VISUAL ANALYTICS FINAL PROJECT: EVALUATION OF MUSICAL GUITAR PERFORMANCES

Eduard Vergés Franch 207530 eduard.verges02@estudiant.upf.edu

1. Objective

The current pandemic has shown that online tools are important for educating, working remotely or socializing. Music conservatories are one of the institutions that have suffered more the consequences of not being able to make presential lessons. Trying to do instrument online classes can be difficult due to quality of the sound, connexion problems and not being able to receive a proper feedback of the professor, whose decisions and observations are being affected for all the above mentioned. Furthermore, music is a pretty common modality of art practiced by many people around the world and in many different knowledge levels.

Music is an important market and the technology must provide solutions to it. In this context arises MTG (Music Technology Group) [1], a research group installed on UPF and in which I will do my internship. Taking into profit the work that I will do there, I decided to do my final project about:

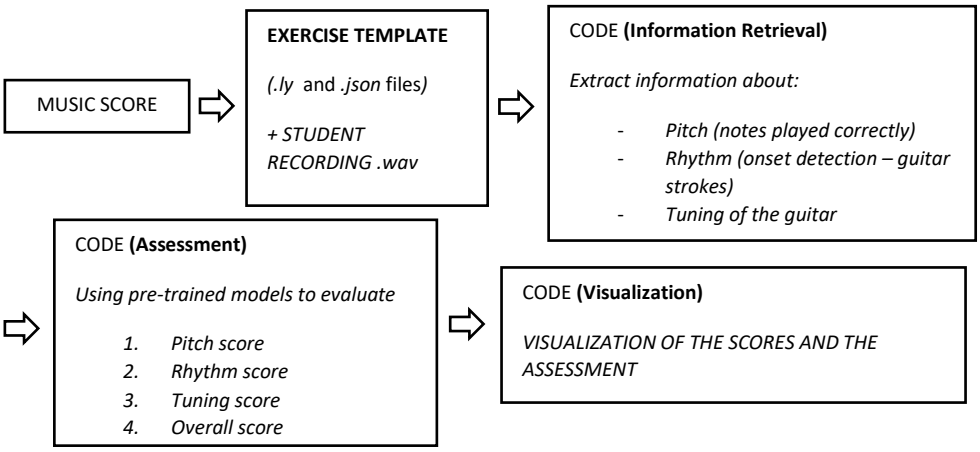
Assessment of guitar music performances for online education

The aim of this work is to implement in local host a script that can assess how a student plays a specific musical exercise. I am going to use some python libraries which are private from the MTG, hence it wouldn't be possible to run the code without having access to that libraries. This work can be classified in the field of MIR (Music Information Retrieval) and MPA (Music Performance Assessment).

2. Workflow, contributions and methodology

Here I am going to describe my contributions to the project and the methodology I followed.

2.1. Workflow of the system



The baseline framework I used is explained in [2]. Basically, it has 3 main steps:

- **Information Retrieval Step:** Here signal processing methods are applied to extract information about the ***pitch*** of the notes played (using NNLS Chroma Classes and then performing a normalization step to reduce the dimensionality and extract probabilities of belonging to a note class), ***rhythm*** (using Spectral Flux onset detection) and ***tuning*** (comparing spectral peaks to equal temperament values). The chroma classes are vectors with 12 dimensions each one

representing a class:

1	2	3	4	5	6	7	8	9	10	11	12
C	C#	D	D#	E	F	F#	G	G#	A	A#	B

- **Assessment:** This part uses pre-trained models such as *'picking_workflow.pkl'* used to evaluate scores with only single notes or *'strumming_worklflow.pkl'* used to evaluate scores with only chords. In my case, I use the first one combined with *'new_model.pkl'* which makes possible to merge chords and notes into a single score. This model takes statistics extracted from the fields mentioned above in (1.) and generate a **score from 1 to 4, being 1 bad and 4 excellent**. They were trained with a little database manually labelled by two musicians, as this project is still in an initial phase. I generated a tableau visualization where you can see a high bias in the score prediction of rhythm and chroma score estimators on 'picking_workflow.pkl' model [3] by running many simulations of possible input statistics.
- **Visualization:** This part generates 2 *.png* files. The first one contains the visualization of the assessment. The notes are coloured depending on the level of correctness (probability) that the signal processing step has obtained. Red indicates bad and green good. In the sound wave, you can observe the deviations of the student performance (areas coloured) from the ground truth beats (black vertical line). As larger the deviation, the area turns more read.

2.2. Contributions

- Create a new exercise template. (lily_was_here.*json* and lily.ly / lily2.*ly* files in *data/* folder)
- Improve visualization of the assessment (make easier to detect fails).
- Improve visualization of the scores (simple and effective).
- Add a Chroma gram (observe the chroma predictions) and a histogram (attack deviations).

2.2. Methodology

The first step was to install all the dependencies needed for executing the code of the MTG libraries. The code works only in *python 3.6.8* and requires, between others, *Essentia [4]* (MIR python library implemented by MTG and publicly available) and *lilypond [5]* (Music annotation software).

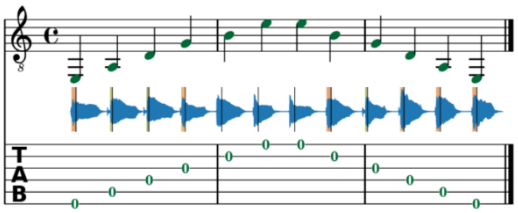


Image 1

After the installation, I had to navigate through different MTG GitHub repositories and put together a code which was able to take as input a music exercise template and a student recording generating a visualization of the performance assessment like *Image1*. This part has the difficulty that there is not been stablished yet a general pipeline to assess exercises. Hence, each exercise can have different processing steps. In my particular case, I had to use *a hacked model* for being able to evaluate a score combining chords and single notes.

When I was able to run this process locally, my next step was to create an exercise template. For this I take the two first lines of the song *Lily Was Here ~ David A. Stewart and Candy Dulfer*. You can find the music score attached in the *data/* repository of this project. The template consists of two files a *.ly (lilypond) file* that make us able to personalize the visualization (colour, sizes …) plus a *.json* (which is the representation of the music score using the symbolic annotation explained in [6]).

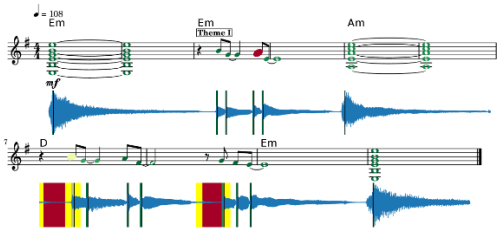


Image 2

My next step was to improve the visualization of the results. For this purpose, I created a macro in the lilypond file which **augment the size of the notes that are being played badly** and also I add code to

highlight the regions where the rhythm from the student deviates much from the original beats (example in Image 2). This makes easier to detect errors (taking into account an educational approach).

Finally, I improve the visualization of the results. Before they where only printed in the screen with a simple *print*. Now, I create a function that returns a *.png* file containing a radar chart with the visualization of the results (example in Image 3).

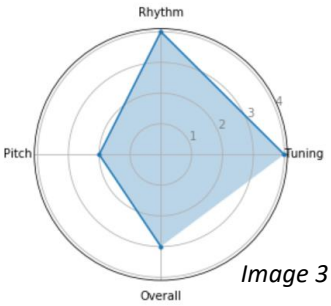


Image 3

The final work I obtained, can take a recording of a student playing the first two lines of Lily Was Here and generate two different *.png* files. The first one contains the visualization of the assessment and it is called *'audiofilename_assessment.png'* (like Image 2) and the second one contains the scores assigned automatically by the system called *'audiofilename_scores.png' (like Image 3)*. The exercise wasn't already been implemented and the visualizations have been improved.

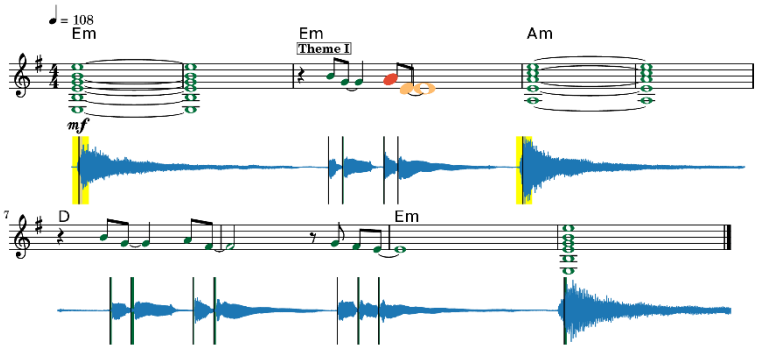
3. Results and Conclusions

3.1 Results explanation and example

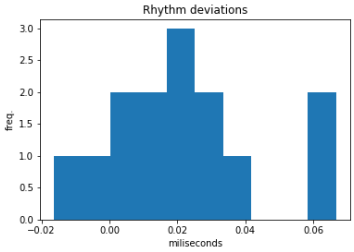
To test my final version of the system I recorded myself playing the song many times, trying to recreate different possible input situations. All the .wav files generated are in the *data/* folder. All the different results obtained are in the *results/* folder. Each recording represents a situation:

Acustic_est.wav	Standard performance (recorded in stereo).
Acustic_mono.wav	Same as before but recorded in mono. See that the score changes due to different audio properties.
Bad_pitch.wav	Performance with many pitch errors
Bad_tempo.wav	Performance with many rhythm errors
Good_performance.wav	Trying to obtain a nice performance. Not easy due to bias in models. Despite being subjectively good, obtaining a 2.0 in rhythm and pitch.
noise.wav	Performance with noises that are detected as guitar strokes
Std_rec_LWH_1_filtered.wav	This performance was recorded by me before all the others. I try to illustrate that the ground truth beats are change (because must be manually computed) and this change depending on the recording method used.

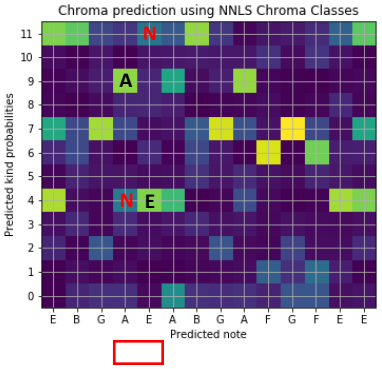
Let's observe some outputs that I have obtained. We will start with a recording representing a standard played performance (*Acustic_est.wav*). This are the results that are visualized:



This are some statistics I obtained:



-->Rhythm variation: 0.0. Good rhythm with small variation as seen in the histogram.
-->Chroma variation: 0.3573. Two notes were badly predicted
-->Tuning variation: 0.0716. Good tuning.



We see that the player has deviated a little bit from the rhythm in the first and the third bars (highlighted areas). Also, we observe that there are two notes that have been played (or predicted by the signal processing steps) badly. In this case, the **notes where played well but the system has predicted them bad**. If we look at the predictions (chroma gram in the right) of the two notes that were coloured with "red" (the first *A* and *E*) we observe that the signal processing steps has obtained noise *(class 4* and class *11* respectively) causing the prediction not being exactly. Noise is represented by **N** in the chroma gram.

These two miss predictions have caused obtaining a 2.0 in Pitch (as observed in [2], the variance of the chroma must be very low to obtain a 4.0 score). With rhythm we were able to obtain a 4.0, meaning that has been mostly perfect (except for very insignificant deviations).

If we look at the timing statistics obtained:

Timing precision-> 1.0, recall-> 1.0 and f-measure->1.0

The precision is calculated by matching all the. json notes (real segments in 'beats') to the predicted segments (note segments of audio in the input audio). If all the segments match the precision and recall will be 1. Due to high bias on the model, for obtaining a 4.0 in rhythm we must have a f-measure of at least 0.97 and a deviation close to 0.0. (The deviation only considers those notes that have deviated more than a threshold from the ground truth). In this case I got a 4.0 of rhythm.

The overall score for the moment is calculated as: *maximum ("Overall", minimum ("Tuning","Rhythm","Pitch"))*. This is a temporal solution before having enough data to generate better predictions.

3.2 Conlcusions

After testing for each audio data in *data/* folder I reached to some conclusions:

- Noises may cause problem as observed in the result of *noise.wav* where the system detects two noises as guitar strokes. Also in the chroma gram above we observe problems with predictions due to noise.
- The bias causes results that subjectively are good in rhytm or pithch like *Good_performance.wav* obtaining a low score.
- Models have to be improved training them with a bigger dataset (in fact, this is the main objective of my intership).
- The new visualizations make able to detect the errors much easier. In case the music score being long, the biggest fails can be detected easily.
- Difficult to predict the ground truth beats. MIR tools still don't provide a exact solution and must be performed using software tool such as (Sonic Visualizer [7]) and adjusting it manually.
- Powerfull system for musical online education.

4. Annexes

[1] MTG: https://mail.google.com/mail/u/1/#inbox/FMfcgxwKjngTcFVKdlrtKmfPgspCPwQb

[2] Music Critic Framework: https://repositori.upf.edu/handle/10230/44130?locale-attribute=en

[3] Bias visualization according to deviation (variance of the results) and f-measure in the case of rhythm.

https://public.tableau.com/profile/eduard.verg.s#!/vizhome/BIAS_FP_VA/Bias

[4] ESSENTIA: https://essentia.upf.edu/index.html

[5] Lilypond: http://lilypond.org/

[6] Symbolic Representation of music score:
https://www.researchgate.net/publication/220723539_Symbolic_Representation_of_Musical_Chords_A_Proposed_Syntax_for_Text_Annotations/link/02e7e518cea7d7374d000000/download

[7] Sonic Visualizer: https://www.sonicvisualiser.org/

4.1. Extra information:

There is an available old demo of music critic in: https://musiccritic.upf.edu/#demo

To predict the chroma (notes) we use the NNLS Chroma classes. This text explains how the NNLS Chroma works and was obtained from: http://www.isophonics.net/nnls-chroma

NNLS Chroma analyses a single channel of audio using frame-wise spectral input from the Vamp host. The spectrum is transformed to a log-frequency spectrum (constant-Q) with three bins per semitone. On this representation, two processing steps are performed: tuning, after which each centre bin (i.e. bin 2, 5, 8, …) corresponds to a semitone, even if the tuning of the piece deviates from 440 Hz standard pitch, and running standardisation: subtraction of the running mean, division by the running standard deviation. This has a spectral whitening effect.

The processed log-frequency spectrum is then used as an input for NNLS approximate transcription (using a dictionary of harmonic notes with geometrically decaying harmonics magnitudes). The output of the NNLS approximate transcription is semitone spaced. To get the chroma, this semitone spectrum is multiplied (elementwise) with the desired profile (chroma or bass chroma) and then mapped to 12 bins. The resulting chroma frames can be normalised by (dividing by) their norm (L1, L2 and maximum norm available).