitch detection algorithm is mainly based on `yin~`. Post-processing, based on the quality factor of `yin~` frequency extraction and on the level of each guitar string signal, allows to remove the spurious notes present mainly during transitions and when the volume fades out because of crosstalk between the strings. Though not critical in most conditions, the following parameters allow the user to fine-tune the pitch detection to his needs :

- Quality threshold : we didn't let a pitch estimate be taken into consideration while the average of the last quality factors was below a user-defined threshold.
- Level threshold : as the coupling between strings is shown to be higher from higher pitch to lower pitch strings than in the opposite direction [52], we defined a level threshold deviation per string, applied cumulatively from string to string.
- Outside range : an allowed range is attributed to each string, depending on the tuning and on an amount of semi-tones per string. This allows to exclude notes that may be detected outside that range because of coupling or crosstalk.
- Mean-Median divergence, expressed in half tones, and median window size : the difference between the mean and the median of the pitch estimates over a user-defined window length is a good measure of pitch stability.
- YIN-Sigmund divergence, expressed in half tones, allows to

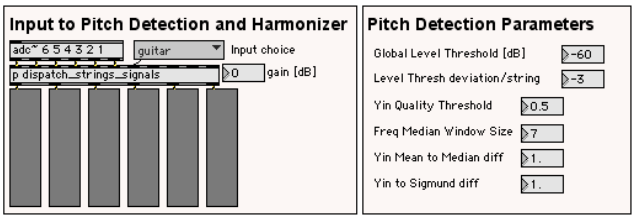

FIGURE 1: *The view meters showing signals coming from the hexaphonic microphone on the left and the user parameters of the pitch estimation on the right*

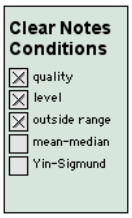

FIGURE 2: *There are five conditions that help the user tailor how notes will be cleared depending on his application.*

limit note detection when an even higher stability is achieved, at the cost of a bigger latency. But it might be desirable in some situations.

One issue is to detect a note being played, but another one potentially as important is to detect the end of a note. There is no consensus about when a note is ended except the start of a new note. Several factors make it indeed a difficult task :

- Guitar strings, mainly the lower pitch ones have a very long decay. The pitch estimation algorithm can keep detecting a pitch with a good quality factor even when the sound of a string becomes masked by the other strings for the human ear.
- String coupling inherent to the instrument design [52] can have the effect of keeping strings vibrating, mainly open strings, as they keep receiving energy from the vibration of other strings.
- Crosstalk between the amplified string signals induced by the hexaphonic microphone, picking up the sound of adjacent strings, might give a reading even when the string doesn't vibrate.
- Depending on the application, one may want to keep the note living on one string until another note is played ; for instance, to provide a synthetic or re-synthesized sustain.
- Fingers on adjacent strings, on the touch or near the bridge can inadvertently damp or re-trigger a note.

For all those reasons it was obvious that we couldn't provide a Fit them All method. Instead, we provide a comprehensive set of parameters that will define when a note is supposed to be considered ended. Conditions that will clear the notes are depicted in Fig. 2. Most of those conditions depend on parameters already tuned for the pitch detection :

- Quality : clears the note if the `yin~` quality falls below the threshold.
- Level : clears the note if the input level is below the level threshold associated with the deviation per string as explained above.
- Outside range : clears the note if its frequency evolves outside the range attributed to each string, something that might happen if the microphone crosstalk bleeds notes from adjacent strings.
- Mean-Median divergence : when a note dies or changes, its frequency changes and the difference between the mean and the median of the last values increase.
- YIN-Sigmund divergence : in the same way, only very stable frequencies yield similar results when comparing the `yin~` and `sigmund~` outputs. Putting a threshold in half-tones between both outputs allows to clear notes when instability occurs.

Those conditions have an effect both on the OSC note output sent to the *Fretboard Grouping* patch, on the synthesizer (currently a simple sawtooth oscillator followed by a resonant filter used only as a prove of concept) and on the harmonizer algorithm if *Mute harmonizer if cleared notes* is selected.

3.1.2. Fretboard Grouping

Hexaphonic pickups appeared in the 80's and were first aimed at making the guitar a synthesizer : some might want to put a bass sound on the 6th and 5th strings, an organ sound on the next two ones and then keeping the guitar sound for the last two strings. This is one example of what could be done with an hexaphonic pickup and an hexaphonic-to-MIDI hardware converter. Plugin that kind of pickup in software applications such as Max/MSP [15] and PureData [57] can expand the guitar sound effects string by string but allows more particularly to go even further in the segmentation of the fretboard. As an hexaphonic pickup directly provides a string-per-string control of the guitar, coupling it with a polyphonic pitch estimation as described below enables you to use the fret as the smallest element of your segmentation and to go deeper in the management of your fretboard. One can think in a "classical way" in terms of chords and scales or, in a more plastic way, in terms of fretboard zones and shapes (chord or geometrical).

The tool we made gives one the possibility to create groups of notes on the fretboard and then checks whether the played note(s) belong(s) to one of the defined groups or not. Using the example of the synthesizer guitar again, one can not only decide of the sound or effect to apply on only one string but on all the note of a defined group. Zones of effects can then be defined on all the guitar fretboard. We added a graphical visualization to this tool so as to have a graphical feedback for the created groups or for the played notes.

Another feature that our object handles is the possibility to detect if a group has been entirely played, meaning if all notes have been played without one extra note (not present in the group) in between. Expanding a bit this entire played group property, we can then recognize whether the group has been played as a chord, as an arpeggio or as a simple group. A more detailed description of this feature is done afterwards. A specific chord or a specific note can then, for instance, be used to trigger sounds or change mappings, etc...

A first version of the tool has been made as Max/MSP and PureData patches. The final version has been written in Java in order to enhance the speed of the recognition and to make it easier to use as everything is gathered in one object to which messages are sent. The choice of Java as the language to write the Fretboard Grouping external was motivated by the development by Pascal Gauthier of pdj [23], a java external plugin for PureData. This plugin is based on the `mxj` Max/MSP object implementation and permits first to write external for PureData in Java (which was not the case before) and second, to use the same code to generate both Max/MSP and PureData external which was not the case when developing in C or C++.

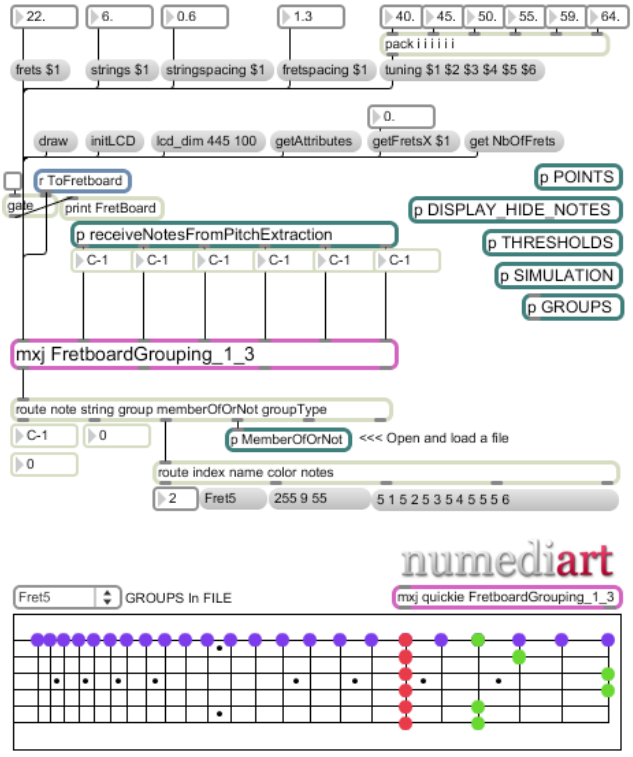

FIGURE 3: Max/MSP help patch of the Fretboard Grouping tool

3.1.2.1. *Graphical Visualization*

We wanted to visualize the guitar fretboard considering its physical parameters (number of strings, number of frets and tuning), the groups the user is working with and the played notes detecting 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 the note to be displayed on the right scale.
- the display parameters : strings spacing and fret spacing factor. These factors can easily lead to a non realistic representation of the fretboard, but they can definitely be helpful to clearly see notes especially in the highest frets.

Other guitar fretboard visualization already exist. We can cite for example the Frets On Fire [38] game (free version of the Guitar Hero game) or the TuxGuitar [68] tablature editor and score viewer software. The second was the most suited representation as the first one gives a non realistic display of the fretboard.

As we wanted an easier integration with the Fretboard Grouping tool and as we first needed a quick implementation, we decided to developed the visualization part for the `lcd` object of Max/MSP environment. Only the visualization for the `lcd` object has been developed for the moment. Further investigations will be led on the OpenGL visualization tool that can be used in both software (GEM library for PureData [16] and `jit.gl.sketch` object for Max/MSP).

3.1.2.2. *Groups*

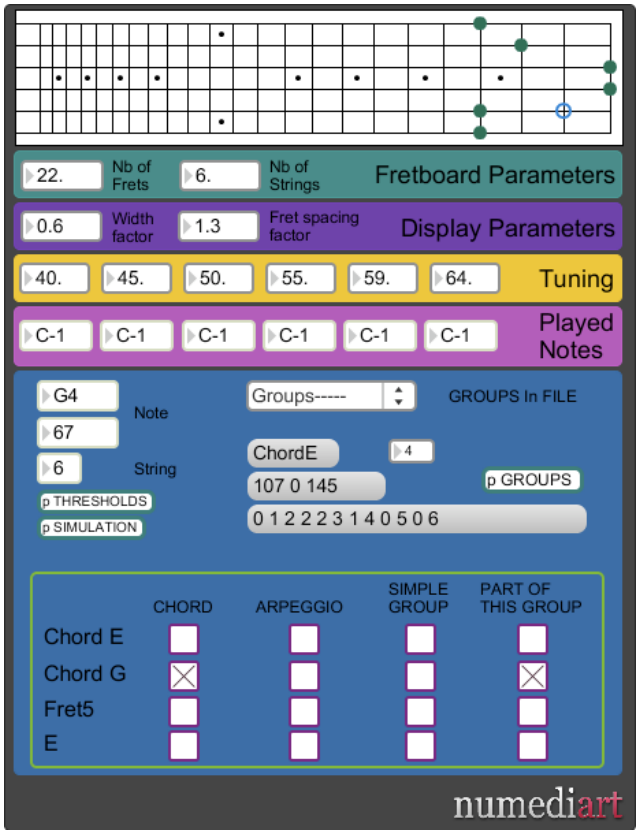

FIGURE 4: Detection of the G chord

The construction of the groups one wants to work with follows these steps :

- played or selected (directly on the display of the fretboard) notes are added to the group
- when all the notes of the group have been selected, the group needs to be saved
- when all the groups are defined, the file, containing the groups definition, needs to be saved too
- read the file to use the groups

A display feature is accessible to give one the possibility to have a visual feedback of which notes are in the group.

Two parameters define a group : completeness and time. Therefor three types of group are defined :

– chords : all notes of a group are played (without extra-group notes) simultanesouly
– arpeggio : all notes of a group are played (without extra-group notes) not simultaneously but under a certain threshold
– simple group : all notes of a group are played (without extra-group notes) above a certain threshold

The discrimination between the different types of groups is, for the moment, based on the time factor. Two thresholds are then defined : one, discriminating between chord and arpeggio and the other one discriminating between arpeggio and simple group. If all notes of one group are played under this chord / arpeggio discrimination threshold the played group is recognized as a chord, if they are played above this threshold and under the arpeggio / simple group threshold it will be recognized as an arpeggio, and if it is below this last threshold, it will be recognized as a simple group. If notes are played interlaced with other notes (of other groups or not in any groups), notes are just recognized as part of the groups.

3.2. Gestural Control : Rear-Mounted Pressure Sensors

The basic premise of this gestural control subproject was to add pressure sensors to the back of an electric guitar in order to add another expressive control to the sound for the guitarist. We decided that one goal of this control should be that it should not require the player to spend much time learning the controller, but should feel like a natural extension of playing the guitar.

3.2.1. Sensors

Due to familiarity, we chose FSR pressure sensors to use for this implementation. They are well-known as reliable and easy to use. We made the decision to use their 4 cm square pressure sensors because we felt that they would cover more area of the back of the guitar. After some testing, it was determined that the 4 cm square 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 to be tested further in the future.

The sensors came with a prebuilt adjustment and op-amp circuit as well as an interface connector so that they could be used with an Interface-Z MIDI controller interface [30].

Due to familiarity, ease, reliability, and interoperability with the FSR pressure sensors, the Interface-Z was chosen as the interface between the physical sensors and the computer using a MIDI controller interface. The Interface-Z allows for up to 8 sensors which vary a control voltage between 0 and 5 volts to be translated into either a 7-bit or 10-bit MIDI controller value. Although the 10-bit controller value is split across two MIDI controllerrs. 7-bit resolution was chosen because it was sufficient for this project.

Pressure-voltage curves are supplied by the manufacturer to describe how the output voltage varies with supplied pressure. It was experimentally determined that for the pressure levels seen at the back of the guitar, the pressure sensors were operating nonlinearly. A $y = x^3$ mapping was implemented to linearize the output of the sensors.

An array of pressure sensors were chosen as the interface to use instead of a position sensor, because the array of pressure sensors could also provide total pressure as another control value It was unknown how many sensors would need to be included in the array to provide enough feedback for the type of control that we wanted. Initially tests were run using only two sensors, but after several players tested the system, we decided that three sensors would be more appropriate. A four sensor array was never tested.

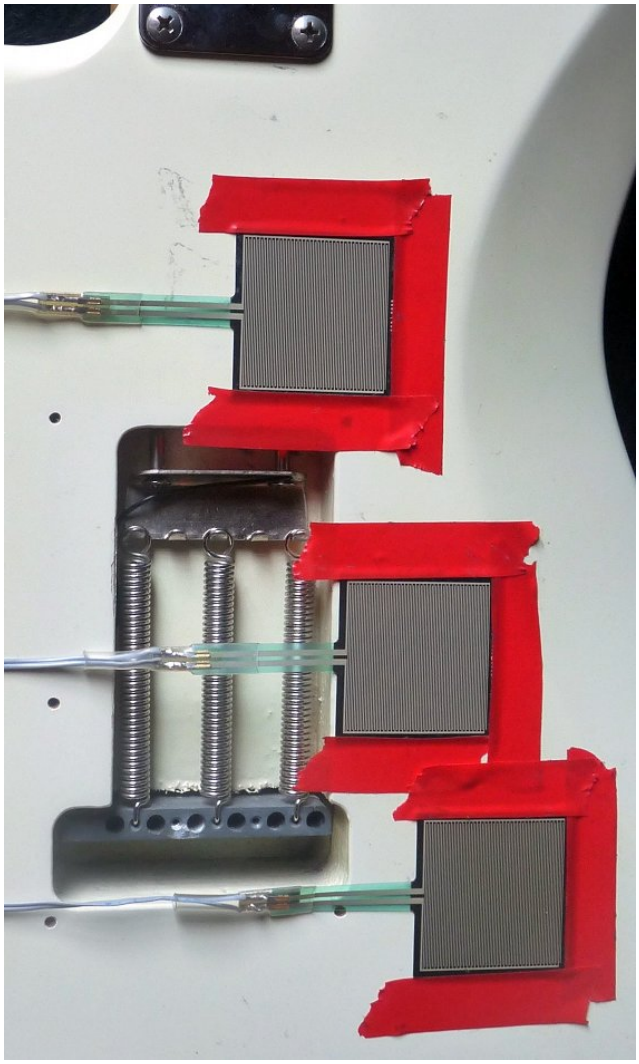

FIGURE 5: Rear of the guitar with the three pressure sensors

By having several different guitarists test the three array system, we made two important discoveries : the system was extremely sensitive to the bodyshape of the player, and the playing position of some players often caused one of the pressure sensors to be favored inappropriately. The solution which we found for this 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.

3.2.2. Mapping of Sensor Data to Control Values

In order to acquire the total pressure, p_{tot} from the array, the following equation was used.

$$p_{tot} = \frac{\sum_{i=1}^{n} p_i}{n} \quad (1)$$

where n is the number of sensors and p_i is the linearized control value representing pressure on an individual sensor. The center of pressure, P_c, along the array is given by a weighted average of particles as follows.

$$P_c = \frac{\sum_{i=1}^{n} p_i P_i}{p_{tot}} \quad (2)$$

where P_i is the position of an individual sensor. This equation is useful because the positions of the sensors do not need to be evenly spaced.

The velocity of P_c, v_c is derived using the following equation.

$$v_c = \frac{P_c(t + \triangle t)}{\triangle t} \quad (3)$$

where $\triangle t$ is 20 msec.

Control of effects and other parameters was tested using both this velocity and a derived acceleration. We decided that in this particular configuration using these higher order controls to directly control effects parameters was generally uncomfortable and unnatural, but using a velocity threshhold as a trigger for effects did feel natural. To allow for a natural velocity trigger, the threshhold values are different in the positive (or right-turning) and negative (or left-turning) directions.

After trying several different effects parameters, the choice made for the proof of concept video was the following.

- P_c is mapped to the center frequency of a bandpass filter. The range of P_c is mapped logarithmically between 600 Hz and 4000 Hz.
- p_{tot} is mapped to the Q of the same bandpass filter. The range of p_{tot} is mapped linearly to a Q value between 10 and 30.
- The velocity trigger is mapped to a gate to a delay line. The gate opens for 300 msec and the delay is set to 300 msec with a feedback factor of 0.8. The trigger values are -7.8 and 8.8 normalized pressure units per second.

These mappings provide an effective demonstration of the capabilities of the pressure sensor array as a sound/effects controller.

3.3. Audio Synthesis

3.3.1. Modal Synthesis

3.3.1.1. Background

Acoustics guitars, naturally producing a certain level of sound thanks to their resonating hollow bodies, faced feedback issues once amplifiers were introduced. Solid-body guitars, first designed to solve these issues, are perceived by some players to afford less playing nuances due to the reduced mechanical feedback of their less resonating body, yet offering a more widespread palette of sounds. Christophe Leduc designed an hybrid solution : the *U-Guitar* [44, 45], a solid-body guitar with resting or floating soundboard. Guitars with on-board DSP such as the Line6 Variax allows the player to virtually choose several models of plucked stringed instruments on one single real instrument. Amit Zoran's Chameleon Guitar and reAcoustic eGuitar [73] allow the guitarist to redesign the instrument virtually (by digital methods) and structurally (by mechanical methods).

There has been much work in computer music exploring methods for generating sound based on physical simulation [31, 14]. With Modalys, also usable in Max/MSP, Iovino et al. propose to model an instrument by connecting elements and calculating the modal parameters of the resulted assembly [31], yet it is tedious to build complex models for any given geometry.

Penttinen et al. [58, 59] proposed a method for real-time guitar body modulation and morphing, which has been recently implemented in Matlab and C/C++ [70]. However, their method is based on digital signal processing, essentially for computational issues, and in particular maximal efficiency.

3.3.1.2. Theory

We propose to bring a physics-based approach to the multimodal guitar, so as to give control parameters that can be easily mapped to parameters from the sensors. For this purpose, we chose the modal analysis which consists of modeling the resonator, here the guitar body, with its vibration modes. The microscopic deformations that lead to sound are expressed as linear combinations of normal modes. Modal parameters, i.e., frequencies, dampings, and corresponding gains are extracted by solving the eigenproblem with the use of a finite element method (see, for example, [53] for more details). The sound resulting from an impact on a specific location on the surface is then calculated as a sum of *n* damped oscillators :

$$s(t) = \sum_{n}^{1} a_i sin(w_i t) e^{-d_i t} \quad (4)$$

where w_i, d_i, and a_i are respectively the frequency, the decay rate and the gain of the mode *i*. The method preserves the sound variety when hitting the surface at different locations.

In the multimodal guitar, the sounding guitar (the main body without the strings) is modeled through a modal approach. In a pre-processing, the modal parameters are extracted for each point on the surface. We chose the method described in [61] due to its robustness and multi-scale structure. It uses the SOFA Framework [29] to get the mass and stiffness matrices . Specific material properties and resizement can be set for the sounding guitar. Modal synthesis is especially well suited to model irrealistic objects.

During real-time, the resulted sounds are calculated through a reson filter (similar to [17]). Modal sounds can also be convolved with outputs of sensors on the fly, giving the user extended flexibility for interactive performance.

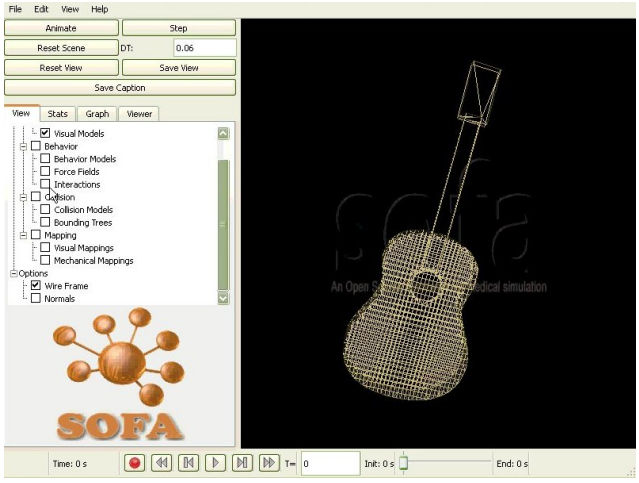

FIGURE 6: Screenshot of an acoustic guitar modelled in the SOFA Framework

3.3.1.3. Interacting with modal sounds

Using the C/C++ code for modal synthesis of bell sounds from van den Doel [18], we implemented a flext object [26, 27], thus compliant with the PureData and Max/MSP environments.The purpose of this tool is to provide more experience with modal sounds. In this manner, interesting sounds can be easily obtained by convolving modal sounds with user-defined excitations.

3.3.2. Infinite Sustain

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

The Infinite Sustain tool goes through these steps :

- Attack ("note on") detection with Max/MSP `bonk~` object
- Spectral composition analysis of the detected note at 2 points (attack time + 100ms and 120ms)
- Synthesis of the sustained note using the 2 analysed spectrums (additive synthesis with the `add_synthw~` object by Todor Todoroff)
- Synthesis of a granular sustained tone using the `munger~` object [3, 4] rewritten using flext object [26, 27]
- Mix of the two synthesis methods to create a lively sustained tone, with lots of timbral variation possibilities
- A tilt sensor controls the sustained tone playback volume

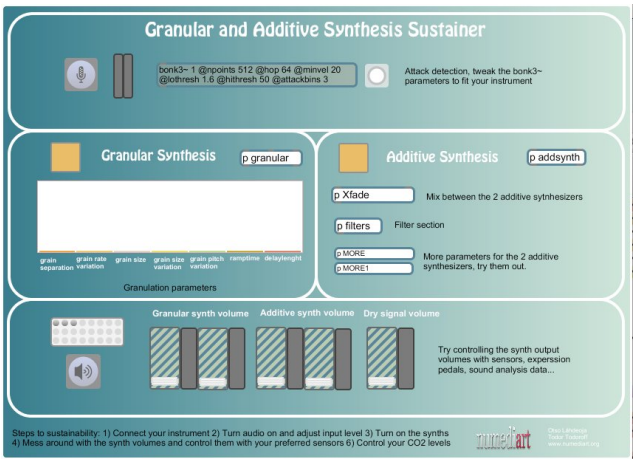

FIGURE 7: Screenshot of the Infinite Sustain tool

3.3.3. 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 : 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 ranging from 0 to 100. With random 0 the playback stays true to the recorded audio, with random 100 the audio events are played back totally randomly, creating a sonic mess highly inspiring.

The Rearranging Looper tool goes then through these steps :

- Record on and off activated by a 1-axis accelerometer on the guitar's head
- Write to a buffer, create a temporal event map with attack detection (`bonk~` object)
- Playback from the buffer
- Adjust behavior ; playback mode and randomness

The program is in its first version and will evolve towards more complex/interesting playback behavior, controlled by the player via sensors and playing.

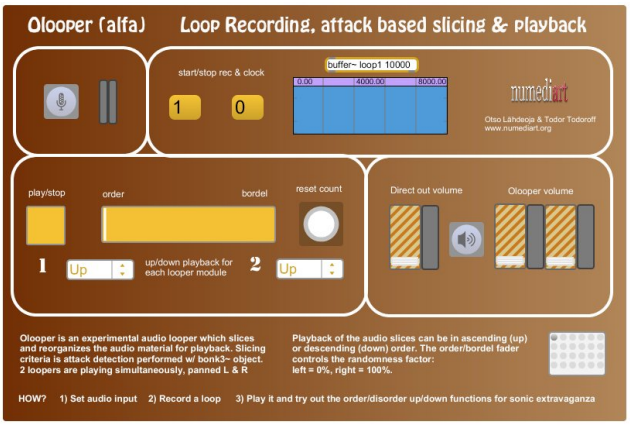

FIGURE 8: Screenshot of the Rearranging Looper tool

3.3.4. Smart Harmonizer

The ability to know the exact note being played before harmonizing, thanks to the hexaphonic pitch extraction algorithm described above (section 3.1.1), opens new interesting possibilities. In the framework of tonal music, those possibilities start with the choice of a scale, referenced to a base or root note. We have implemented some common scales like major, minor natural, minor harmonic, minor melodic ascending and descending. Currently, there is a `coll` object that loads a file containing the intervals in half tones within an octave for each defined scale :

```
major, 0 2 4 5 7 9 11;
minor_natural, 0 2 3 5 7 8 10;
minor_harmonic, 0 2 3 5 7 8 11;
minor_melodic_asc, 0 2 3 5 7 9 11;
minor_melodic_desc, 0 2 3 5 7 8 10;
```

The selected set of intervals, chosen by name with a menu, are repeated over the whole range of MIDI notes, starting from the root note defined by the user, and stored in another `coll` object containing all the notes of the chosen, now current, scale that will be used by the harmonizer. More scales can be defined by the user and it would also be quite easy to define a set of notes that don't repeat at each octave, simply by loading in the current scale `coll` a list of notes belonging to the set contained in a text file. Those

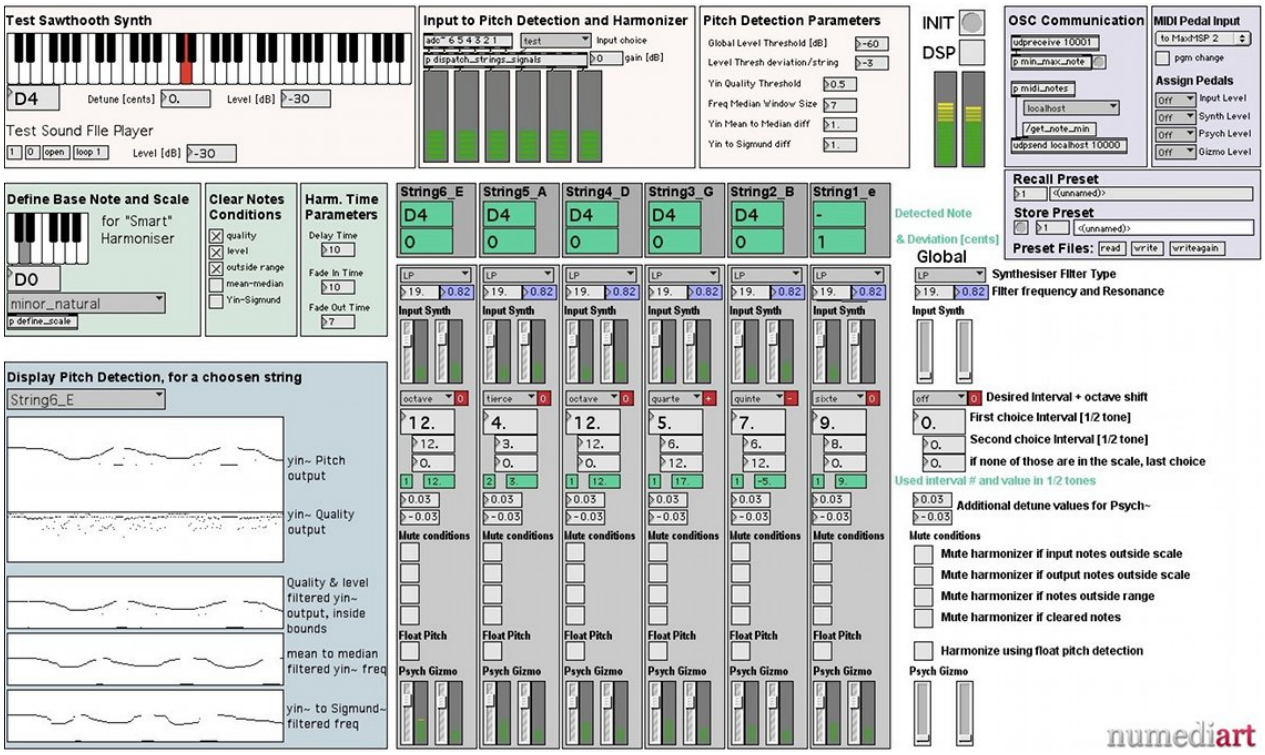

FIGURE 9: Shown here with a synthetic test tone, the Max/MSP patch that extracts the pitches independently for each string, using an hexaphonic microphone, and harmonizes the played notes in a smart way, i.e. taking into account the notes being played, a chosen scale and desired intervals defined independently for each string.

sets could be defined using a virtual keyboard, a MIDI or OSC input, defined with the *Fretboard Grouping* tool, or simply in a text editor.

In order to define how to harmonize, we also need rules to modify the desired intervals depending on whether or not the harmonized note with the desired transposition factor is part of the chosen scale. Therefore we use the following rules, also stored in a text file loaded by a `coll` object, where the first interval is the preferred one, and the second, the choice if the first fails to transpose the note within the chosen scale :

```
unison, 0 0;
second, 2 1;
third, 4 3;
fourth, 5 6;
fifth, 7 6;
sixth, 9 8;
seventh, 11 10;
octave, 12 12;
```

In order to augment the flexibility of the program, those first and second intervals can be freely entered by the user, including fractional half-tones. Though the two proposed choices in the predefined intervals will always yield at least one note in the scale for traditional western scales, it may not be the case for non-western ones or with non-integer half-tones. Therefore we added a third choice (see Fig. 9). Experiments with a viola player and quarter or eight-tones yielded very surprising and interesting results. But the availability of three levels of choice allow also to force an interval by specifying three times the same one, or to allow only one interval if it fits within the scale and no transposition otherwise, by specifying the desired interval as the first choice and 0 for the second and the third ones. We also added the possibility to shift the result one octave down or up with the red menu (-, 0, +) at the right of the interval menu. The actual harmonization is done with a combination of the `psych~` and `gizmo~` objects whose levels may be adjusted differently for each string. We kept the two algorithms, the first in the time domain and the second in frequency domain, as they do sound quite differently. And, at almost no cpu cost, one may define two additional sound transpositions with `psych~`. This can be used for chorus-like effects or for octave doubling.

Thus, using the knowledge of the played notes, a scale or a set of notes, and rules determining the desired harmonization factor depending on the formers, we defined an adaptive transformation, depending on the input notes. And different intervals or rules may be applied individually to each guitar string.

We described how the transposition factor can be chosen in a smart way, but there are times where one wants to mute the output of the harmonizer. We implemented the following mute functions :

- *Mute input notes outside the chosen scale* : if the played note is not in the scale, there will be no harmonizing.
- *Mute output notes outside the chosen scale* : if the input note transposed following the user-defined rule doesn't fit in the chosen scale, it won't be harmonized.
- *Mute output notes outside range* : a range is defined for each string, from the lowest note playable on that string (the tu-

ning can be easily defined) to that note transposed by a user-defined amount of semitones ; if the harmonized note would fall outside the range, it isn't harmonized.
- *Mute harmonizer if cleared notes* : the pitch extractor has parameters that define what conditions stop the note. the harmonization can be muted when a clear note condition is met.

We might also want to prevent some played notes to be heard on some strings. That would for instance be the case if some string/note combinations are used as a kind of program change, to change a preset that would for instance change the scale or the harmonizing choices. This hasn't been implemented yet.

Finally, it is possible to *Harmonize using float pitch detection* : the harmonized note is "retuned", that is, transposed to the closest tempered scale note, by taking into account the detected deviation from the tempered note given by the pitch extractor. Besides allowing the harmonizer result to be exactly in tune, it opens another interesting application : if the harmonizing interval is set to unison, the resulting note can create beatings in regard to the played note when the string is bent, or simply thicken the sound as no played note is exactly in tune, and as the deviation in cents from the perfect tuning evolves over the duration of a note.

4. EDUTAINMENT SUB-PROJECT : STUDY WORKBENCH

Knowing that fitting an hexaphonic pickup requires a structural modification of the guitar, will it help enhance the ergonomics (non-invasiveness) and allow a better performance in terms of signal processing ? Should we use a low-cost alternative that requires a webcam to analyse the guitarist's gestures using computer vision techniques [5, 6] and a monophonic guitar sound, so that to extract all the necessary information of both methods through multimodal fusion [64] ? In the following part, we will narrow down the scope to focus on the multimodal musical transcription.

The precursor of the multimodal musical transcription is Gillet's work transcribing drum sequences [24], where joint features are used. Ye Wang borrowed the philosophy of the audio-visual speech recognition to try to realize the violin transcription based on the weighted sum of ouput values of modalities [69]. They both show the trend and promising future of multimodal transcription. Garry Quested and Marco Paleari separately proposed different methods on the guitar with various modalities and deterministic fusion methods [64, 56], where [56] produced 89% of the recognition rate. However, their simple deterministic nature of the fusion schemes fails to handle missing samples in audio and dropping frames in video, which are always an issue in the real-time application, and unexpected output of modalities. Hence, the recorded experiment data requires no information loss, which leads to the demand of very expensive high end recording equipment and unaffordable computation load for a real-time application. Moreover, the static weights on each modality in the fusion scheme do not allow the system to adjust according to the reliability of modalities, such as change of lighting condition to cameras, and our crucial objective—user skills.

Considering their experience and working towards a real-time audio-visual system of the multimodal musical transcription, we build the system with the following characteristics :

1. Have low-cost hardware and software setup
2. Provide reliable real-time modalities
3. The audio modality is the main source of the information (see section 4.2), while the video modality provides supplementary information (see section 4.3)
4. Provide low information loss by building a stable data acquisition approach (see section 4.5)
5. Have a probability-based fusion scheme to handle missing data, and unexpected or misinterpreted results from single modalities to have better multi-pitch transcription results (see section 4.4)
6. Include the updatable weights of each modalities in the fusion scheme to allow system to adjust according to users and reliability of modalities.
7. Visualize simultaneously the guitar score and feedback from the score following evaluation (see section 4.6)

A real-time system of the multimodal musical transcription mainly consists of three parts : data acquisition, modality construction, machine learning and multimodal fusion. In the following sections, we first introduce our design and implementation of the overall system, and then detail them right after that.

4.1. System Overview

In our system, two webcams and a finger tracking algorithm form the video modality, soundcard and a polyphonic pitch estimator form the audio modality, and finally a multimodal algorithm fuse these two modality, shown in the top figure of Fig. 10. A recording is system is also necessary during all the process from design and fine-tune modalities, to multimodal fusion mechanism design, shown in the bottom figure of Fig. 10. The system hardware overview is shown in Fig. 11, where we adopts the most common and low-cost devices, including :

- 1 laptop with Microsoft Windows XP
- Two USB Logitech webcams (two concurrent high speed Sony PS3 Eye cannot currently run at the same time on the same computer)
- 1 Line6 Variax 300 guitar
- 1 TASCAM US-144 USB soundcard
- 1 external FireWire/USB hard-drive for storing offline training data (non-necessary for the final system)

We also use an ARToolKitPlus [1] marker on the guitar headstock and colored markers for fingertracking on finger nails.

The problem now urns to choose a open source/low cost platform which can provide stable, simple, and quick development on data acquisition, modality construction, and machine learning and multimodal fusion, where any of them requires tedious work and complicated analysis process, based on our low cost hardware setup. EyesWeb [28] from InfoMus Lab, DIST-University of Genova, has been a very successful platform for audio-visual real-time applications since 1997, providing an intuitive visual graphical development environment and powerful support without charge. Pure Data is also another effective open source real-time graphical programming environment for audio, video, and graphical processing [57]. However, they can not easily make lossless synchronized recording using multi-cameras and one microphone at the same time. Their lack of support of easily integrating programs in various languages slows down the early prototyping process.

OpenInterface [42, 43], developed by TELE Lab, University of Louvain, is an ideal choice in our application, which allows to fast prototype real-time multimodal interactive systems (or online system with constant delay), and implement synchronized raw data

recorder. It provides a graphical programming environment and interfacing programs in all the main-stream programming languages such as C/C++, Java, Matlab [48] and .NET. Shown in Fig. 10, all of the data acquisition, finger tracking, pitch estimation, and multimodal fusion algorithm, and recording function is implemented in OpenInterface.

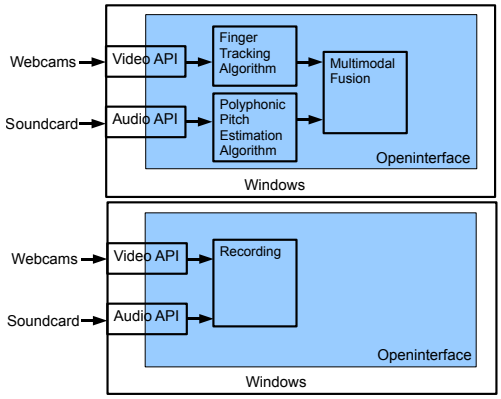

FIGURE 10: System Software Architecture Overview. Top : Runtime. Bottom : Recording

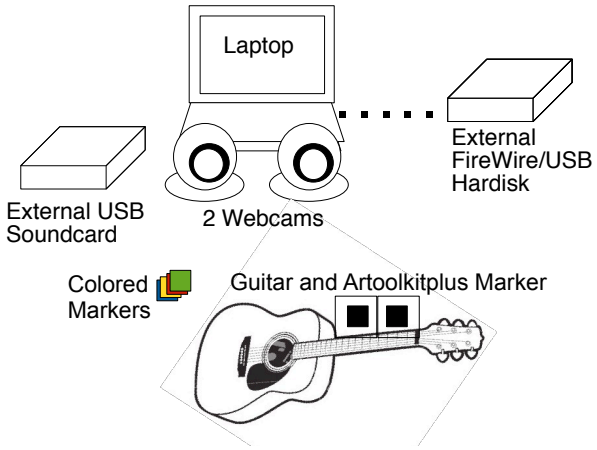

FIGURE 11: System Hardware Overview

Providing higher abstraction and statement power, Matlab allows programmers to design and implement algorithm more easily and faster. Thanks to the easy integration of the Matlab engine by OpenInterface, we can easily have a recording system of raw data, quickly integrate an "online" complete system with a short constant delay, and comfortably display the results. Our finger tracking and pitch estimation algorithm implementation also benefit from the same fact.

Our system diagram using OpenInterface both for online and offline recording is shown in Fig. 12, including the multi-cameras, audio, and ARToolkitplus components. Except the other components for video-driving strict synchronization, the Matlab processor component calls recording function of raw data, finger tracking, and pitch estimation algorithms in Matlab.

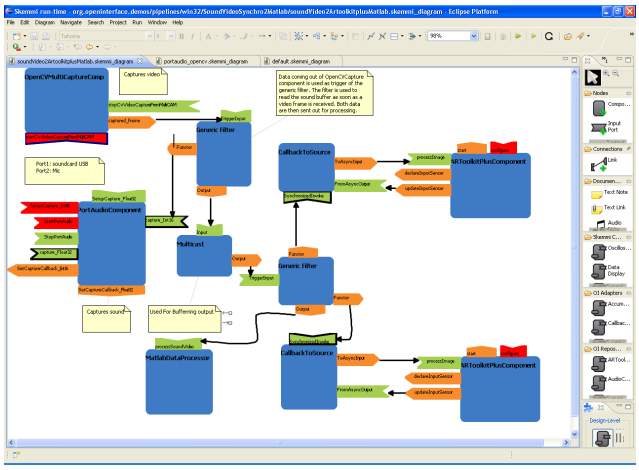

FIGURE 12: System Diagram (online and offline recording) using OpenInterface [42, 43]

4.2. Audio Analysis : Polyphonic Pitch Estimation from a Monophonic Signal

4.2.1. Background

As opposed to the method proposed in section 3.1.1, we wanted to perform an estimation of the fundamental frequency of each note played by the musical instrument without using expensive or non-factory devices such as hexaphonic pickups, not available on most affordable guitars.

The third running of the annual Music Information Retrieval Evaluation eXchange (MIREX) [19] gives an up-to-date overview of methods and algorithms for multiple fundamental frequency estimation & tracking (see the contest results here [51]), yet it focuses on accuracy of the results, thus algorithms don't need to run in real time, a mandatory feature for our system.

Due to the limited time available, we decided to use methods that we already implemented, whose source code was available, and which were designed to work in real time.

Klapuri's method [36] doesn't implement polyphony estimation, but the output salience values can be used to estimate polyphony (eq.(8) in [36]). The polyphony estimation is not very reliable however, especially if levels of component sounds vary (not equal levels).

Arshia Cont's realtime algorithm [13] is based on a non-negative sparse decomposition of the incoming signal as a product of two matrices :

- the first matrix is filled with the spectrum of all the single notes that the instrument can produce, the notes being recorded offline one-by-one so as to train the system ;
- the second matrix, a vector, indexes all the notes that are detected in the audio signal, containing a high value at each row that corresponds to each detected note, a low value for all other non-present notes.

As sound is harmonic by nature, harmonics can overlap between notes. "Sparse" means that the whole playing should be described by the smallest number of note components, given a predefined maximum number of decomposition iterations. Arshia Cont offers for free his algorithm as a `transcribe~` [13, 12] object, for Max/MSP [15] and PureData [57] : to our knowledge, it is the only realtime implementation available, for free, working inside

the opensource PureData modular environment.

O'Grady and Rickard also proposed an algorithm based on non-negative matrix decomposition, but running offline and requiring an hexaphonic pickup [54].

4.2.2. Implementation

We used Arshia Cont's realtime algorithm. In order to have the algorithm work with guitar sounds, we had to record guitar notes one-by-one so as to train the algorithm, using the Matlab code Arshia Cont provided us.

We modified the C++ `transcribe~` [13, 12] flext object [26, 27] object and the Matlab template training source codes so as to exchange template files between the two using MAT files, instead of the former text files that caused us issues with parsing on different OS platforms. We used libmatio, a LGPL C library [47]. Additionaly to the template matrix contained in the template file, we added metadata that is written on the MAT file after the template training and loaded and parsed in the flext object :

- `lowest_midi_pitch` (21 for piano, 40 for guitar, 38 for "Drop-D" guitar), defining how the pitch list is indexed ;
- `fft_size` and `hop_size` for a consistency check (MAT file and the `transcribe~` object settings values won't match, the user is adviced to change the object settings)
- (optional) `instrument`, `author`, `institute`, `date` and `email` for further contacts

We hope that Arshia Cont will integrate our modifications to the next `transcribe~` release. Note that everything in the chain can rely on free software as the template training works as well with Octave [25].

Due to the fact that this algorithm has been shown to provide accuracy of at most 88% [11] for the piano, visual modality should help increase the robustness of the pitch transcription and provide extra information such as identifying which strings have been played.

4.3. Computer Vision Analysis

As a role to provide finger positions and complement the audio modality, we survey the literature and seek a robust method, which should estimate the three dimensional position of four fingers—the two dimensional position above the fingerboard plane and the distance between fingers and the fingerboard. [71] detected fingers on the violin by using the Condensation algorithm [33], in which the joint finger model dynamics and the Gaussian skin color model are employed. However, for the note inference, it does not provide satisfying results (14.9% full-matches and 65.1% partial matches between audio and video results), perhaps because of lack of another camera to provide three-dimensional information.

We adopt a method based on Chutisant Kerdvibulvech's work which made use of two webcams and an integration of the bayesian classifier, ARTag, and Particle Filtering and colored markers to provide four finger three dimensional positions relative to the marker mounted on the guitar [35]. Though Chutisant Kerdvibulvech's paper does not provide scientific evaluation of precision under a large scale test, which is one of the drawbacks and challenges to adopt this method, his survey based on several user survey did showed a subjective positive tracking result.

In his method, the ARTag marker appearing on the images is used to calculate projection matrices used later in the tracking algorithm to map the three dimensional guitar coordinate into two dimensional image coordinate [35]. Bayesian classifier and image processing techniques form a probability map for each pixel of the whole images using pre-trained classifier and adaptive classifier, which is used to reduce the impact from slight change of the lighting condition [35]. Combining these parameters, particle filtering tracks finger markers by generating particles in a three-dimensional guitar coordinate system, and has them mapped onto the two dimensional coordinate using the projection matrices given the probability map as weights on each particle, and finally obtain the average finger markers three-dimensional positions. Two images are used to enhance the precision of the results [35].

Based on the main concepts, we re-implement this method with some modifications : Instead of using ARTag to calculate the projection matrix for calibration of the guitar position, we make use of ARToolKitPlus [1] due to its advantages of high recognition performance, free charge, and the lack of availability of ARTag. Image processing techniques also have been modified. The simplified structure of our implementation has been shown in Fig. 13. The tracking result is shown in Fig. 14. Since we use OpenCV as a main interfacing for video acquisition, the maximum frame rate we can reach is 15 fps due to the limitation of the OpenCV.

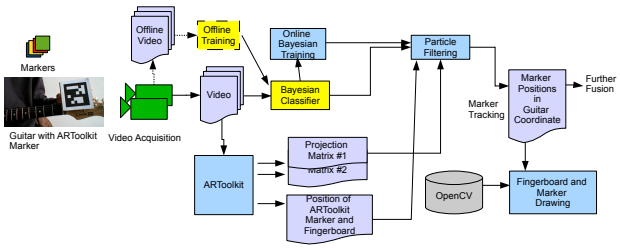

FIGURE 13: Our Finger Tracking Implementation

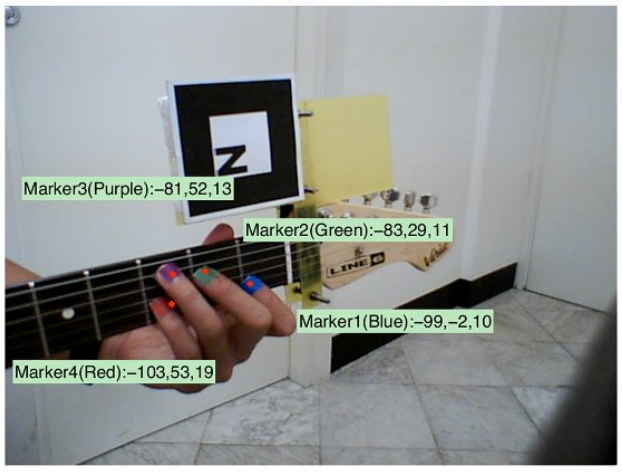

FIGURE 14: Tracking Result of Our Implementation

4.4. Multimodal Fusion and Mapping Scheme

4.4.1. State-of-the-art

Since each modality has its own recognition in each modality, we should choose one among methods of intermediate fusion.

The intermediate fusion methods, like probabilistic graphical model methods, such as Bayesian networks, can handle the imperfect data, generate its conclusion so that its certainty varies according to the input data [34]. Bayesian Networks or Dynamic Bayesian networks is very suitable for fusing our noisy or sometimes even missing results of locating finger position and audio pitch recognition results. Hence, for the fusion method, we adopt the Bayesian Networks method here. In our proposed scheme, weights of each modalities, representing the reliability of each modality, and majority vote techniques are included. The overall system is shown in Fig. 15.

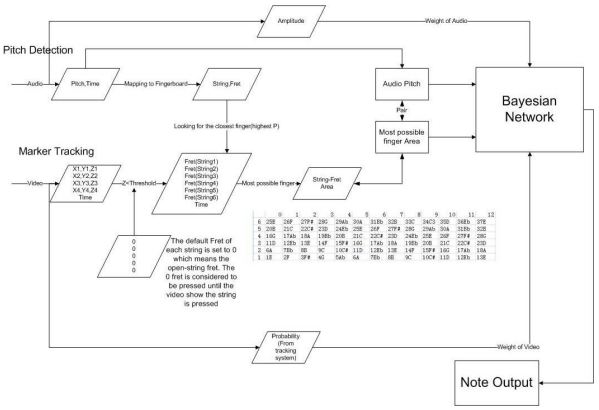

FIGURE 15: Overall Fusion Scheme

4.4.2. *Preprocessing system*

In order to produce the best reasonable finger-pitch pairs to train and test the Bayesian Networks (BN). We need first to synchronize the recognition streams of the two modalities, and then the paring process is needed as the proposed system is a multi-pitch score follower, where more than one finger is tracked and maybe more than one pitch may be played at the same time.

4.4.3. *Synchronization problem of two modalities*

For the video modality, the frame rate is fixed as $30fps$. For the audio modality, consisting in estimating the fundamental frequency of each note played by the musical instrument, we used Arshia Cont's realtime algorithm [13]. The frame rate of informative data stream produced from this algorithm is not fixed. It presents the pitch and its amplitude when there are one or more onset(s) are detected. Thus we use these onset signals as the signs of synchronization. When there is an onset appearence at time T_P, we look for the closest T_F from the video frames and the corresponding frame F_{Tp}. Then we group the pitch happened at T_P and fingers positions from frame F_{Tp} together and mark these groups by T_P.

4.4.4. *Event pairing problem of synchronized data*

After the synchronization processing, now we have the temporal groups containing the data from the audio and video modalities. What we do following that is to pair the pitch data with the most probable finger that played this pitch. In order to do that, we first mapping the pitch to the guitar finger board area according to Fig. 16. As a single pitch could be related to three String-Fret combinations in maximum, we list all possible String-Fret combinations down.

Str \ Fr	0	1	2	3	4	5	6	7	8	9	10	11	12
6	25E	26F	27F$_\#$	28G	29A$_\flat$	30A	31B$_\flat$	32B	33C	34C$_\#$	35D	36E$_\flat$	37E
5	20B	21C	22C$_\#$	23D	24E$_\flat$	25E	26F	27F$_\#$	28G	29A$_\flat$	30A	31B$_\flat$	32B
4	16G	17A$_\flat$	18A	19B$_\flat$	20B	21C	22C$_\#$	23D	24E$_\flat$	25E	26F	27F$_\#$	28G
3	11D	12E$_\flat$	13E	14F	15F$_\#$	16G	17A$_\flat$	18A	19B$_\flat$	20B	21C	22C$_\#$	23D
2	6A	7B$_\flat$	8B	9C	10C$_\#$	11D	12E$_\flat$	13E	14F	15F$_\#$	16G	17A$_\flat$	18A
1	1E	2F	3F$_\#$	4G	5Ab	6A	7B$_\flat$	8B	9C	10C$_\#$	11D	12E$_\flat$	13E

FIGURE 16: Pitch-Finger board mapping table

Then, we need to map the finger data (X, Y, Z) to the corresponding String-Fret as well. In guitar playing, there are six notes that can be played without pressing any fret. They are called 0-Fret notes and they appear commonly during the guitar playing. In the cases that a 0-Fret note is played, it is highly possible that no finger would be tracked on the corresponding 0-Fret area. Our solution to this problem is setting the default frets that are pressed on all strings to 0 at each video modality frame. Then a later modification is made when the finger position data (X, Y, Z) indicates so. When we deal with the finger data (four fingers in maximum) in a temporal group. We look at the Z first. If Z is smaller than or equal to Z_{th} , then we map the corresponding X-Y pair to the String-Fret combination to change the default fret(which is 0) on that string. The Z_{th} here means the threshold Z of pressing. If a Z is smaller than Z_{th}, we consider this finger as in the pressing status. In our case, we chose 8 mm as the Z_{th}. During the mapping from X-Y pair to String-Fret combination, we use the idea of 'area' to classify the finger position. The 'area' of 'string S_A and fret F_B' is defined as from the midpoint of S_{A-1} and S_A, Mid(S_{A-1}, S_A), to Mid(S_A, S_{A+1}) vertically and from Mid(F_{B-1}, F_B) to Mid$(F_B$ and $F_{B+1})$ horizontally. In the case of first string or zero fret, we use Mid(S_2, S_1) and Mid(F_1, F_0) as the low vertical boundary and left horizontal boundary respectively. The distance from the nut to the n^{th} fret can be expressed as

$$D = L - \frac{L}{2^{\frac{n}{12}}} \tag{5}$$

So the distance from the nut to the midpoint of n^{th} and $(n+1)^{\text{th}}$ fret can be expressed as

$$D_m = (1 - \frac{1 + 2^{\frac{1}{12}}}{2^{\frac{n+13}{12}}})L \tag{6}$$

In our case, L is equal to $640mm$ and the distance between two strings is $7.2mm$ in average. Due to the lack of absolute parallelism of six strings, the square area we use here is an approximate one. In order to achieve higher accuracy, we are going to use Neural Networks as the nonlinear transformation function between X-Y and String-Fret in the future improvement. As this is the proto type of system and the lack of parallelism is so little, we will use these square areas for now.

After we scan all the finger data in one temporal group, we will have six Fret-String combinations from video modality and in maximum three Fret-String combinations from audio modality from that group. Then, we calculate the closest (highest possibility) finger that related to the pitch. Now, we have the pitch that is played and the related finger area ready for the BN.

4.4.5. Fusion System : Bayesian Network

Inspired by [50], we created a Bayesian network as our fusion scheme, shown in Fig. 17. According to [50], a Bayesian network over universe U with observed evidence e is expressed in the following equation :

$$P(U,e) = \prod_{A \in U} P(A|pa(A)) \cdot \prod_{i} e_i \tag{7}$$

where $P(U,e)$ is the joint probability of U and e, and $pa(A)$ is the parent set of A. In our network, the observed evidence e is the observed outputs from the pitch detection, the frets and strings from the finger position tracked, as well as their corresponding weight. The evidences are represented by the nodes Pitch Detected (PD), Fret Detected (FD), String Detected (SD) and their corresponding weight is represented by node WA and WV respectively. Note (N), Fret(F), String(S) and Image(I) are the unobserved variables. The Combination Played node (C) is the played score and the fret as well as string that player used which we are trying to find. Each arrow represents a conditional probability. The values of the observed evidence are represented by $PD = pd$, $FD = fd$, $SD = sd$, $WA = wa$, and $WV = wv$, respectively. The inference equation is then derived as :

$$\begin{aligned} P(C|N,I) = & P(Pd) \cdot P(wa) \cdot P(fd) \cdot P(sd) \cdot P(wv) \\ & \cdot P(N|PD, W_a) \cdot P(F|fd, wv) \\ & \cdot P(S|sd, wv) \cdot P(I|F,S) \end{aligned} \tag{8}$$

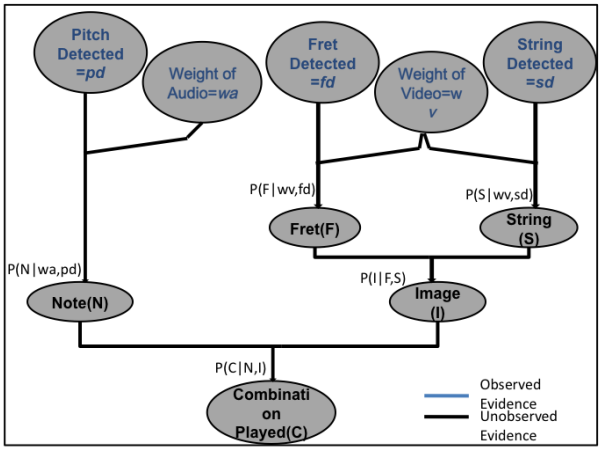

FIGURE 17: The Bayesian Network Diagram

4.4.6. Modality Weight Computation and Majority Vote

Before we feed the data to the BN, the Modality Weight WA and WV reflect the reliability of the two modalities and how much they affect the fusion result. WA and WV are decided by two factors : modality failure detection and modality reliability.

Modality failure detection try to detect if the device is failure : if either audio or video signal within a time frame has no values, then the modality should be give zero weight until non-zero signals are detected.

It is not easy to determine reliability of the each modalities, though many parameters in the algorithm of the two modalities provide limited clues. Hence, instead of having an explicit setup of the weights of the two modalities, by the concept of majority vote, we can generate several reasonably distributed sets of weights. They are multiplied by corresponding failure detection result and then normalized. Each pair of weights will lead to a specific classification results in the Bayesian network, and the result of the majority is our final result. It provide a way to avoid calculating weights explicitly, which might not improve or even degrade the performance of the system.

4.5. Data Recording for Training

While our aim is to recognize multiple notes at the same time, it is reasonable to first reach the single note recognition and the the two note recognition as our preliminary goals. We further limit the range to recognize notes into first 30 positions, that is, from $1E$ to $29A_b$ in Fig. 16. To train the system, at least 10 pairs of video and audio information in each class should be recorded. We have made 300 data samples for single notes, while having to make 10 notes for each common frequent pair(should be much less than $C_2^{30} \cdot 10 = 4350$).

In the recording setup, projection matrices from two images are computed, synchronized, and stored together with an audio vector of 4096 samples in 44.1 kHz, and two images in the size of 288×352 in the same data packet. All these data will be fed to the following recognition algorithm to produce the results. The two images with the ARToolKitPlus [1] output are shown in Fig. 18. The runtime display of the recording system is shown in Fig. 19, where sound samples, projection matrices are drawn at the right hand side using the function of Matlab.

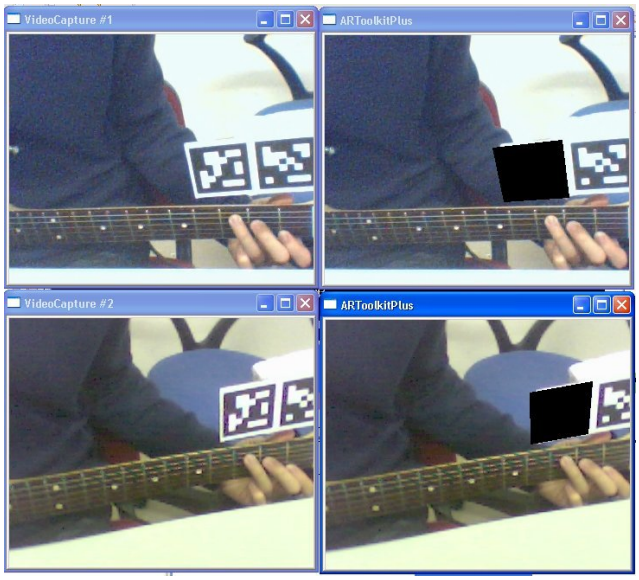

FIGURE 18: Guitar tracking with ARToolkitplus

4.6. Visualization

4.6.1. Available guitar score visualization tools

So as to provide both a visualization of the score to be played by the guitarist and visual feedback on how accurately the score is actually played by the guitarist, we initially planned to modify

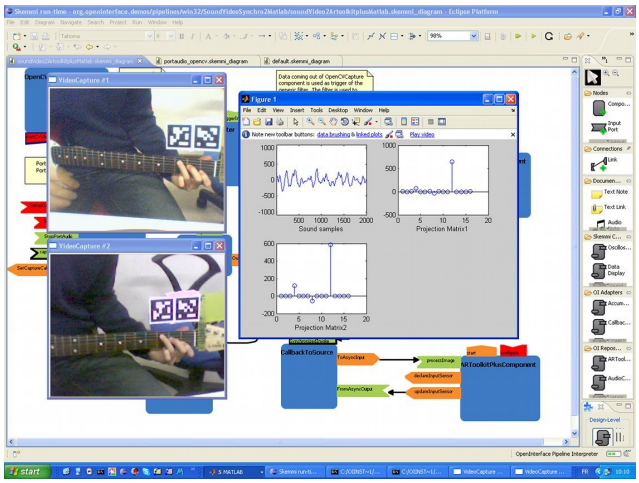

FIGURE 19: Data Recording Runtime

an open-source equivalent of the *Guitar Hero* video game, named Frets On Fire [38], because of its engaging and fun visual interface. While we are still in the design phase of our system and it can not yet estimate in real time the accuracy of the guitar playing, we needed to use a tool that assisted us for the recording of the system training database and the testing of the system. A guitar score visualization application such as TuxGuitar [68] is better aimed for that purpose, as it allows to browse the score backwards in time once the score has been played and annotated. Once the real time implementation of our system will be ready, TuxGuitar will also position itself a perfect companion to Frets On Fire, complementary to the entertaining feel of the latter because the former displays the score with classical and tablature notations necessary to learn the guitar further than technique and fluidity.

The FTM library [66, 32], already available for Max/MSP and in development for PureData [72], offers a classical score visualization with the `ftm.editor` object. Once the porting in PureData is mature enough and if a guitar score visualization is added to the `ftm.editor`, this could be seen as another alternative that would be more tightly integrated into the main framework integrating modalities, PureData for the realtime version of our system.

4.6.2. *Choice and modifications*

We achieved initial results by creating an OpenSoundControl plugin for TuxGuitar that allows remote control through the OSC protocol, both on Matlab and patcher applications such as Pd and Max/MSP. Currently supported commands are :

- loading a score file (`/load <file_location>`),
- initiating the playback (`/play`),
- stopping or pausing the playback (`/stop`).

We still need to modify the plugin so as to provide the visual feedback on the accuracy of the playing. Current notes to be played on the score are colored in red by default in TuxGuitar. We plan to use colors as well to label and distinguish accurately played notes (for example, in green) and wrong/missed notes (for example, in orange) and the note to be played on the score (for example, in red, the most salient color).

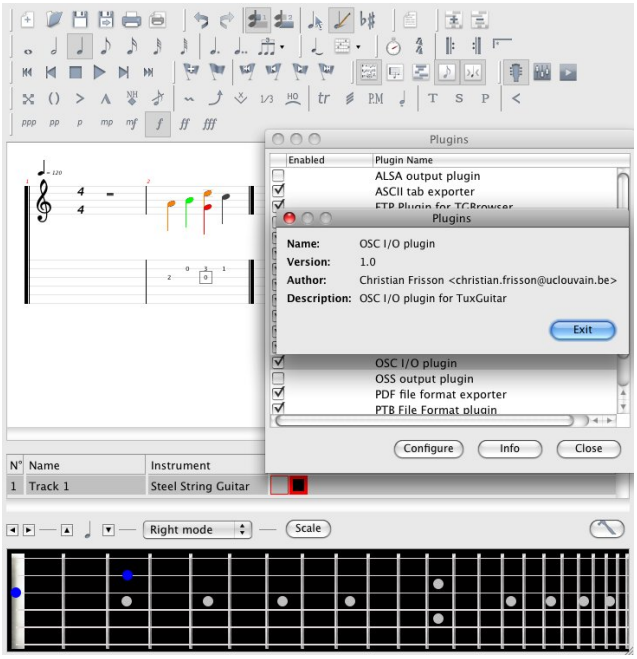

FIGURE 20: Mockup of the TuxGuitar [68] interface featuring colored notes giving a visual feedback of the score following. A first version of our OSC communication plugin is also set on.

5. CONCLUSION AND PERSPECTIVES

5.1. Performance Toolbox

We achieved a usable toolbox for hexaphonic or monophonic guitar. These tools are available for the Max/MSP [15] and/or PureData [57] environments. We worked on all the part of the chain of augmented music, from extraction and use of audio signal features to digital audio effects manipulated by gestural control with sensors. All the tools (Polyphonic Pitch Extraction, Fretboard Grouping, Rear-Mounted Pressure Sensors, Modal Synthesis, Infinite Sustain, Rearranging Looper, Smart Harmonizer) need to be developed for both environments (i.e Max/MSP and PureData) and documentation and/or tutorials will be provided so that everything can be freely downloadable on the eNTERFACE'09 and numediart websites and be directly usable.

Concerning the Fretboard Grouping tool, more efforts will be put in the chord / arpeggio discrimination (i.e not to base it only on the time needed to play the group). To achieve a better discrimination, one track that we will follow, will be to add time between notes directly in the group definition. Doing that, the chord / arpeggio discrimination would become obsolete as groups will not be considered anymore only as a gathering of notes but as a gathering of notes through time. A group record feature can then be added to the module so that one can record a group played in a specific way.

Physical sound synthesis sometimes lacks realism. One interesting approach can be to use pre-recorded sounds relevant to specific play on guitar such as sliding, slapping, etc.., to add more texture to the modal sounds. Since we proposed granular synthesis to enrich the guitar sustain, we could collect audio grains for enhancement of both modal sounds and sustains. By using the approach from Picard et al. [60], audio grains could be automatically

extracted from recordings. In addition, the audio grains could be properly retargeted to specific outputs of sensors during runtime.

5.2. Study Workbench

We built a careful design of the multimodal score following system. We prototyped many algorithms for : finger tracking, guitar headstock 3D location, multiple pitch estimation from a monophonic audio signal of the guitar, bayesian networks for multimodal fusion and synchronized modalities recording (mono audio, multiple cameras). Each module works efficiently offline and separately. We build an initial synchronized recordings database.

We still need to do a proper study and working implementation of the score visualization and visual feedback of the score following. We need to test and improve the system so that it can run in real time. We also need to undertake user testing so as to evaluate and validate the precision and performance of the system. To complement the multimodal fusion, we could build an ontology of the guitar playing domain, by extracting, classifying and interpreting "guitar techniques", such as "bends", "hammer-ons", "pull-offs", "artificial harmonics", the plucking position [67].

6. ACKNOWLEDGMENTS

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

Wen-Yang Chu is funded by the FRIA grant of FNRS, Région Wallonne, Belgium.

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

Cécile Picard work is partially funded by Eden Games, and ATARI Game Studio in Lyon, France.

Ao Shen is supported and funded by his supervisor Neil Cooke and EECE Department, University of Birmingham.

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 the 1-month workshop of this project took place.

We would like to thank Ashia Cont (IRCAM) for having provided us the source codes for the `transcribe~` flext object (C++) [12] and the related template training (Matlab). We are grateful to Damien Tardieu (FPMs/TCTS) for having helped us modify the aforementioned code to adapt it to the guitar.

We offer thanks to Emmanuel Vincent and Anssi Klapuri who provided us their implementation of the polyphonic pitch estimation.

We show our appreciation to Jean-Yves Lionel Lawson (UCL-TELE) for assisting building the recording and integrated OpenInterface system of the "edutainment" part.

We would like to thank Otso Lähdeoja, Anne Sedes and the organizing team from CICM, for having welcomed us at Identités de la Guitare Electrique - Journées d'étude interdisciplinaires à la Maison des Sciences de l'Homme Paris Nord, on May 18-19 2009, to present the objectives of the Multimodal Guitar project. A publication on the proceedings will follow [22].

7. REFERENCES

7.1. Scientific references

[2] John Bowers and Phil Archer. « Not Hyper, Not Meta, Not Cyber but Infra-Instruments ». In: *Proceedings of the 2005 International Conference on New Interfaces for Musical Expression (NIME05)*. 2005. Pp. 5–10. P.: 1.

[3] Ivica Ico Bukvic et al. « munger1~ : towards a cross-platform swiss-army knife of real-time granular synthesis ». In: *Proc. ICMC*. 2007. URL: http://ico.bukvic.net/PDF/ICMC2007_munger1.pdf. P.: 7.

[5] Anne-Marie Burns. « Computer Vision Methods for Guitarist Left-Hand Fingering Recognition ». MA thesis. McGill University, 2006. Pp.: 1, 9.

[6] Anne-Marie Burns and Marcelo M. Wanderley. « Visual Methods for the Retrieval of Guitarist Fingering ». In: *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06)*. Paris, France 2006. Pp. 196–199. Pp.: 1, 9.

[7] Ozan Cakmakci, François Bérard, and Joëlle Coutaz. « An Augmented Reality Based Learning Assistant for Electric Bass Guitar ». In: *Proceedings of the International Conference on Human Interaction (CHI03)*. 2003. P.: 1.

[8] Gavin Carfoot. « Acoustic, Electric and Virtual Noise : The Cultural Identity of the Guitar ». In: *Leonardo Music Journal* 16 (2006). Pp. 35–39. P.: 1.

[9] Alain de Cheveigné. « Procédé d'extraction de la fréquence fondamentale d'un signal sonore au moyen d'un dispositif mettant en oeuvre un algorithme d'autocorrélation ». Pat. 01 07284. 2001. P.: 2.

[10] Alain de Cheveigné and Hideki Kawahara. « YIN, a fundamental frequency estimator for speech and music ». In: *J. Acoust. Soc. Am.* 111.4 (Apr. 2002). Pp. 1917–1930. P.: 2.

[11] Arshia Cont. « Realtime Multiple Pitch Observation using Sparse Non-negative Constraints ». In: *International Symposium on Music Information Retrieval (ISMIR)*. 2006. URL: http://cosmal.ucsd.edu/arshia/papers/ArshiaCont_ismir2006.pdf. P.: 11.

[13] Arshia Cont, Shlomo Dubnov, and David Wessel. « Realtime Multiple-pitch and Multiple-instrument Recognition For Music Signals using Sparse Non-negative Constraints ». In: *Proceedings of Digital Audio Effects Conference (DAFx)*. 2007. URL: http://cosmal.ucsd.edu/arshia/index.php?n=Main.Transcribe. Pp.: 10–12.

[14] Perry R. Cook. *Real Sound Synthesis for Interactive Applications*. A. K. Peters, 2002. P.: 6.

[17] Kees van den Doel, Paul Kry, and Dinesh K. Pai. « FoleyAutomatic : Physically-based Sound Effects for Interactive Simulation and Animations ». In: *ACM SIGGRAPH 01 Conference Proceedings*. 2001. Chap. Modal Synthesis for Vibrating Objects. ISBN: 978-1568812151. URL: http://www.cs.ubc.ca/~kvdoel/publications/foleyautomatic.pdf. P.: 6.

[18] Kees van den Doel and Dinesh K. Pai. « Audio Anecdotes III : Tools, Tips, and Techniques for Digital Audio ». In: ed. by Ken Greenebaum and Ronen Barzel. 3rd ed. Source code available at http://www.cs.ubc.ca/~kvdoel/publications/srcmodalpaper.zip. A. K. Peter, 2006. Chap. Modal Synthesis for Vibrating Objects, pp. 99–120. ISBN: 978-1568812151. URL: http://www.cs.ubc.ca/~kvdoel/publications/modalpaper.pdf. P.: 7.

[19] J. Stephen Downie. « The music information retrieval evaluation exchange (2005-2007) : A window into music information retrieval research ». In: *Acoustical Science and Technology* 29.4 (2008). Pp. 247–255. P.: 10.

[20] Richard Mark French. *Engineering the Guitar : Theory and Practice*. Springer, 2008. ISBN: 9780387743684. P.: 1.

[21] G. Friedland, W. Hurst, and L. Knipping. « Educational Multimedia ». In: *IEEE Multimedia Magazine* 15.3 (2008). Pp. 54–56. ISSN: 1070-986X. DOI: 10.1109/MMUL.2008.71. P.: 1.

[22] Christian Frisson et al. « eNTERFACE'09 Multimodal Guitar project : Performance Toolkit and Study Workbench ». In: *Actes des Journées d'étude interdisciplinaires sur l'Identités de la Guitare Electrique*. (to appear). Maison des Sciences de l'Homme Paris Nord, Paris, France 2010. URL: http://www.guitarelectrique.fr. P.: 15.

[24] O. Gillet and G. Richard. « Automatic transcription of drum sequences using audiovisual features ». In: *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05)*. Vol. 3. 2005. P.: 9.

[26] Thomas Grill. « flext - C++ layer for Pure Data & Max/MSP externals ». In: *The second Linux Audio Conference (LAC)*. 2004. URL: http://lad.linuxaudio.org/events/2004_zkm/slides/thursday/thomas_grill-flext.pdf. Pp.: 7, 11.

[31] Francisco Iovino, René Caussé, and Richard Dudas. « Recent work around Modalys and Modal Synthesis ». In: *ICMC : International Computer Music Conference*. Thessaloniki Hellas, Greece 1997. Pp. 356–359. P.: 6.

[33] M. Isard and A. Blake. « Contour tracking by stochastic propagation of conditional density ». In: *Lecture Notes in Computer Science* 1064 (1996). Pp. 343–356. P.: 11.

[34] Alejandro Jaimes and Nicu Sebe. « Multimodal human-computer interaction : A survey ». In: *Computer Vision and Image Understanding* 108.1-2 (2007). Special Issue on Vision for Human-Computer Interaction. Pp. 116–134. ISSN: 1077-3142. P.: 12.

[35] Chutisant Kerdvibulvech and Hideo Saito. « Guitarist Fingertip Tracking by Integrating a Bayesian Classifier into Particle Filters ». In: *Advances in Human-Computer Interaction* 2008 (2008). P.: 11.

[36] A. Klapuri. « Multiple fundamental frequency estimation by summing harmonic amplitudes ». In: *7th International Conference on Music InformationRetrieval (ISMIR-06)*. 2006. P.: 10.

[37] David Kushner. « The Making of The Beatles : Rock Band ». In: *IEEE Spectrum* (Oct. 2009). Pp. 26–31. URL: http://spectrum.ieee.org/consumer-electronics/gaming/the-making-of-the-beatles-rock-band. P.: 1.

[39] Otso Lähdeoja. « An Approach to Instrument Augmentation : the Electric Guitar ». In: *Proceedings of the 2008 Conference on New Interfaces for Musical Expression (NIME08)*. 2008. P.: 1.

[40] Otso Lähdeoja. « Guitare électrique augmentée : une approche du contrôle gestuel des "effets" de la guitare électrique ». In: *Articles des Journées d'Informatique Musicale*. 2008. P.: 1.

[41] Otso Lähdeoja. « Une approche de l'instrument augmenté : Le cas de la guitare électrique ». In: *Actes de la conférence francophone d'Interaction Homme-Machine (IHM)*. 2007. URL: http://www.lahdeoja.org/ftplahdeoja/augmented_guitar/otso.lahdeojaIHM08.pdf. P.: 1.

[42] J.Y.L. Lawson et al. « An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components ». In: *Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems*. ACM. 2009. Pp. 245–254. Pp.: 9, 10.

[44] Christophe Leduc. « Instruments de musique à cordes frottées ou pincées ». Pat. FR2677160. Dec. 4, 1992. P.: 6.

[45] Christophe Leduc. « Musical instruments having bowed or plucked strings ». Pat. US5339718. Aug. 23, 1994. P.: 6.

[46] Nicolas Leroy, Emmanuel Fléty, and Frederic Bevilacqua. « Reflective Optical Pickup For Violin ». In: *Proceedings of the 2006 International Conference on New Interfaces for Musical Expression (NIME06)*. 2006. P.: 1.

[50] Ankush Mittal and Ashraf Kassim. *Bayesian Network Technologies : Applications and Graphical Models*. Hershey, PA, USA: IGI Publishing, 2007. ISBN: 1599041413, 9781599041414. P.: 13.

[52] Axel Nackaerts, Bart De Moor, and Rudy Lauwereins. « Measurement of guitar string coupling ». In: *Proceedings of the International Computer Music Conference (ICMC)*. 2002. URL: ftp://ftp.esat.kuleuven.ac.be/pub/SISTA/nackaerts/reports/report_ICMC2002.ps.gz. Pp.: 2, 3.

[53] James F. O'Brien, Chen Shen, and Christine M. Gatchalian. « Synthesizing sounds from rigid-body simulations ». In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA'02)*. ACM, 2002. Pp. 175–181. ISBN: 1-58113-573-4. P.: 6.

[54] Paul D. O'Grady and Scott T. Rickard. « Automatic Hexaphonic Guitar Transcription Using Non-Negative Constraints ». In: *Proceedings of the Irish Signal and Systems Conference*. 2009. URL: http://eleceng.ucd.ie/~pogrady/papers/OGRADY_RICKARD_ISSC09.pdf. P.: 11.

[55] Dan Overholt. « The Overtone Violin ». In: *Proceedings of the 2005 Conference on New Interfaces for Musical Expression (NIME05)*. 2005. P.: 1.

[56] M. Paleari et al. « A multimodal approach to music transcription ». In: *15th IEEE International Conference on Image Processing, 2008. ICIP 2008*. 2008. Pp. 93–96. P.: 9.

[58] Henri Penttinen, Aki Härmä, and Matti Karjalainen. « Digital Guitar Body Mode Modulation With One Driving Parameter ». In: *Proceedings of the COSTG-6 Conference on Digital Audio Effects (DAFX-00)*. Sound samples available at http://www.acoustics.hut.fi/demos/dafx2000-bodymod/. 2000. P.: 6.

[59] Henri Penttinen, Matti Karjalainen, and Aki Härmä. « Morphing Instrument Body Models ». In: *Proceedings of the COSTG-6 Conference on Digital Audio Effects (DAFX-01)*. Sound samples available at http://www.acoustics.hut.fi/demos/dafx2001-bodymorph/. 2001. P.: 6.

[60] Cécile Picard, Nicolas Tsingos, and François Faure. « Retargetting Example Sounds to Interactive Physics-Driven Animations ». In: *AES 35th International Conference on Audio for Games*. 2009. URL: http://www-sop.inria.fr/reves/Basilic/2009/PTF09. P.: 14.

[61] Cécile Picard et al. « A Robust And Multi-Scale Modal Analysis For Sound Synthesis ». In: *Proceedings of the International Conference on Digital Audio Effects*. 2009. URL: http://www-sop.inria.fr/reves/Basilic/2009/PFDK09. P.: 6.

[62] Cornelius Poepel. « Synthesized Strings for String Players ». In: *Proceedings of the 2004 Conference on New Interfaces for Musical Expression (NIME04)*. 2004. P.: 1.

[63] Miller Puckette. « Patch for guitar ». In: *Pd-convention*. Workshop Pd patches available here http://crca.ucsd.edu/~msp/lac/. 2007. URL: http://crca.ucsd.edu/~msp/Publications/pd07-reprint.pdf. P.: 2.

[64] G Quested, R D Boyle, and K Ng. « Polyphonic note tracking using multimodal retrieval of musical events ». In: *Proceedings of the International Computer Music Conference (ICMC)*. 2008. URL: http://www.comp.leeds.ac.uk/roger/Research/Publications/Garry08.pdf. P.: 9.

[66] Norbert Schnell et al. « FTM - Complex Data Structures for Max ». In: *Proceedings of the International Conference on Computer Music (ICMC)*. 2005. URL: http://recherche.ircam.fr/equipes/temps-reel/articles/ftm.icmc2005.pdf. P.: 14.

[67] Caroline Traube and Philippe Depalle. « Extraction of the excitation point location on a string using weighted least-square estimation of comb filter delay ». In: *Proceedings of the Conference on Digital Audio Effects (DAFx)*. 2003. URL: http://www.elec.qmul.ac.uk/dafx03/proceedings/pdfs/dafx54.pdf. P.: 15.

[69] Y. Wang, B. Zhang, and O. Schleusing. « Educational violin transcription by fusing multimedia streams ». In: *Proceedings of the international workshop on Educational multimedia and multimedia education*. ACM New York, NY, USA. 2007. Pp. 57–66. P.: 9.

[70] Shi Yong. *Guitar Body Effect Simulation : a warped LPC spectrum estimation and a warped all-pole filter implemented in Matlab and C++*. Course Project MUMT 612 : Sound Synthesis and Audio Processing. McGill University, 2007. URL: http://www.music.mcgill.ca/~yong/mumt612/mumt612.html. P.: 6.

[71] B. Zhang et al. « Visual analysis of fingering for pedagogical violin transcription ». In: *Proceedings of the 15th international conference on Multimedia*. ACM New York, NY, USA. 2007. Pp. 521–524. P.: 11.

[72] IOhannes Zmölnig et al. « Freer Than Max - porting FTM to Pure data ». In: *Proceedings of the Linux Audio Conference (LAC-2008)*. 2008. URL: http://lac.linuxaudio.org/2008/download/papers/20.pdf. P.: 14.

[73] Amit Zoran and Pattie Maes. « Considering Virtual and Physical Aspects in Acoustic Guitar Design ». In: *Proceedings of the 2008 Conference on New Interfaces for Musical Expression (NIME08)*. 2008. P.: 6.

7.2. Software, hardware and technologies

[1] « ARToolKitPlus ». v.2.1.1. 2006. URL: http://studierstube.icg.tu-graz.ac.at/handheld_ar/artoolkitplus.php. Pp.: 9, 11, 13.

[4] Ivica Ico Bukvic et al. « munger1~ : towards a cross-platform swiss-army knife of real-time granular synthesis ». v. 1.3.2. 2009. URL: http://ico.bukvic.net/Max/disis_munger~_latest.zip. P.: 7.

[12] Arshia Cont. « transcribe , Pd/Max/MSP object for Real-time Transcription of Music Signals ». 2007. URL: http://cosmal.ucsd.edu/arshia/index.php?n=Main.Transcribe. Pp.: 10, 11, 15.

[15] Cycling'74. « Max/MSP ». URL: http://www.cycling74.com. Pp.: 2, 3, 10, 14.

[16] Mark Danks et al. « GEM (Graphics Environment for Multimedia) ». v.0.91. 2009. URL: http://gem.iem.at. P.: 4.

[23] Pascal Gauthier. « pdj, a java external plugin for pure data ». URL: http://www.le-son666.com/software/pdj/. P.: 4.

[25] « GNU Octave ». v.3.2.2. 2009. URL: http://www.octave.org. P.: 11.

[27] Thomas Grill. « flext - C++ layer for Pure Data & Max/MSP externals ». retrieved from SVN. 2009. URL: http://puredata.info/Members/thomas/flext/. Pp.: 7, 11.

[28] DIST-University of Genova InfoMus Lab. « The EyesWeb XMI (eXtended Multimodal Interaction) platform ». Version 5.0.2.0. URL: http://www.eyesweb.org. Pp.: 1, 9.

[29] INRIA. « The SOFA Framework (Simulation Open Framework Architecture) ». GPL license. 2009. URL: http://www.sofa-framework.org. P.: 6.

[30] Interface-Z. « Sensors and sensor interfaces ». URL: http://interface-z.com. Pp.: 2, 5.

[32] IRCAM. « FTM ». URL: http://ftm.ircam.fr. P.: 14.

[38] Sami Kyöstilä et al. « Frets On Fire ». Version 1.3.110. URL: http://fretsonfire.sourceforge.net. Pp.: 4, 14.

[43] Lionel Lawson. « The OpenInterface platform ». 2009. URL: http://www.openinterface.org. Pp.: 9, 10.

[47] « MAT File I/O Library ». v. 1.3.3 used, LPGL license. 2009. URL: http://sourceforge.net/projects/matio/. P.: 11.

[48] MathWorks. « Matlab ». v. 2008b. URL: http://www.mathworks.com. P.: 10.

[49] Keith Mc Millen. « StringPort ». URL: http://www.keithmcmillen.com. P.: 2.

[51] « Multiple Fundamental Frequency Estimation & Tracking Results ». 2007. URL: http://www.music-ir.org/mirex/2007/. P.: 10.

[57] « pd-extended, a PureData installer including most of the libraries from the Pd CVS repository ». Most recent release (0.40.3). URL: http://puredata.info/downloads. Pp.: 1–3, 9, 10, 14.

[65] Roland. « GK3 hexaphonic pickup and related hardware ». URL: http://www.roland.com. P.: 2.

[68] « TuxGuitar ». v. 1.1, GPL license. 2009. URL: http://tuxguitar.herac.com.ar. Pp.: 4, 14.