Master's Thesis on Sound and Music Computing

Universitat Pompeu Fabra

Computational modelling of expressive music performance in hexaphonic guitar

Marc Siquier Peñafort

Supervisor: Sergio Giraldo Méndez

September 2017

Master's Thesis on Sound and Music Computing

Universitat Pompeu Fabra

Computational modelling of expressive music performance in hexaphonic guitar

Marc Siquier Peñafort

Supervisor: Sergio Giraldo Méndez

September 2017

"We learned more from a three minute record
than we ever learned in school"

Springsteen, Bruce. "No Surrender". *Born in the U.S.A.*

Acknowledgement

The present work was carried out at Music and Machine Learning Department of the Music Technology Group at Universitat Pompeu Fabra, Barcelona, Spain.

First of all, I want to thank the supervisor of this thesis Sergio Giraldo, whose previous work, advise and ideas have been fundamentals to this project.

Secondly, I want to thank all the new friends that I've met during this whole master year. It's always nice to meet really different people but with the same interests than you from who you can learn a lot of things.

Last but not least, I am very grateful to my family for their unconditional support and encouragement over these years. Many thanks.

Abstract

Computational modelling of expressive music performance has been widely studied in the past. While previous work in this area has been mainly focused on classical piano music, there has been very little work on guitar music, and such work has focused on monophonic guitar playing. In this work, we present a machine learning approach to automatically generate expressive performances from non expressive music scores for polyphonic guitar. We treated guitar as an hexaphonic instrument, obtaining a polyphonic transcription of performed musical pieces. Features were extracted from the scores and performance actions were calculated from the deviations of the score and the performance. Machine learning techniques were used to train computational models to predict the aforementioned performance actions. Qualitative and quantitative evaluations of the models and the predicted pieces were performed.

Resum

El modelatge computacional de interpretacions expressives de peces músicals ha estat àmpliament estudiat en el passat. Tot i que el treball previ en aquesta àrea s'ha centrat principalment en la música clàssica per piano, hi ha hagut molt poc treball sobre música per guitarra i aquest s'ha centrat en la guitarra monofònica. En aquest treball, utilitzem aprenentatge automàtic per generar automàticament interpretacions expressives a partir de partitures de música no expressiva per a guitarra polifònica. Tractem la guitarra com a un instrument hexafònic, obtenint una transcripció polifònica de les peces musicals interpretades. A partir de les partitures s'han extret diverses característiques i s'han calculat accions interpretatives a partir de les desviacions entre la partitura i la interpretació del músic. Diverses tècniques d'aprenentatge automàtic s'han utilitzat per entrenar models computacionals i predir les accions interpretatives esmentades anteriorment. Finamlent s'han realitzat avaluacions qualitatives i quantitatives dels models i les peces predites.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Music is a very important part of the life of most people. Depending on our mood or moment in the day or lives, music can be understood in very different ways. In some moments we understand music as a simple distraction, a soundtrack to our daily tasks without paying much attention to it. In other moments, when we consciously listen to music, we can be very touched and excited by it. This engaging part of music is largely due to the human component added to the performance. Instead of reading a score, musicians play the music on their own way, by changing (unconsciously or consciously if the performer wants to achieve a specific goal) a lot of "parameters" of it such as intensity, velocity, volume or articulation of each note. Moreover, people can clearly distinguish the manipulation of sound properties done by different performers and create preferences based on these differences.

The study of music expressive performance, from a computational point of view, consists of characterizing the deviations that a musician, when performing a musical piece, introduces in the performance. In this work we are going to focus on modelling guitar scores and performances. In Section 1.1 we briefly explain how this thesis is organised.

Computational modelling of expressive music performance has been widely studied in the past. While previous work in this area has been mainly focused on classical piano music, there has been very little studies on guitar music, and such work has

focused on monophonic guitar playing. One of the main challenges of focusing this study to guitar is the polyphonic nature of the guitar. The complexity of polyphonic sound transcription is well known, so to solve this issue, the use of a hexaphonic guitar is chosen, in which each of the strings is processed as an independent monophonic sound source, simplifying the transcription of the sounds.

In this thesis, we present a machine learning approach to automatically generate expressive performances from non expressive music scores for polyphonic guitar. We treated guitar as an hexaphonic instrument, by transcribing each string separately we were able to obtain a polyphonic transcription of performed musical pieces. Features were extracted from the scores and Performance Actions were computed from the deviations of the score and the performance. Machine learning techniques were used to train computational models in order to predict the aforementioned Performance Actions. Qualitative and quantitative evaluations of the models and the predicted pieces were performed.

1.1 Structure of the report

In this chapter we discuss the motivation of this master's thesis, the main objectives and we explain briefly the structure of this report. The rest of this thesis is organised as follows: in Chapter 2, we present some related work on expressive music performance modelling specially focused on polyphonic music. In Chapter 3, the tools and resources (hardware, software and data) used in this work are described. In Chapter 4, we present the proposed methodology. In Chapter 5, the evaluation measures and results are presented. We conclude with a brief conclusion and provide suggestions for future improvements in Chapter 6.

In order to complement this thesis we present 3 Appendices: Appendix A providing links to all On-line resources from this thesis, Appendix B documenting the dataset used for this work, and Appendix C gathering all responses to the On-line Survey.

1.2 Motivation

While most studies about guitar modelling are focused on monophonic performances, the aim of this master thesis is to investigate the modelling of expressiveness in polyphonic (hexaphonic) guitar music. We will base our approach on previous studies by Giraldo [1] who computed expressive performance models for monophonic jazz guitar. Treating guitar as a monophonic instrument limits hardly the polyphonic nature of the instrument, but avoids the problems related to polyphonic music transcription. The main objective of this thesis will be to define a set of features extending previous work on monophonic guitar performances to polyphonic performances. Those features aim to represent the different nuances in time, duration or volume that the guitarist introduces when performing a musical piece, appearing both in the temporal or *horizontal* axis (as a monophonic melody), but also should represent the *vertical* axis representing the simultaneity between notes. The features should represent the variations in time and energy that the performer introduces, and those will depend on the context of the note if it is part of the melody or the harmonic accompaniment.

I personally believe that this concept of music expression plays a major role in how we appreciate musical experiences. A musical piece does not sound the same (or we do not feel it the same way) played by two different players. Even the same piece does not sound the same when played twice by the same player.

Having a knowledge about the exact Performance Actions that a guitar player performs when reading and performing a musical piece could help us in many directions. By just replicating these nuances we would theoretically be able to sound like a concrete guitar player. Understanding little nuances that expert players perform could also help less-trained musicians to improve their playing. These models could also be implemented in music annotation software, in order to generate expressive performances from user-composed scores. That way, the plain score to midi conversion could be substituted by an expressive playback, much closer to the performance of an expert guitarist, and thus improve the overall user experience.

1.3 Thesis statement

In this section we present the main hypothesis of this master's thesis:

It is possible to computationally capture and model the expressive nuances that a musician introduces when performing a musical piece, taking the polyphonic guitar as a study case.

1.4 Objectives

The aim of this work is to study and predict computationally predict the little nuances or *Performance Actions* that musicians do when performing a musical score, focusing on time (*Onset Deviation*) and amplitude (*Energy ratio*) deviations, taking the polyphonic (hexaphonic) guitar as a study case. This study will consider, as explained in section 1.2, both horizontal or melodic axis and vertical or harmonic axis.

The specific objectives are as follows:

- To create a database of hexaphonic recordings played by a guitarist and their corresponding scores.
- To automatically transcribe the audio of the hexaphonic recordings into a machine-readable format (MIDI).
- To adapt existing code libraries to extract descriptors from the score which allow us to characterise the notes vertically and horizontally.
- To create code libraries which allow us to align and compare the transcribed hexaphonic recordings to the score in order to extract performance actions.
- To provide some examples of polyphonic performance to score alignments.
- To generate different models that try to predict performance actions (onset deviation and energy ration) by using Machine Learning techniques.
- To analyse which descriptors influence more the accuracy of these models, so to say, which descriptors represent more the behaviour of the musician.

- To obtain not only quantitative machine learning results but also qualitative results by surveying different users.

Chapter 2

State of the art

In this section we will review the state of the art in music expression, giving an overview of the past and present research in the field. We specifically focus on polyphonic music expression modelling where machine learning has been used to predict some kind of performance actions.

The state of the art of this work can be divided in two parts: firstly, in section 2.1, we review the works related to music expressive performances modelling. Secondly, in section 2.2 we provide examples of works that try to automatically transcribe guitar, focusing on those treating guitar as an hexaphonic instrument transcribing each string separately.

2.1 Music expression modelling

Music Expression is defined as the manipulation a performer does in duration (enlarge shorten notes), onset (delay or anticipate notes), energy (play notes louder or softer) and embellishment (add or substract notes).

Performance Actions (PAs) can be defined as musical resources used by musicians to add expression when performing a musical piece, which consist of little nuances (variations in timing, pitch, and energy) that are not indicated in a score. In the same context, ornamentation can be considered as an expressive musical resource

used to embellish and add expression to a melody. These PAs are what make music expressive and differentiate it from a robotic performance, these little nuances, done mostly unconsciously, are part of our human nature and it's what makes us feel and enjoy a musical performance as something unique. This uniqueness of a performance based on the variation in timing, dynamics, timbre and pitch was first proposed by Juslin [2]. Ramírez and Hazan [3] add the gradation that those little variations should be clearly distinguishable for listeners.

In the past, music expression has been mostly studied in the context of classical music and most research focuses on studying timing deviations (onset nuances), dynamics (energy) and vibrato (pitch nuances). Some studies try to obtain rules to represent that performance actions by hand from music experts. There are several expert-based systems studying this field from different perspectives. The KTH group developed in 2009 a set of several rules [4] for predicting tempo, energy and pitch variations included in a system called *Director Musices*. Parts of the rule system were implemented in other programs (see for instance the work by Sundberg [5] that tries to use rules to predict *Inter Onset intervals* or Bresin [6] who try to generate macro rules for predicting PAs).

In Table 2.1 a brief summary of this three systems is displayed. This is not meant to be an extensive overview of non-machine learning systems as our approach will be purely computational. However, we found interesting to just mention briefly a few of this works on trying to understand music expression from a theoretical and rule-based point of view.

Author	System	Instrument
KTH [4]	Director Musices	General
Sundberg [5]	Inter-Onset	Piano
Bresin [7] [6]	DM mapped to emotions	General

Table 2.1: State of the art few expert-based methods table review.

On the other hand, machine-learning-based systems try to obtain the set of rules (expressive models) directly from the music performance by trying to directly measure the PAs applied by the performer. This PAs are computed by measuring deviations

of the expressive performance (done by a professional performer) with respect to a neutral or robotic data (such as strict MIDI representations of the score). For an overview of theses methods see the review by Goebl [8], from where we can see that most of the proposed expressive music systems are in classical music, and most of these systems are based on piano performances. Those kinds of machine-learning systems started arising when simple synthesiser keyboards or digital pianos were used to capture expressive performances. Those devices allowed accurate timing and loudness data to be sent via MIDI (Musical Instrument Digital Interface) to a computer.

In order to obtain these expressive performance models, several types of machine learning algorithms have been used, Bresin [7] tries to model piano performances using Artificial Neural Networks (ANN) by trying to learn automatically the Director musices rules stated by KTH group. Camurri [9] also applied ANN in order to obtain music expression for flute performances. He also developed a 2D representation of the expression space using non-linear projections in order to be able to choose between different emotions and their middle points.

Widmer [10] (also in [11]) used rule-based learning and meta-learning algorithms in order to cluster piano performances. He developed a new rule discovery algorithm named PLCG (Partition+Learn+Cluster+Generalise) that can find simple, robust partial rules models (sets of classification rules) in complex data where it is difficult or impossible to find models that completely account for all the data. PLCG is an ensemble learning method that learns multiple models via some standard rule learning algorithm, and then combines these into one final rule set via clustering, generalization, and heuristic rule selection. He also uses this algorithm and discovered rules to predict multi-level timing and dynamics [11].

Grindlay [12] utilises Hidden Markov Models in order to extract Performance Actions from performances from both students pianists and professional pianists in order to model different performances. He uses HMM in order to predict time variations from a non-expressive score. In his work Miranda [13] uses a generative performance system based on genetic algorithms in order to predict those time variations.

Contrary to classical music scores, performance annotations (e.g. ornaments, dynamics and articulations) are seldom indicated in popular music scores, and it is up to the performer to include them based on their musical background. Therefore, in popular music it may not always be possible to characterise ornaments with the archetypal classical music conventions (e.g. trills and *appoggiaturas*).

Several approaches have been proposed to generate expressive performances in non-piano-classical music. Arcos [14] proposed a system that generates jazz solo saxophone expressive performances, based on case-based reasoning. In his work, several recordings of a tenor sax playing different Jazz ballads were made. These recordings were analysed to extract information related to several expressive parameters. This set of parameters and the scores constitute the set of cases of a case-based system. From this set of cases, the system infers a set of possible expressive transformations for a given new phrase applying similarity criteria, based on background musical knowledge, between this new phrase and the set of cases.

Gratchen [15] also applies case-based reasoning to generate models for ornamentation and tempo variations for jazz saxophone music. His system automatically performs melodic and expressive analysis, and when a new musical performance must be tempo-transformed, it uses the most similar example tempo-transformation to infer the changes of expressiveness that are necessary to make the result sound natural.

Ramírez [3] generates a tool in order to both generate and explain expressive music performances of monophonic Jazz melodies for saxophone. The tool consists of three components a melodic transcription component which extracts a set of acoustic features from monophonic recordings, a machine learning component which induce both an expressive transformation model and a set of expressive performance rules from the extracted acoustic features, and a melody synthesis component which generates expressive monophonic output (MIDI or audio) from inexpressive melody descriptions using the induced expressive transformation model.

Puiggros [16] tries to generate automatic characterization of ornamentation from bassoon recordings in order to generate expressive synthesis. His work addresses

the characterization of expressive bassoon ornaments by analysing audio recordings played by a professional bassoonist. This characterization is then used to generate expressive ornaments from symbolic representations by means of Machine Learning

Previous work on guitar expressive performance modelling has mainly been done by Sergio Giraldo [1] who uses machine learning techniques to model ornamentation and PAs in monophonic jazz guitar performances according to the characteristics of the notes' context. Features extracted from scores and their corresponding audio recordings performed by a professional guitarist are used to train computational models for predicting melody ornamentation. Several machine learning techniques were explored to induce regression models for timing, onset, and dynamics (i.e. note duration and energy) transformations, and an ornamentation model for classifying notes as ornamented or non-ornamented.

Bantula [17] models expressive performance for a jazz ensemble of guitar and piano. The aim of her project is to study the influence of piano accompaniment into the performance of a guitar melody and vice versa. Based on a set of real performances, she extracts information from both score and recordings and using machine learning techniques she trains models for both piano and guitar performances. From our point of view, the interesting part of this work is the polyphonic treatment done to the piano, extracting features for chords played such as *density, weight* or *range*. Kirke et al [18] models polyphonic piano recordings with generative experiments that show that multiple polyphonic expressive actions can be found in human expressive performances.

In Table 2.2 we can see an overview of authors, methods and instruments where music expression modelling using machine learning was applied.

2.2 Automatic hexaphonic guitar transcription

When we think about music transcription we usually think about a music expert listening repeatedly a musical piece and writing it down to a traditional score notation. Defined by Klapuri [19] music transcription is *"the process of analysing an acoustic*

Author	Method	Instrument	Mono/Poly
Arcos [14]	Case based reasoning	Saxophone	monophonic
Bantula [17]	Several methods	Jazz ensemble	polyphonic
Bresin [6] [7]	*ANN*	Piano	monophonic
Camurri [9]	*ANN*	Flute	monophonic
Giraldo [1]	Several methods	Guitar	monophonic
Gratchen [15]	Case based reasoning	Saxophone	monophonic
Grindlay [12]	*HMM*	Piano	monophonic
Kirke [18]	Generative models	Piano	polyphonic
Miranda [13]	Genetic Algorithms	Piano	monophonic
Puiggros [16]	Several methods	Bassoon	monophonic
Ramirez [3]	Several methods	Saxophone	monophonic
Widmer [11] [10]	Rule-based meta-learning	Piano	monophonic

Table 2.2: State of the art machine learning methods table review. Non exhaustive table.

musical signal so as to write down the musical parameters of the sounds that occur in it". Thus, the traditional main goal of music transcription is to represent music as detailed as possible, so it can be accurately reproduced afterwards.

Nowadays, we also think about music transcription as the way to convert acoustic music signal to a machine readable format, such as MIDI, XML or piano-roll representation in order to be analysed and processed produce a notation reflecting the most relevant information about the musical events within it, as an output.

One of the main difficulties when facing automatic music transcription is given by the number of voices a musical signal has, or the number of sound sources that are present in it. The more sound sources in the same sound signal, the more difficult the transcription becomes. Limiting the problem to one single source (as it is in our case) makes us confront with another problem: monophonic and polyphonic source. A monophonic source produces only one note at a time, while a polyphonic source can play multiple notes simultaneously.

The monophonic transcription case is considered as solved by state of the art techniques (Klapuri 2004 [19]). However, polyphonic case is really far from being solved specially for multi-instrumental contexts. The main problem of polyphonic transcription is multiple fundamental frequency estimation and tacking, which is a very

difficult task when two or more concurrent sounds contain partials that share some frequencies. Knowing in advance the different sources eases a bit the task [20].

There are some works with good results in multiple fundamental frequency detection by Klapuri [21] who tries to estimate multiple fundamental frequencies calculating the salience, or strength, of a F0 candidate as a weighted sum of the amplitudes of its harmonic partials. This F0 salience spectrum is found by optimization using generated training material. Benetos [22] proposes an efficient, general-purpose model for multiple instrument polyphonic music transcription. His model is based on probabilistic latent component analysis and supports the use of sound state spectral templates, which represent the temporal evolution of each note (e.g. attack, sustain, decay).

In our case, working with guitar music, transcription process is a difficult task due to the polyphonic nature of the sound it emits. This polyphony is caused by the different strings of the guitar played together, which leads to several notes sounding at the same time (chords). As said before, limiting the player to just produce a monophonic melody eases the process as there exist very good and state of the art approaches for monophonic music transcription such as the autocorrelation method, Yin or spectral peak picking, among others.

However, the main goal of this project is to extend a monophonic system to a polyphonic one, so polyphonic transcription is one of the main tasks of it. Reviewing the literature we find a few approaches for transcribing polyphonic guitar music. Fiss & Kwasinksi propose a system for automatic guitar audio transcription in real time [23]. This approach is based on the STFT (Short Time Fourier Transform) to compute the spectrogram used to extract information about peak locations. Afterwards they try to correct the note detector by taking into account the probability of each note being produced among the six strings of the guitar, and thus, avoiding the ambiguity of polyphonic guitar.

As stated above, avoiding polyphony makes the transcription task much easier. So, the idea of capturing and transcribing each string separately makes the task at

hands easier. Thus, in order to be able to record each string separately we could ask the musician to think about the whole song but just play the notes on one single string, and make him repeat the process six times changing the selected string. That approach would probably be very difficult for the musician as well as non musical at all.

Solving the hexaphonic recording problem, O'Grady & Rickard [24] proposed a solution based on the Roland GK-3 divided pick-up [25] which captures separately each string in order to be processed with a guitar synthesiser and create really strange and creative sounds with it. For the transcription of these six different signals they used Non-Negative Matrix Factorization (NMF) where a matrix V is factorised into two matrices W and H, with the property that all three matrices have no negative elements. In this case, for one string W_{string} containing the magnitude spectrum of all possible notes played on that string, the resulting H_{string} is an activation matrix indicating the position in time in which each note was played.

In the following sections, the steps followed towards the implementation of the project are described, by firstly analysing the tools and resources that were used (Section 3), and then explaining the methodology that we followed and all its little steps (Section 4).

Chapter 3

Materials

For this project, we have used several materials which can be divided into 3 categories: hardware, software and data.

Hardware

- Roland GK-3: the hexaphonic recordings were done using this special divided pick-up
- Breakout Box [24]: this adaptor box was needed in order to convert the output from the GK-3 to 6 standard Jack connectors.
- PC: Intel Core i5-6600 CPU @ 3.30GHz, 16.0GB RAM

Software

- ProTools HD 10: it was used in order to generate a mix the 6 strings channels. It also was used to synthesise midi both from the transcribed performances and from the predicted performance.
- MuseScore 2: the scores in XML format for the performances were written using MuseScore.
- Python: the code for transcribing guitar performances was developed by using Python.

- Essentia [26]: we used a few algorithms (in order to transcribe guitar) from this open-source C++ library for audio analysis and audio-based music information retrieval.
- Matlab: the code for extracting the features from the performance and from the scores was developed using Matlab.
- MIDI Toolbox [27]: the MidiToolBox Library (implemented in Matlab) allowed us to process easily MIDI data.
- Weka Data Mining Software [28]: it was used to train and test different machine learning models, to implement feature selection and to analyze the results.
- BealeJS [29] (browser based evaluation of audio quality and comparative listening environment) which provides a framework to create browser based listening tests and is purely based on open web standards like HTML5 and Javascript.

Data

For this work we used a set of three recordings done by Helena Bantula for her Master's thesis [17] consisting of one recording of *Darn that dream* a Jazz standard by Jimmy Van Heusen and Eddie De.Lange and two recordings of *Suite en la* a classical piece by Manuel M. Ponce. Their corresponding scores where written using MuseScore and extracted as XML files.

Please see Appendix B for a full review of the dataset, where to find it, how is it structured and which files are used.

Chapter 4

Methodology

In Figure 4.1 we present a block diagram of the whole system from where we can see that four separate stages of this thesis can be defined: data acquisition (guitar recording), transcription, feature extraction and models computation. Expressive hexaphonic guitar recordings were done using the Roland GK-3 divided pick-up, which is able to separate sound from each string [30]. The main output of this first stage is a new dataset consisting of hexaphonic recordings recorded by a guitar player with different performance actions of the performance.

After this step, transcription of each individual string is computed. After doing a score alignment with the original score and the transcription of the expressive guitar performance, feature extraction needs to be done.

Feature extraction is performed following an approach in which each note is characterised by its *nominal*, *neighbouring*, and *contextual* properties. Here is where the most of the research in this thesis takes place: checking for literature in expressive piano modelling, combining it with previously mentioned features of monophonic expressive guitar modelling,... Afterwards, several machine learning and feature selection algorithms are applied to predict those performance actions (timing, pitch, energy,...) and ornaments introduced by the musician when performing a musical piece.

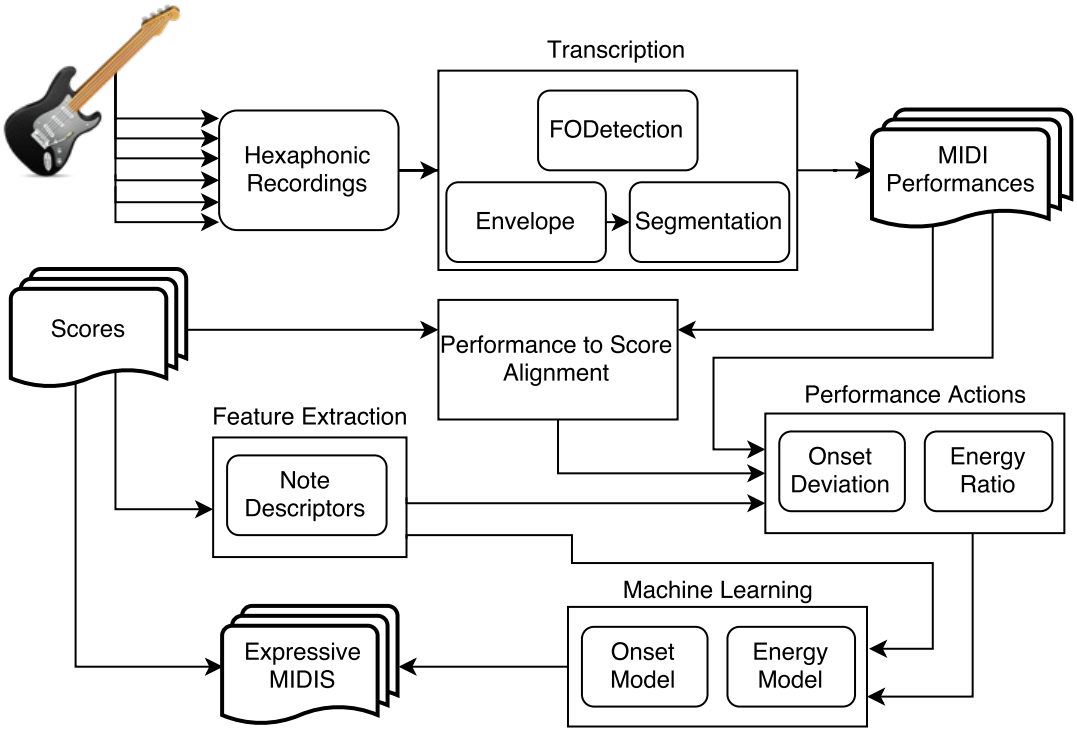

Figure 4.1: Block diagram of the whole system.

4.1 Data acquisition

In order to obtain hexaphonic recordings and get each string nicely separated we used the Roland GK-3 divided pick-up that is easily attached to any steel-stringed electric guitar and acts as a sound transducer device. It is able to separate very good the sound from each string and delivers accurate performance data.

However, the output of this pick-up consists of a 13 pin DIN cable that allows to connect the guitar to guitar synthesisers such as Roland's popular GR-55 and at the same time to fed electrically the pick-up. So, in order to be able to record each string separately we need to adapt the pick-up output so the sound of each string can be inputted to the computer through an independent input channel of an audio interface.

To do this, a Breakout Box circuit was built by I.Angulo [30] for his master's thesis last year based on the specifications by O'Grady [24], so we reused it. As we can see in Figure 4.2, the final box has an input for the 13 pin DIN cable and 6 separate

Jack connector cables are outputted, one for each string. Also, two batteries are needed inside the box in order to fed the pick-up.

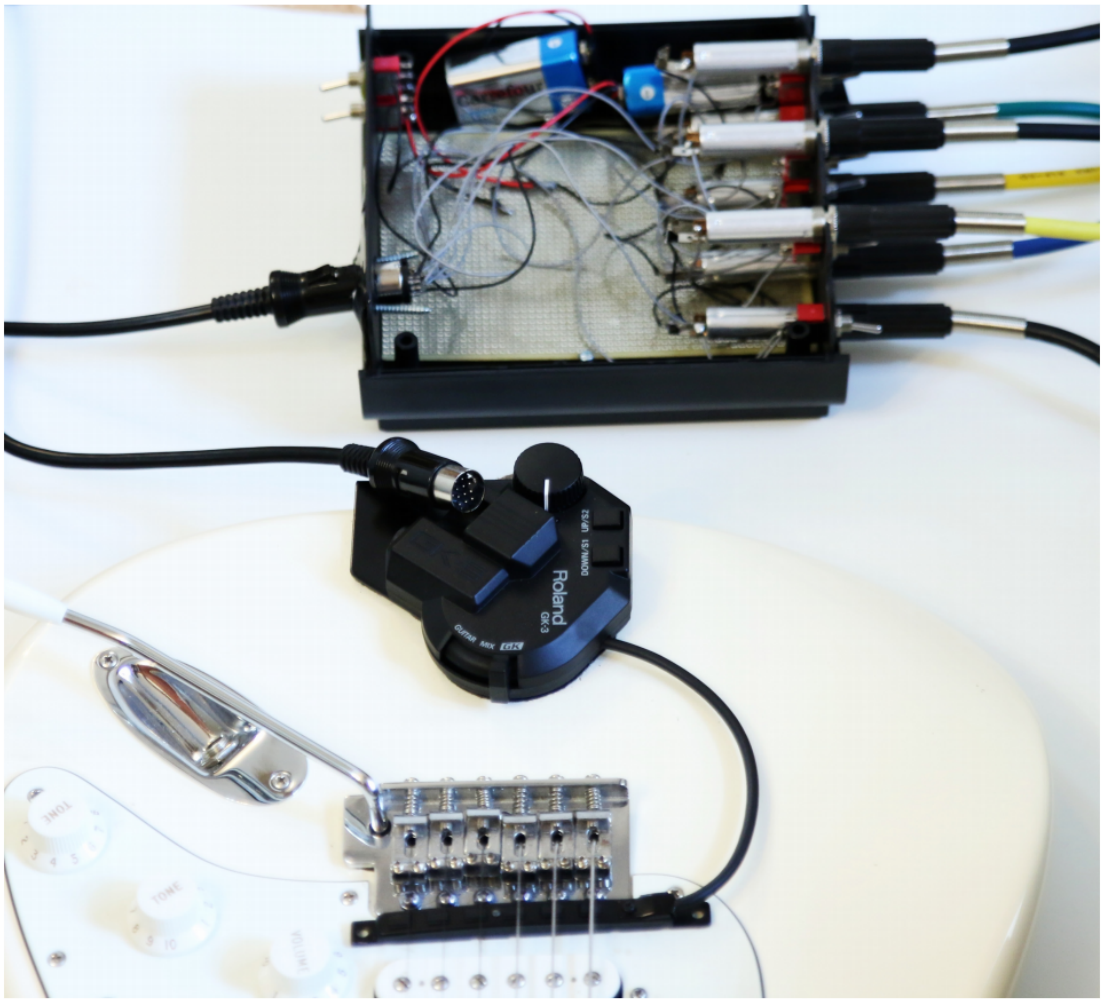

Figure 4.2: Roland GK-3 and Breakout Box setting. (Angulo, 2016)

In this study, and as explained in chapter 3, the final dataset consists of 3 audio recordings (one recording of *Darn that dream* and two recordings of *Suite en La*) resulting in a total of 1414 notes recorded by an amateur guitarist and their corresponding music scores saved as xml files using Muse Score 2. In Figure 4.3 we can see a few bars from *Darn that dream* score with annotated chords. This score is saved as an xml file in order to be able to characterise each note by a set of descriptors as explained in the following sections.

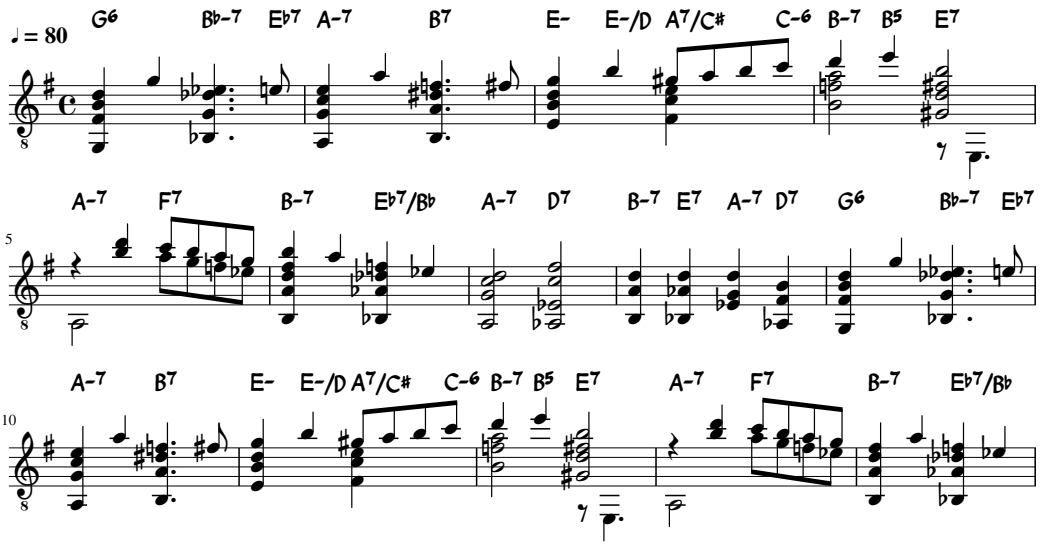

Figure 4.3: *Darn that dream* first 14 bars with annotated chords.

4.2 Hexaphonic guitar transcription

Once we have the six audio input signals, all the processing involved in the transcription is done using Python and Essentia [26] algorithms. A Python script has been created in which all the steps of the transcription are included, giving the user the option to configure every parameter of it.

The aim of this section is to obtain machine readable (and understandable) representation from the audio recordings in order to be able to compute descriptors and performance actions. In order to obtain a note representation based on pitch, onset, duration and offset for each note, the audio signal from each string from the guitar is automatically transcribed into a MIDI format. Each signal is independently processed, following the hexaphonic concept in which each string is considered as a monophonic sound source.

This step is based on the previous work of Bantula, Giraldo and Ramírez [17]. However, the algorithm has been modified a bit in order to correctly transcribe the hexaphonic audio (that can contain leakage from the other strings) instead of simple monophonic ones.

For doing this, we first need a fundamental frequency detector in order to obtain the

pitch of each string. The original algorithm used the YIN algorithm [31], however, in order to improve the system we changed it to Melodia which offers more robustness against the leakage from other strings and helps to detect the main pitch present in the signal. In order to provide a better F0 detection, we also tuned the parameters of this algorithm. The minimum and maximum frequency (the range) of the detector is set differently to each string according to the frequencies that each guitar string can produce. As can be seen in Table 4.1 we considered from each string a range of one octave because in this particular recordings and scores the guitarist never passes the twelfth fret (first octave).

String	Frequency	Pitch	MIDI	Range
1 (E')	329.63 Hz	E4	64	[320-660]
2 (B)	246.94 Hz	B3	59	[240-500]
3 (G)	196.00 Hz	G3	55	[190-400]
4 (D)	146.83 Hz	D3	50	[140-300]
5 (A)	110.00 Hz	A2	45	[100-230]
6 (E)	82.41 Hz	E2	40	[80-170]

Table 4.1: Guitar Strings Frequencies. Frequency refers to the base frequency of the string. MIDI refers to MIDI note number and Range to the Frequency range in Hz considered for transcription.

The power envelope of the signal is used to apply an adaptive noise gate in order to filter out none pitched sounds. That power envelope is also used in order to compute MIDI velocity (energy is linearly mapped to a value from 0 to 127). Finally, the filtered frequency pitch profile obtained is rounded to a MIDI note number.

In Figure 4.4 we can see the four stages of the process: we start from the audio (a), we obtain a pitch profile using Melodia (b) and using an adaptive threshold over the audio wave envelope (c) we obtain a filtered pitch profile.

Following the algorithm proposed by Bantula, a rule based filter is applied in order to remove short notes and gaps merging them with corresponding neighbour notes based on a cognitive perspective of the perception of time.

Following previous step, onsets and offsets are detected from differentiating the cleaned pitch profile. This means for a pitch remove next one, so the changes in pitch

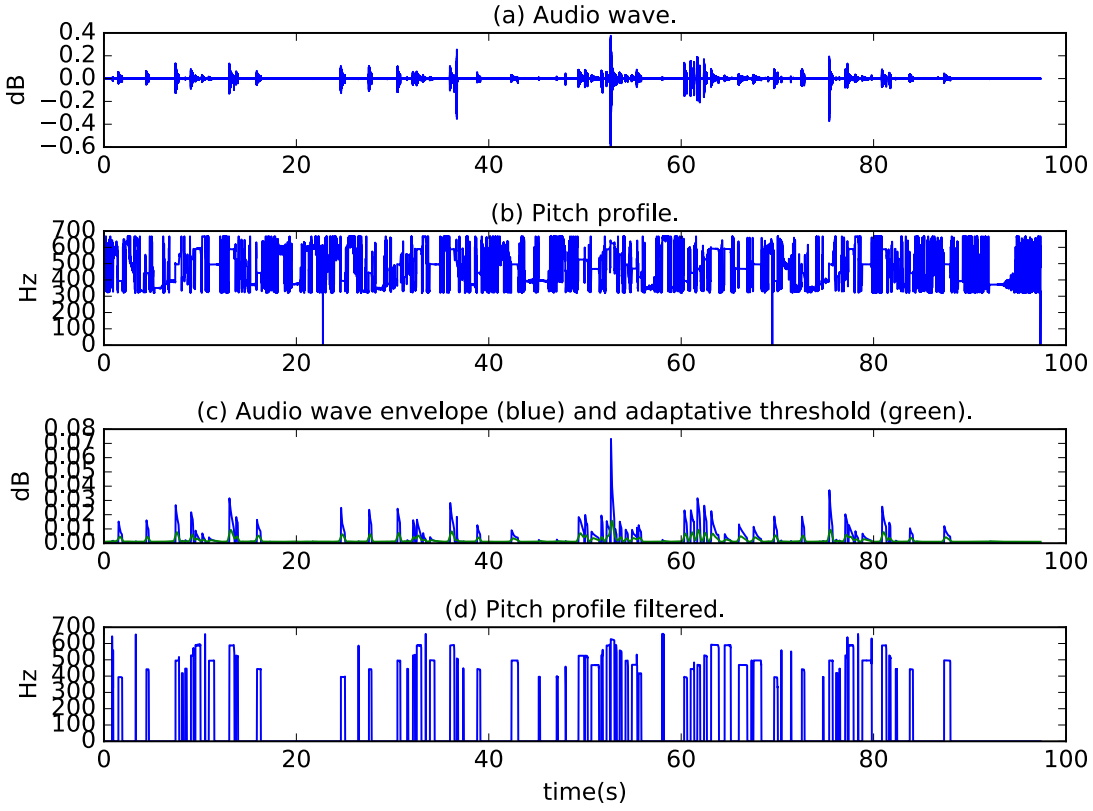

Figure 4.4: One string automatic transcription

become positive or negative peaks. Positive ones (above a threshold) are labelled as onsets and negative ones (below a threshold) are labelled as offsets. Duration is computed by subtracting the offset to the onset.

After all this process, a few manual corrections were performed by changing pitch, eliminating notes or time stretching the performance in order to have a better alignment with the score. Afterwards, all six transcriptions (one from each string) were merged in order to obtain a single MIDI file from the performance. MIDI channel was used in order to label the notes according to the string where were played. In Figure 4.5 we can see a piano-roll representation of a transcribed hexaphonic performance, showing each string in a different colour.

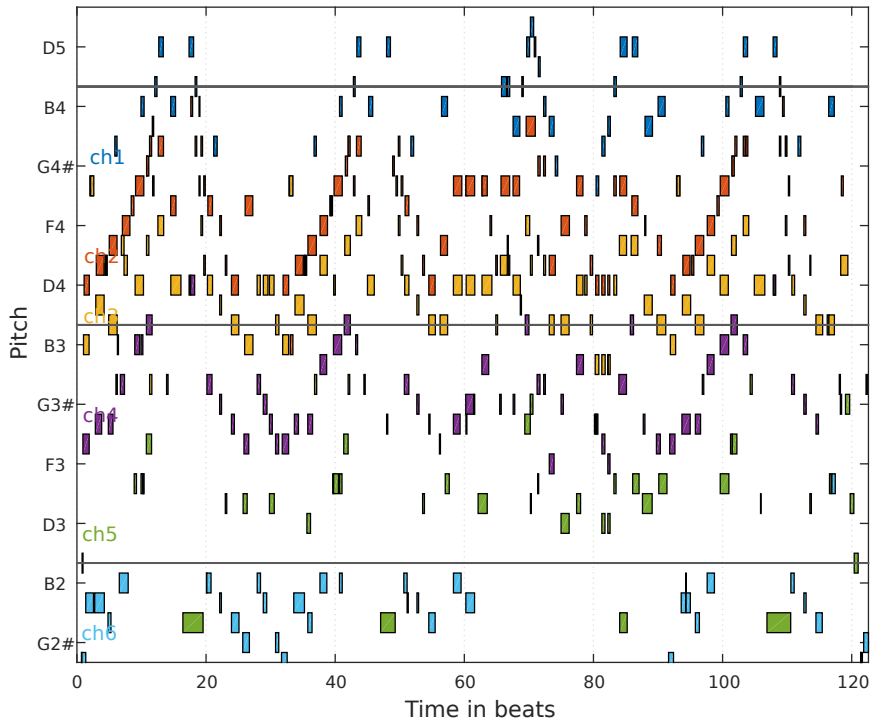

Figure 4.5: Piano-roll representation of an hexaphonic performance.

4.3 Feature extraction

In this section we describe how we analyse the music scores in order to extract multiple descriptors for each note. Afterwards, performance to score alignment is computed and performance actions (*Onset Deviation* and *Energy ratio*) are extracted.

4.3.1 Note Descriptors

Feature extraction from the music scores is performed following an approach similar to that of Giraldo [1] but extended and computationally adapted to polyphonic scores, in which each note is characterised by its *nominal*, *neighbouring* and *contextual* properties, taking into account both horizontal (time or melodic) and vertical (simultaneous or harmonic) axis. The complete list of the descriptors extracted from the music scores can be found in Table 4.2.

- **Nominal**: This descriptors refer to the intrinsic or intra-note properties of

Code	Descriptor	Abbreviation	Units	Formula	Range
7	Duration	ds_n	Seconds	ds_0	[0,+∞]
2	Duration	db_n	Beats	db_0	[0,+∞]
6	Onset	ons_n	Seconds	ons_0	[0,+∞]
1	Onset	onb_n	Beats	onb_0	[0,+∞]
15	Onset in bar	obm_n	Beats	$ob_0 \% bpb$	[0,+bpb]
4	Pitch	p_n	Semitones	p_0	[1,127]
16	Chroma	ch_n	Semitones	$p_0 \% 12$	[0,11]
5	Energy	v_n	MIDI vel	v_0	[1,127]
3	String	str_n	String num	$channel_0$	[1,6]
10	Prev. duration	pds_n	Seconds	ds_{-1}	[0,+∞]
9	Prev. duration	pdb_n	Beats	db_{-1}	[0,+∞]
12	Next duration	nds_n	Seconds	ds_1	[0,+∞]
11	Next duration	ndb_n	Beats	db_1	[0,+∞]
18	Prev. interval	$pint_n$	Semitones	$p_{-1} - p_0$	[-60,60]
19	Next interval	$nint_n$	Semitones	$p_1 - p_0$	[-60,60]
13	Prev. inter-onset dist.	$piod_n$	Seconds	$os_0 - os_{-1}$	[0,+∞]
14	Next. inter-onset dist.	$piod_n$	Seconds	$os_1 - os_0$	[0,+∞]
28	Narmour	$nar1_n$	Label	$nar(p_{-1}, p_0, p_1)$	[P, D, R, ID]
29		$nar2_n$		$nar(p_{-2}, p_{-1}, p_0)$	[VR, IR, VP, IP]
30		$nar3_n$		$nar(p_0, p_1, p_2)$	[dyadic, monadic, none]
33	Is a Chord	ich_n	Boolean	$isChord_0$	{true, false}
34	Is a Pedal	pdl_n	Boolean	pdl_0	{true, false}
17	Simultaneous notes	sim_n	Number	$simult_0$	[0,+∞]
8	Measure	m_n	Bars	m_0	[0,+∞]
31	Tempo	t_n	Bpm	t_0	[30,260]
20	Key	k_n	Semitones	k_0	[-6,6]
35	Mode	mod_n	Label	mod_0	{major, minor}
23	Chord root	chr_n	Semitones	chr_0	[0,11]
24	Chord type	cht_n	Label	cht_0	{+, 6, 7, 7#11, 7#5, 7#9, 7alt, 7♭5, 7♭9, Maj7, dim, dim7, m, m6, m7, m7♭5, major}
21	Note to key	$n2k_n$	Semitones	$ch_0 - k_0$	[0,11]
25	Note to chord	$n2ch_n$	Semitones	$ch_0 - chr_0$	[0,11]
26	Is chord note	$ichn_n$	Boolean	$isChNote$	{true, false}
27	Metrical strength	mtr_n	Label	$metStr_0$	{Very strong, Strong, Weak, Very weak}
32	Phrase	ph_n	Label	$phrase_0$	{initial, middle, final}

Table 4.2: Complete list of descriptors extracted from music scores.

score notes. So to say, this descriptors are needed in term to define completely a note. *Duration* (computed from offset and onsets) and *Onset* are given both in seconds and beats, as the descriptor in seconds depends on the tempo of the piece. *Onset in bar* refers to the position of the onset related to the beats per bar measure. If its 1 the first beat on the bar, 2 second beat on the bar, and so on. *Pitch* is directly the MIDI note number, and *Chroma* is the pitch modulus 12, so to say, the pitch class of the note, the pitch without taking into account octave changes. *Energy* descriptor gives us how loud is the note played (loudness), and its directly taken from MIDI velocity (how fast the note is played). Finally, *String* number is also extracted for each note directly from the MIDI channel.

This descriptors need no computation as they are intrinsic properties of each note.

- **Neighbouring**: neighbouring or inter-note descriptors refer to the relations of the note with its neighbouring or simultaneous notes. Each note is characterised by *Previous duration* and *Next duration* given both in seconds and beats by subtracting previous and next note duration to current one. Also *Previous interval* and *Next interval* is the difference between in pitch between the current note and the next or previous one given in semitones. *Inter-onset distance* refers to the onset difference between current and previous or next note. *Simultaneous notes* counts the number of simultaneous notes to the current note within a given threshold.

  The computation of Previous and Next note properties is a bit tricky when talking about polyphonic scores. In order to clarify this problem, we will explain it with an example as can be seen in Figure 4.6 where we focus in the first bar of *Darn that dream*. From a computational point of view (or from a digital score parser) if we focus on the low G note in the first chord, the next note would be the F♯ note in the same chord. However, from a musical (or harmonic) point of view, after that low G it comes the low B♭ in the 3rd beat chord. The same happens for the high G note in the second beat. Their next

notes can be considered either the low B♭ (parsing the score) or the high E♭ (melodic continuation).

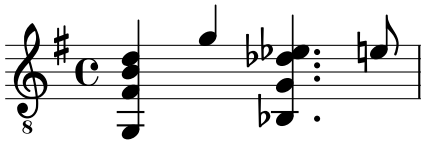

Figure 4.6: *Darn that dream* first bar.

In our case, we decided to opt for the straight digital score parsing. When reading and playing cords with a guitar they are not read as singles notes but as a group, and they are usually played from lower to higher pitch in an *arpeggiated* way. Following this methodology, in the previous example (Figure 4.6, after the low G in the first chord it comes the F♯ note on top of it with an *Inter-onset distance* of 0 seconds, then B, D, G, B♭, G, and so on. So, parsing chords from bottom to top instead of searching for melodic continuation, in addition to make easier the computation, it helps the system understand chords as a group of ordered notes with an *Inter-onset distance* of 0 seconds.

Following the work of Bantula [17] on piano polyphonic music, notes have been also labelled as *Chord notes* or *Pedal notes* depending on the simultaneity of other notes, differentiating between notes that have been played at the same time as a chord and notes that are played as a basis in order to have a melody in top of them.

In this neighbouring category of descriptors, categorization based on the implication-realization (I-R) model of *Narmour* [32] has been also computed. This model parses melodies and obtains for each note a label depending on the previous and next notes. This computation has also been adapted in order to take into account polyphonic melodies as explained in Figure 4.6.

- **Contextual**: This descriptors refer to the context, background or properties of the song in which the note appears on. Some of this descriptors, such as MEASURE, *Tempo*, *Key* or *Mode*, are the same for the whole song but may be

useful if we merge different datasets (songs) into one. *Chord root* and *Chord type* refer to the actual chord of that note (labelled manually per bar or per measure in the score).

The rest of this descriptors are just computations of each note respecting to those first descriptors. *Note to key* and *Note to chord* refer to the distance in semitones of the actual note both to the general key root of the piece and to the actual chord root. *Is chord note* gives us a boolean label depending if the note is part of those notes pre-defined by the chord root and type as can bee seen in Table 4.3. *Metrical strength* categorise notes occurring at strong or weak beats within a bar. And *Phrase* descriptor labels notes depending on the melodic segmentation approach by Cambouropulos [33] into initial, middle and final notes.

4.3.2 Performance to score alignment

In this stage, and in order to compute performance actions in the next step, we need to know which notes on the performance correspond to which notes in the score, or mostly known as performance to score alignment.

This is done automatically and in this case we used Dynamic Time Warping (DTW) techniques in order to match performance notes to score. Those notes are aligned depending on a cost function based on onset, pitch and duration deviations. All these deviations can be weighted with a parameter in order to penalise more errors in pitch than in onset i.e. Firstly, and before applying DTW, performances have been manually time-stretched to match the score length in order to obtain a better automatic alignment. Afterwards, we compute a cost or similarity matrix of notes on the performance against score notes. As we can see in Figure 4.7 after the cost matrix computation, an optimal path is retrieved in order to found the alignment with less global cost.

Some restrictions have been done to this optimal path computation in order to apply some rules. Horizontal paths are forbidden in order to ensure that each performance

Chord type	Intervals	Example (C as root)
M (major)	0 4 7	C E G
m (minor)	0 3 7	C E♭ G
2 (sus2)	0 2 7	C D G
sus (sus4)	0 5 7	C F G
dim	0 3 6	C E♭ G♭
+ (Aug)	0 4 8	C E G♯
Maj7	0 4 7 11	C E G B
6 (6th)	0 4 7 9	C E G A
m7	0 3 7 10	C E♭ G B♭
m6	0 3 7 9	C E♭ G A
mMaj7	0 3 7 11	C E♭ G B
m7♭5	0 3 6 1 0	C E♭ G♭ B♭
dim7	0 3 6 9	C E♭ G♭ A
7 (7th)	0 4 7 10	C E G B♭
7#5	0 4 8 10	C E G♯ B♭
7♭5	0 4 6 10	C E G♭ B♭
7sus	0 5 7 10	C F G B♭
Maj9	0 2 4 7 11	C D E G B
69 (6/9)	0 2 4 7 9	C D E G A
m9	0 2 3 7 9	C D E♭ G A
9 (9th)	0 2 4 7 10	C D E G B♭
7♭9	0 1 4 7 10	C D♭ E G B♭
7#9	0 3 4 7 10	C D♯ E G B♭
13 (13th)	0 2 4 7 9 10	C D E G A B♭
7♭9♭13	0 1 4 7 8 10	C D♭ E G A♭ B♭
7alt	0 1 3 4 6 8 10	C D♭ E♭ E G♭ A♭ B♭

Table 4.3: Chord description. A list of chords definitions. Numbers on the Intervals column indicate the index of the notes belonging to the chord, (zero indexed, in 12 semitones).

note has just one score note as reference. However, we need to allow vertical paths so one score note can be assigned to several performance notes in order to obtain a minimum cost path, to allow the player add ornamentation notes and to ensure that all performance notes have a score match. In Figure 4.8 we can see an example of an automatic performance to score alignment. In this plot, score has been shifted two octaves up in order to have a better visualization.

If we zoom in to the first 20 notes (Figure 4.9) we can see the result of aforementioned restrictions. Each performance note (bottom half, each colour represents one string) has just one score note as reference while score notes can have multiple per-

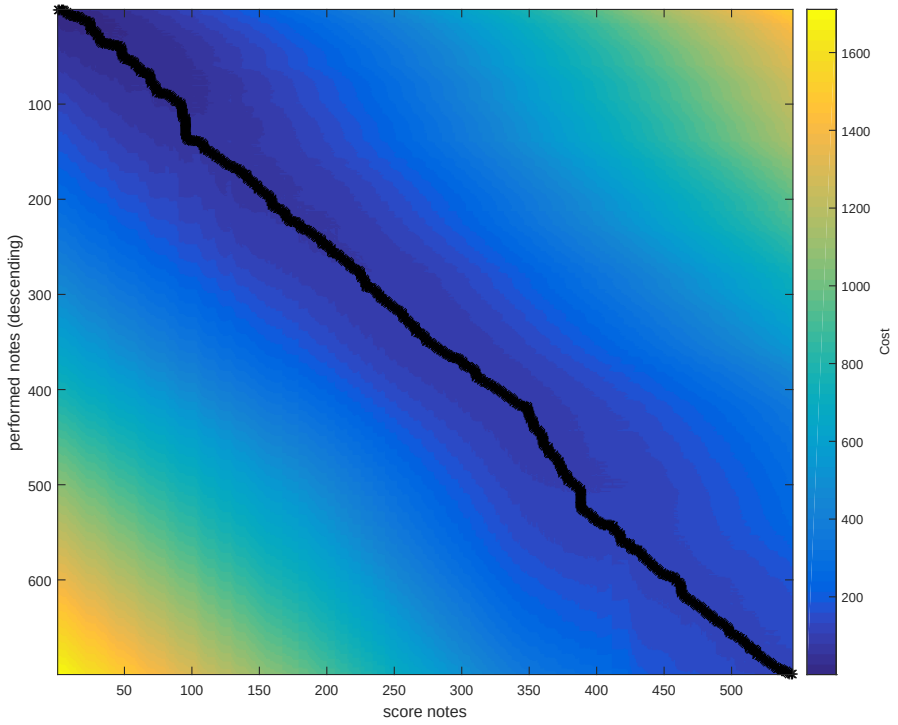

Figure 4.7: *Suite en La* similarity matrix with optimal path,

formance notes assigned. In this plot we can see how chords are usually played in guitar, strumming from low pitch strings to higher ones and not playing all notes simultaneously.

4.3.3 Performance actions

At this point we need to compute the performance actions that will be modelled afterwards using machine learning. This is a simple stage as it only consists in computing variations between score notes and they corresponding performance notes, which alignment was computed in Section 4.3.2. The two performance actions that we will be modelling in this work are *Onset deviation* and *Energy ratio*:

- *Onset deviation*: This performance action is computed by subtracting each performance note onset (in seconds) to its corresponding score note onset.

$$Onset\_dev_i = Ons\_per_j - Ons\_score_i$$

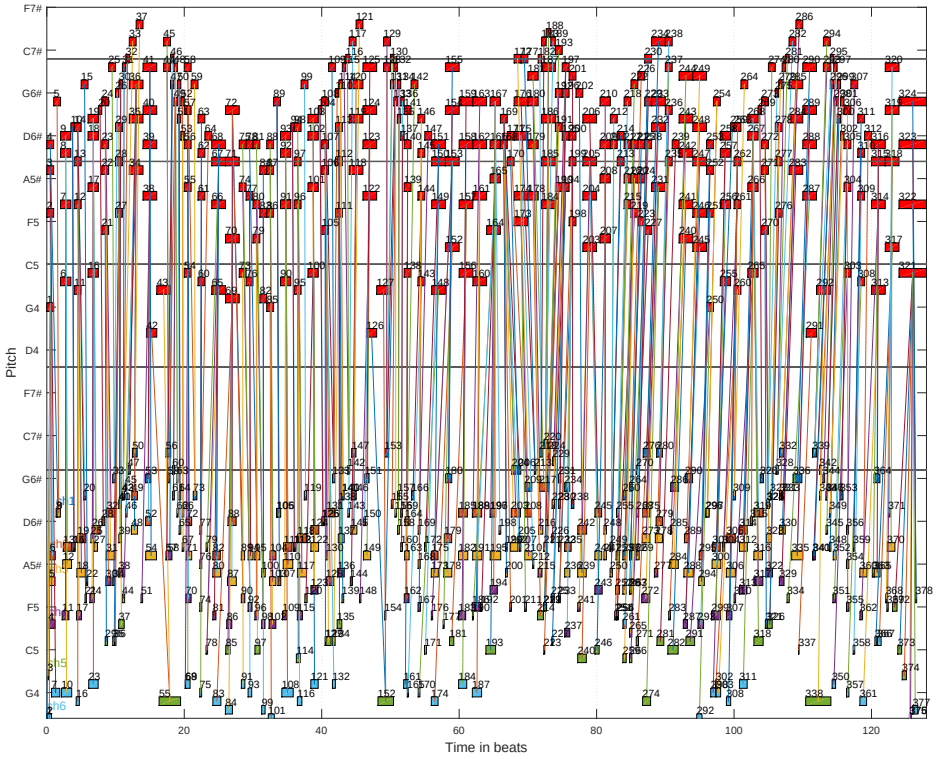

Figure 4.8: *Darn that dream* performance to score alignment. (Score notes are shifted two octaves up)

- *Energy ratio*: Computed by dividing each performance note energy (or midi velocity) to its corresponding score note energy (80 by default as it corresponds to default score midi velocity).

$$Energy\_ratio_i = \frac{V\_per_j}{V\_score_i}$$

Note duration was also considered to be a performance action of this work, but was finally rejected as the duration is not a characteristic of each note but a default characteristic of each guitar or sound, depending on the decay function of it. As each guitar has its own decay, it can be shortened by muting the strings with the picking hand. However, a guitar decay can not be naturally extended, this is only possible affecting the signal by plugging in the guitar to a driven amplifier, an effects stomp box, or any digital signal processing software.

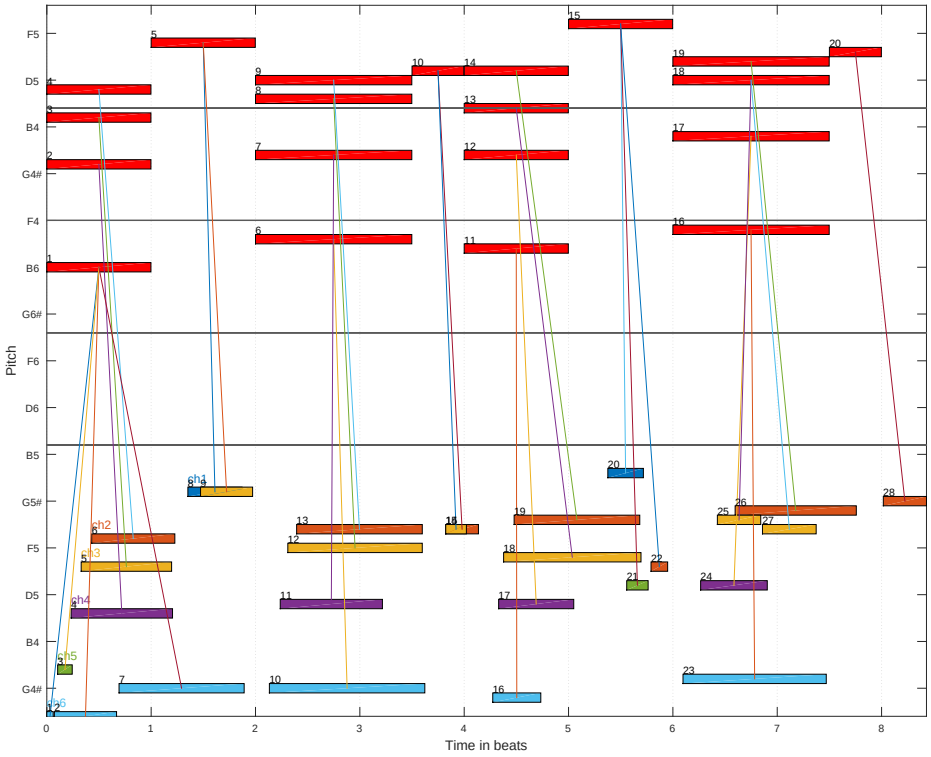

Figure 4.9: *Darn that dream* first 20 notes performance to score alignment.

After modelling these performance actions and in order to compute expressive midis from predicted values we will just reverse the previous formulas in order to compute *Onsets* and *Energy* from deviations and ratios. Predicted *Onset deviation* is added to the score onset and predicted *Energy ratio* is multiplied by the score energy (or MIDI velocity) in order to obtain predicted Onsets and Energy values.

4.4 Machine Learning modelling

In this stage we are trying to predict previous performance actions (*Onset deviation* and *Energy ratio*) with all note descriptors computed in Section 4.3.1. We created a dataset (an `arff` file) for each song and for each performance action. So we have six different datasets, two performances of *Suite en la* and one performance of *Darn that dream*, and we have *Onset deviation* dataset and *Energy ratio* dataset for each performance.

We also build two bigger datasets (one for *Onset deviation* and one for *Energy ratio*) by merging all three performances datasets.

In order to model those datasets and obtain predicted performance actions we are going to use Weka and we are going to train and test different machine learning models in order to see which one models better our data. Feature selection will be also done in order to see which features represent better our data.

Chapter 5

Results

In this chapter we are going to present the obtained results using the previous models, both from a quantitative point of view, by measuring correlation coefficient over predicted data and from a qualitative point of view, by surveying a few listeners with predicted and real performance synthesis. Firstly, in Section 5.1 we are going to explain what measures are we going to consider for both results analysis. In Section 5.2 we are going to apply feature selection in order to see what features are more relevant in order to obtain better prediction for performance actions. In Section 5.3 we present separately the results from both evaluations, quantitative and qualitative. Finally, in Section 5.4 we discuss the obtained results.

5.1 Evaluation Measures

For the quantitative evaluation we are going to use Correlation Coefficient as evaluation metric. Correlation coefficient tells us how much predicted PAs and computed ones are related. It gives values between -1 and 1, where 0 is no relation, 1 is very strong linear relation and -1 is an inverse linear relation.

Dataset	Best Subset	Ranked
"All (onset_dev)"	6, 8, 9, 10, 13, 14, 17	6, 17, 8, 13, 10, 14, 9, 5, 27, 26, 24, 31, 20, 18, 16, 35, 23, 25, 32, 29, 12, 11, 22, 4, 19, 15, 33, 3, 7, 34, 21, 2, 30, 28, 1
"All (energy_rat)	6, 7, 9, 20, 21, 24, 27, 32	6, 7, 22, 32, 9, 24, 20, 27, 5, 35, 23, 31, 1, 8, 2, 10, 13, 14, 19, 4, 18, 26, 3, 17, 30, 11, 15, 33, 34, 25, 12, 28, 29, 21, 16

Table 5.1: Selected Features using Best Subset and Ranker with wrapped Decision Tree. Code for descriptors can be found in table 4.2.

5.2 Feature Selection

In Table 5.1 we present for each PA dataset, two different feature selection algorithms. In the middle column, best subset of features is shown and in the right column all features are ranked using Ranker with wrapped Decision Tree. As we can see, for *Onset Deviation* we achieve the best performance with just 7 features out of 35, being those: *Onset in seconds*, *Measure*, *Previous duration in beats*, *Previous duration in seconds*, *Previous inter-onset distance*, *Next inter-onset distance* and *Simultaneous notes*.

For *Energy Ratio* we achieve the best performance with a subset of 8 features out of 35, being those: *Onset in seconds*, *Duration in seconds*, *Previous duration in beats*, *Key*, *Note to key*, *Chord type*, *Metrical Strength* and *Phrase*

In Figure 5.1 we present the Correlation Coefficients (CC) between predicted and actual Performance Actions for Onset deviation and Energy Ratio while adding features by ranking order. In red we show the accuracy for the whole Train dataset and in blue the results with 10 fold Cross-Validation. For both PAs the best accuracy (using CV) was obtained with the set containing the first 5 best ranked features, as adding more features just makes CC decrease.

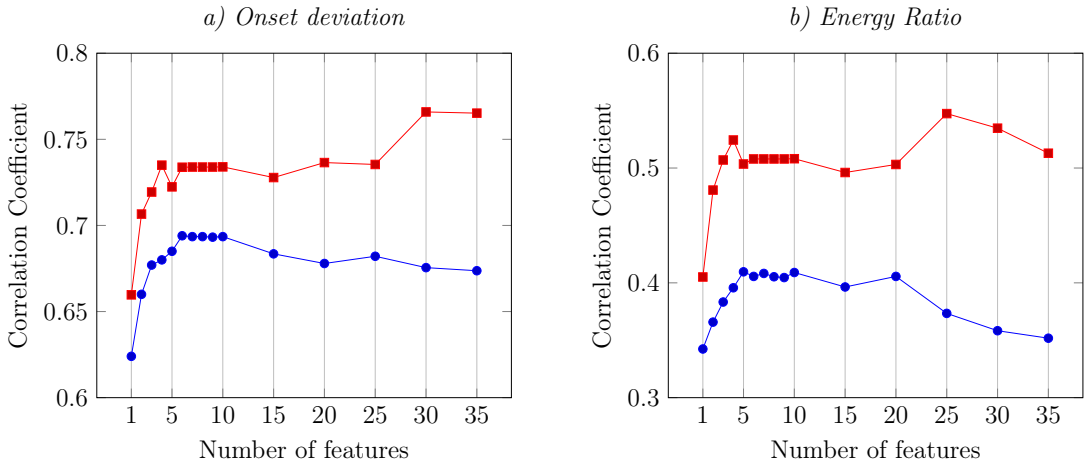

Figure 5.1: Results depending on the number of selected features according to table 5.1. Algorithm used: Decision Tree. Shown values correspond to Correlation Coefficients.

5.3 Evaluation Results

In this section we present both quantitative and qualitative results. The proposed approach was quantitatively evaluated by measuring *Correlation Coefficient* (CC) obtained with the models studied and qualitatively evaluated by asking listeners to compare predicted and real performances.

5.3.1 Quantitative evaluation

In Table 5.2 we show the results comparing different Machine Learning algorithms both with cross-validation and with the whole Train dataset. In the top half of the table we present the results by performance (1 *Darn that dream* and 2 *Suite en La*) and by Performance Action (*Onset deviation* and *Energy ratio*). In the bottom half of the table we merged those three performances into a big dataset and we present the results for this complete large ("All") dataset, for this large dataset with just the 5 top features ("$All_{5features}$", and for this large dataset with the best subset of features ("$All_{bestsubset}$").

Several machine learning algorithms have been tested for each one of this cases of study. From left to right: Decision Trees, K-Nearest Neighbours (K=1), K-Nearest Neighbours (K=2), K-Nearest Neighbours (k=4), Support Vector Machines and

Dataset (feature)	D.Tree cv/train	k_1NN cv/train	k_2NN cv/ train	k_8NN cv/train	SVM cv/train	ANN cv/train
'Darn (energy)'	0.37/0.53	0.18/1	0.27/0.78	0.28/0.52	0.37/0.55	0.26/0.98
'Darn (onset)'	0.70/0.87	0.35/1	0.42/0.83	0.52/0.69	0.57/0.68	0.47/0.99
'Suite (energy)'	0.35/0.59	0.24/1	0.31/0.77	0.32/0.53	0.23/0.38	0.17/0.70
'Suite (onset)'	0.77/0.88	0.28/1	0.35/0.80	0.33/0.53	0.30/0.40	0.29/0.79
'Suite2 (energy)'	0.32/0.70	0.21/1	0.24/0.77	0.17/0.45	0.19/0.31	0.18/0.66
'Suite2 (onset)'	0.83/0.92	0.43/1	0.48/0.85	0.51/0.85	0.44/0.52	0.40/0.78
'All (energy)'	0.35/0.51	0.22/1	0.26/0.78	0.27/0.51	0.21/0.33	0.23/0.63
'All (onset)'	0.67/0.77	0.30/1	0.36/0.81	0.42/0.60	0.39/0.45	0.29/0.67
'All (energy)'$_{5features}$	0.41/0.50	0.30/1	0.37/0.80	0.37/0.57	0.14/0.21	0.14/0.36
'All (onset)'$_{5features}$	0.69/0.72	0.38/1	0.61/0.82	0.65/0.75	0.30/0.31	0.44/0.43
'All (energy)'$_{bestsubset}$	0.41/0.51	0.30/1	0.37/0.79	0.37/0.57	0.16/0.21	0.15/039
'All (onset)'$_{bestsubset}$	0.69/0.73	0.37/1	0.58/0.82	0.64/0.73	0.30/0.32	0.48/0.48

Table 5.2: Results comparing different ML models (10 fold Cross-Validation). All datasets correspond to the three datasets merged into one. Shown values correspond to Correlation Coefficients.

Train	Test	D.Tree		ANN	
		energy	onset	energy	onset
'Darn'	'Suite'	0.013	0.156	0.047	0.008
'Darn'	'Suite2'	0.091	0.183	0.033	0.075
'Suite'	'Darn'	0.017	0.140	0.107	0.032
'Suite'	'Suite2'	0.324	0.392	0.148	0.253
'Suite2'	'Darn'	0.043	0.099	0.079	0.027
'Suite2'	'Suite'	0.240	0.384	0.190	0.227

Table 5.3: Results mixing songs for Train/Test. Whole song Datasets have been used in order to train or test. Shown values correspond to Correlation Coefficients.

Artificial Neural Networks. All this models were computed using Weka.

In Table 5.3 we show the results of training with one dataset and testing with another one. This results are generated using Decision Trees and Artificial Neural Networks as they show to be the best algorithms in Table 5.2. As we can see, the results are very poor as we are mixing two very different music styles (a Jazz standard and a Classical piece). We also can see that if we train and test with different performances of the same score we achieve a Correlation Coefficient around 0.3 which might indicate that PAs are more performance dependant than piece dependant.

	intro1			intro2			middle			end		
	Perf	Pred	Score	Perf	Pred	Score	Perf	Pred	Score	Perf	Pred	Score
Med	40	39	50	63	46	76	45	52	68	42	42	69
Avg	41	43	52	59	48	71	47	48	65	44	46	69

Table 5.4: Numeric results of the on-line survey.

5.3.2 Qualitative evaluation

For the qualitative survey, several synthesised pieces obtained by the models were compared to both the score (dead pan synthesis) and the performed (synthesised version) piece. Participants were asked to to guess how "human" they sounded by comparing among them through an on-line survey. Please see Appendix C for a complete overview of the On-line survey. They were given 4 different tests with three excerpts each one (Performance, Prediction and Score synthesis) and were asked to rate from 0 to 100 the "humanness" of each one related to the other two.

In Table 5.4 we can see the median and average punctuation (over 12 participants) that each concrete audio was given. As we can see, the on-line survey consisted on 4 different excerpts with three different synthesis each one (a Performance synthesis, a Predicted synthesis and a plain Score synthesis). As we saw on the previous Figure 5.2 values for Performance and Prediction are very close and there is a little preference for Score synthesised midis.

In Figure 5.2 we display all On-line survey results gathered by type. I shows that participants perceived the score synthesis more "human" than the actual performance and predicted score. However, we obtained similar results among the performed piece and the predicted one, which might indicate that our models predictions are close to actual human performances. Full survey results can be found at Appendix C.

5.4 Discussion

Analysing the information provided in previous sections it can be seen that: in general, the algorithm which achieves better results using 10 fold Cross-validation is the Decision Trees, outperforming the three proposed K-NN, Support vector machines

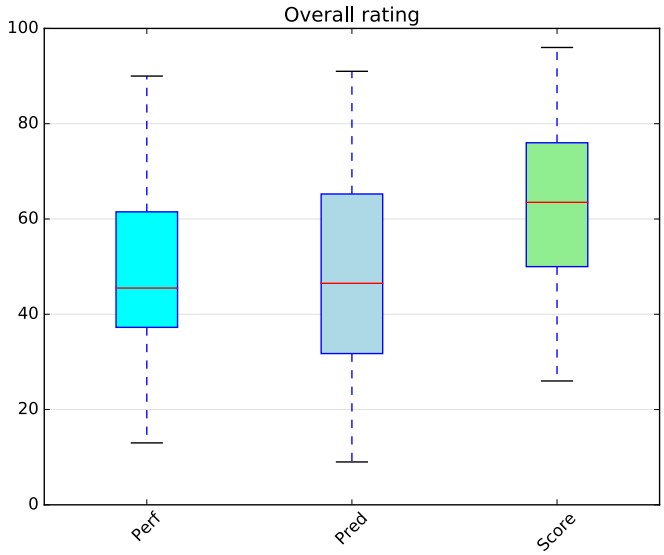

Figure 5.2: Results of the on-line survey with performance, predicted and score synthesised midis.

and artificial neural networks.

Figures 5.3 and 5.4 show an excerpt of *Onset deviation* and *Energy ratio* predictions obtained with Decision Trees, respectively. The curves show *Onset deviation* with respect the score, and *Energy ratio* with respect to the mean loudness . The blue lines correspond to the deviations performed by the musician in the performance, and the red lines correspond to the deviation predicted by the model. In both figures it is shown how the model follows in a consistent way the deviations done in time and energy by the performer.

In Figure 5.5 we can see a piano-roll representation of the first 20 notes with predicted Onsets. As we can see the model follows the typical strumming guitar pattern by playing first the low strings and then the higher ones. This strumming pattern was visible in the performance transcription (see Figure 4.9) where notes were played (*arpeggiated*) from low pitch to high pitch so we can also visually see that our models represent the performance actions done by the player.

In Figure 5.6 we can see how feature selection helps us to improve the results obtained. In the Y-axis we plot the Correlation Coefficient while bars represent different datasets by using all features, the best subset or the 5 best ranked features.

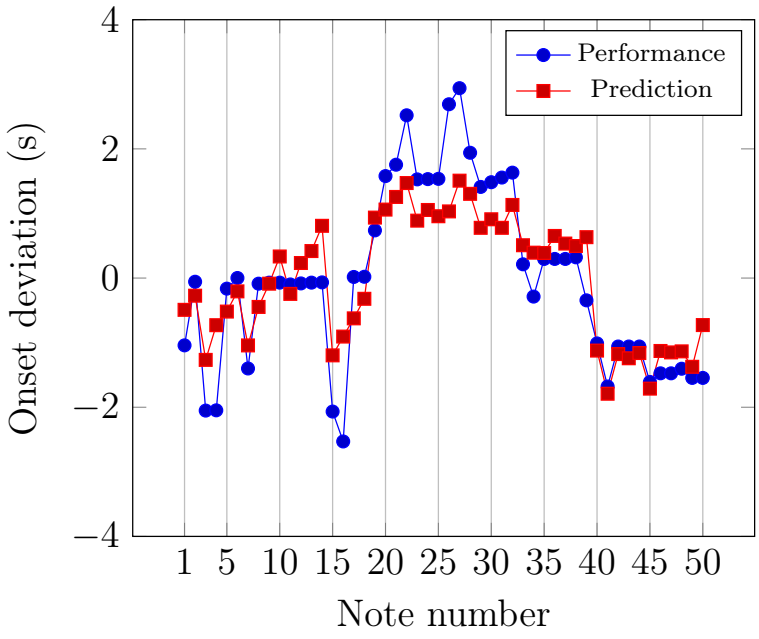

Figure 5.3: Onset deviation in performance and prediction

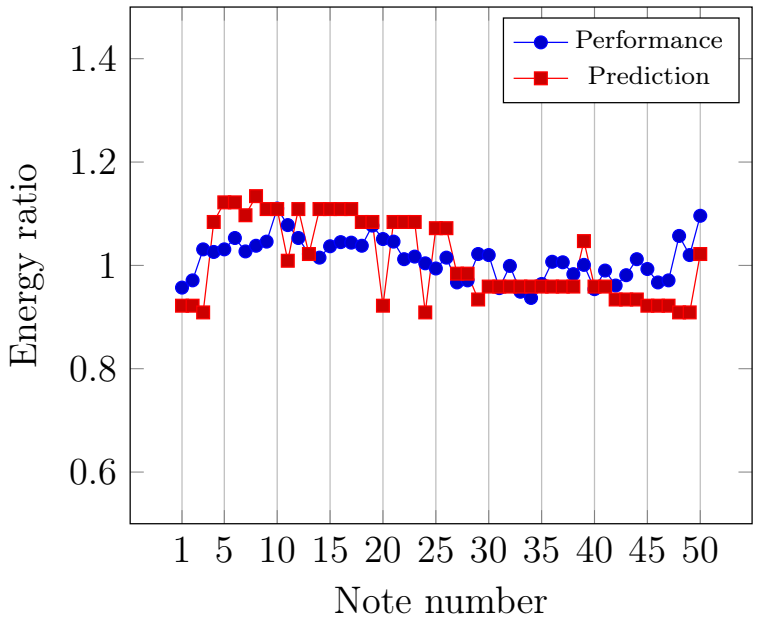

Figure 5.4: Energy Ratio in performance and prediction

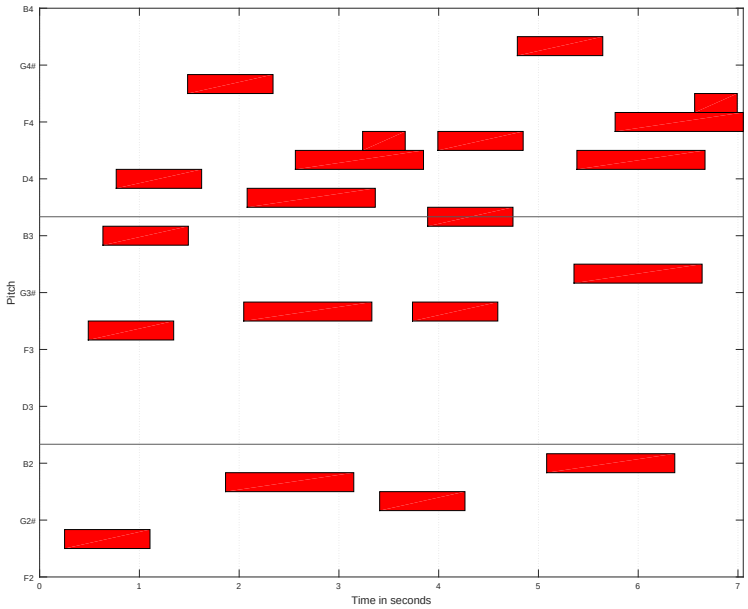

Figure 5.5: *Darn that dream* first 20 notes predicted with piano-roll representation.

Displayed values are obtained from the bottom half of Table 5.2. We can see that by using best subset (red column) or 5 best ranked features (green column) we can improve the Correlation Coefficient by the same amount as we obtain the same results with both feature selection algorithms.

This results are very important as they show that with just a few features (5 features) we can obtain better results than with all features (35 features), so probably it makes no sense to research on adding more and more features to the system but to improve those that obtained a better ranking using feature selection.

Our assumption about why the score synthesis is graded better than the performance or the prediction in the qualitative evaluation is that the virtual instrument does not synthesise very well the guitar and when two notes are played with very short inter-onset time it creates very strange artefacts. Moreover, as it is a midi synthesis we are used to hear it very perfect as it always comes from a music score, so introducing those artefacts and time deviations can sound really strange to us, as if the score wasn't well written.

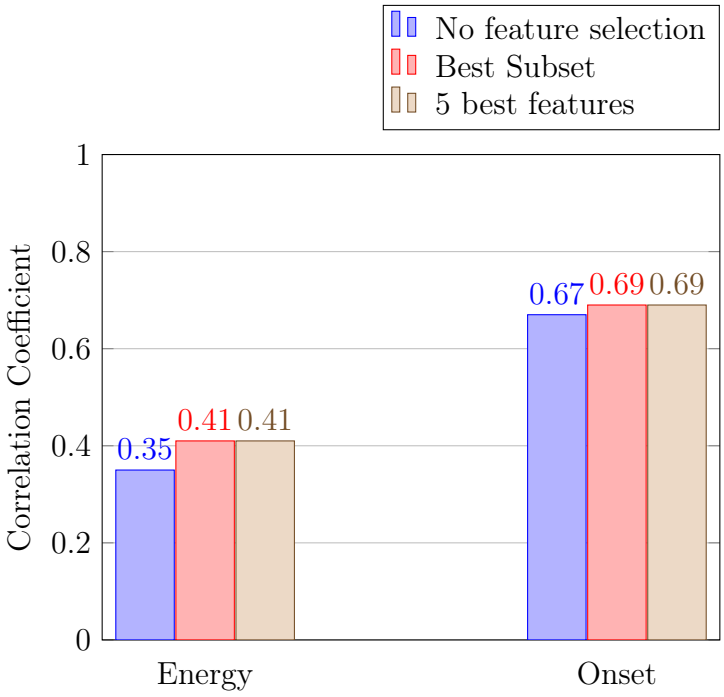

Figure 5.6: Results with feature selection comparison using Decision Trees. In this plot we use the "All" dataset.

One of the comments received in the On-line survey was:

> *I based my "humanness" mainly in the attack of the instrument, when the time between one attack and the next is really short, it feels robotic and hasty. I tried not to guide myself on synthesis but it is difficult. The best of my grades went to a somehow "slower" performance with more "rallentando" than the other versions.*

This comment reinforces our assumption that the problem of higher punctuation to the score synthesis is due to the MIDI synthesis and the artefacts added when two onsets are really close. As the user says, it is very difficult to rate "humanness" from a MIDI synthesis as the sound itself is not human.

Another comment on the survey reflects the difficulty of rating "humanness". The "human" term could be some how ambiguous as it may refer to the smoothness of the performance or to the compositional aspect. In general we found that the qualitative evaluation is subjected to different perceptual aspects (mainly to the guitar

synthesis) that are out of the main focus of this thesis work. However, from this qualitative evaluation we can state two things: firstly, this is not a valid experiment to rate how "human" the predictions are as the users prefer score synthesis over synthesised human performances. Secondly, we think that this evaluation is a valid experiment in order to confirm that our predictions and models are really close to actual human performances.

Chapter 6

Conclusions

In this work we have applied machine learning techniques in order to generate models for musical expression in polyphonic guitar music, by training different models for *Onset Deviation* and *Energy Ratio*. We treated polyphonic guitar as an hexaphonic instrument by capturing and transcribing each string separately. We extracted descriptors from the scores in terms of the melodic (Horizontal) as well from the harmonic (Vertical) context. We computed PAs from the aligned transcribed performance and the scores. We trained different models using machine learning techniques. Models were used to predict PAs that later were applied to the scores to be synthesised. Feature selection analysis and accuracy tests were performed to assess models performance. Perceptual tests were conducted on the predicted pieces to rate how close they sound to a human performance. Results indicate that descriptors contain sufficient information to generate our models able to predict performances close to human ones.

6.1 Contributions

Main contributions of this this Master's thesis are:

- Most of the research in this topic has been carried out in monophonic jazz guitar. As far as our knowledge goes it has not been done in polyphonic

guitar.

- We proposed an new framework extending previous work from monophonic guitar to polyphonic.
- We created a new analysed dataset consisting on hexaphonic audio recordings, their corresponding scores, automatic transcriptions and score to performance alignments.
- We provide an On-line repository with all code and data. Please see Appendix A.
- Our work has shown to have relevance in the research community as it has been accepted at the *MML 2017 - 10th International Workshop on Machine Learning and Music*.

6.2 Future Work

Future work of this Master's thesis could be:

- To enlarge our dataset of hexaphonic recordings with their corresponding digital scores in order to have more solid models.
- To study the interpretability of the different models.
- To improve the performance to score alignment trying to avoid manual preprocessing time-stretch step.
- To study how models generalise into the same musical style, performer,... This means having the same performer performing different pieces and studying variance between pieces or by having the same musical piece played by diverse performers and study how each one performs it.
- To study the musical sense behind feature selection.
- To study a sequential modelling. Current implementation is note-based so each note is modelled by its own descriptors. By implementing sequential modelling, we would be able to model entire phrases or sequences at once.
- To improve our qualitative evaluation (i.e. by improving guitar synthesis)

Bibliography

[1] Giraldo, S. & Ramírez, R. A machine learning approach to ornamentation modeling and synthesis in jazz guitar. *Journal of Mathematics and Music* **10**, 107–126 (2016).

[2] Juslin, P. N. & Sloboda, J. Communicating emotion in music performance: A review and theoretical framework. In *Music and Emotion* (2001).

[3] Ramirez, R. & Hazan, A. A Tool for Generating and Explaining Expressive Music Performances of Monophonic Jazz Melodies. *International Journal on Artificial Intelligence Tools* **15**, 673–691 (2006).

[4] Friberg, A., Bresin, R. & Sundberg, J. Overview of the KTH rule system for musical performance. *Advances in Cognitive Psychology* **2**, 145–161 (2009).

[5] Sundberg, J., Friberg, A. & Bresin, R. Attempts to reproduce a pianist's expressive timing with Director Musices performance rules. *Journal of New Music Research* **32**, 317–325 (2003).

[6] Bresin, R. & Friberg, A. Emotional Coloring of Computer-Controlled Music Performances. *Computer Music Journal* **24**, 44–63 (2000).

[7] Bresin, R. Artificial Neural Networks Based Models for Automatic Performance of Musical Scores. *Journal of New Music Research* **9800** (1998).

[8] Goebl, W. *et al.* ' Sense ' in Expressive Music Performance : Data Acquisition , Computational Studies , and Models. *Artificial Intelligence* 1–36 (2005).

[9] Camurri, A., Dillon, R. & Saron, A. An experiment on analysis and synthesis of musical expressivity. *... of 13th Colloquium on Musical ...* (2000).

[10] Widmer, G. Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence Widmer / Artificial Intelligence* **146**, 129–148 (2003).

[11] Widmer, G. & Tobudic, A. Playing Mozart by Analogy: Learning Multi-level Timing and Dynamics Strategies. *Journal of New Music Research* **32**, 259–268 (2003).

[12] Grindlay, G. & Helmbold, D. Modeling, analyzing, and synthesizing expressive piano performance with graphical models. *Mach Learn* **65**, 361–387 (2006).

[13] Miranda, E. R., Kirke, A. & Zhang, Q. Artificial Evolution of Expressive Performance of Music: An Imitative Multi-Agent Systems Approach. *Computer Music Journal* **34**, 80–96 (2010).

[14] Arcos, J. L., de Mántaras, R. L. & Serra, X. Saxex: A case-based reasoning system for generating expressive musical performances. *Journal of New Music Research* **27**, 194–210 (1998).

[15] Grachten, M., Serra, X. & Universitat Pompeu Fabra. *Expressivity-aware tempo transformations of music performances using case based reasoning* (Universitat Pompeu Fabra, 2006).

[16] Puiggròs, M., Gómez, E., Ramírez, R.-F., Serra, X. & Bresin, R. Automatic characterization of ornamentation from bassoon recordings for expressive synthesis. *9th International Conference on Music Perception and Cognition* (2006).

[17] Bantula, H., Giraldo, S. & Ramírez, R. Jazz Ensemble Expressive Performance Modeling. *Proc. 17th International Society for Music Information Retrieval Conference* 674–680 (2016).

[18] Kirke, Alexis, Miranda, E. R. An Overview of Computer Systems for Expressive Music Performance. In *Guide to Computing for Expressive Music Performance*, 1–47 (2013).

[19] Klapuri, A. P. Automatic Music Transcription as We Know it Today. *Journal of New Music Research* **33**, 269–282 (2004).

[20] Argenti, F., Nesi, P. & Pantaleo, G. Automatic music transcription: from monophonic to polyphonic. In *Musical Robots and Interactive Multimodal Systems*, 27–46 (Springer, 2011).

[21] Klapuri, A. Multiple Fundamental Frequency Estimation by Summing Harmonic Amplitudes. *Proceedings of the International Symposium/Conference on Music Information Retrieval (ISMIR)* 216–221 (2006).

[22] Benetos, E., Weyde, T. *et al.* An efficient temporally-constrained probabilistic model for multiple-instrument music transcription (2015).

[23] Fiss, X. & Kwasinski, A. Automatic real-time electric guitar audio transcription. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 373–376 (2011).

[24] O'Grady, P. & Rickard, S. Automatic hexaphonic guitar transcription using non-negative constraints. *IET Irish Signals and Systems Conference (ISSC 2009)* 22–22 (2009).

[25] Roland. Roland GK-3 divided pickup. URL `https://www.roland.com/global/products/gk-3`.

[26] Bogdanov, D. *et al.* Essentia: An audio analysis library for music information retrieval. In *ISMIR*, 493–498 (2013).

[27] Eerola, T. & Toiviainen, P. Midi toolbox: Matlab tools for music research (2004).

[28] Hall, M. *et al.* The weka data mining software: an update. *ACM SIGKDD explorations newsletter* **11**, 10–18 (2009).

[29] Kraft, S. & Zölzer, U. Beaqlejs: Html5 and javascript based framework for the subjective evaluation of audio quality. In *Linux Audio Conference, Karlsruhe, DE* (2014).

[30] Angulo, I., Giraldo, S. & Ramirez, R. Hexaphonic guitar transcription and visualization. In *TENOR 2016, International Conference on Technologies for Music Notation and Representation.*, 187 – 192 (2016).

[31] Cheveigne, A. D. & Kawahara, H. YIN, a fundamental frequency estimator for speech and music **111** (2002).

[32] Narmour, E. *The analysis and cognition of melodic complexity: The implication-realization model* (University of Chicago Press, 1992).

[33] Cambouropoulos, E. Musical rhythm: A formal model for determining local boundaries, accents and metre in a melodic surface. *Music, gestalt, and computing* **1317**, 277–293 (1997).

Appendices

Appendix A

On-line Resources

An On-line Repository has been created in order to share all of this project's resources, so that researchers and every person interested in the topic can use this information for future work. Code documentation can be found in file: `Readme.md`. Data documentation can be found in Appendix B.

Link to the On-line repository with code and data:
`https://github.com/Marcsiq2/masterthesis`

Also, a few synthesised few synthesised MIDI examples have been uploaded to Soudncloud in order to be accessible to everyone.

Link to Soundcloud:
`https://soundcloud.com/marc-siquier-penyafort/sets/master-thesis`

Finally, the On-line survey code can also be found at GitHub:

Link to On-line survey repository:
`https://github.com/Marcsiq2/Marcsiq2.github.io`

Appendix B

Dataset Documentation

Data for this thesis can be found at our On-line repository (Appendix A) inside the `Files` folder with the following structure:

```
Files
├── dataOut
│   ├── arff
│   ├── arff_cleaned
│   └── nmat
├── extracted_midi
│   ├── Darn_that_dream
│   └── Suite_en_la
├── Figures
├── guitar_in
│   ├── Darn_that_dream
│   ├── Suite_en_la
│   └── Suite_en_la_v2
├── Predictions
│   └── Midis
├── scores
│   ├── midi
│   ├── musescore
│   ├── pdf
│   └── xml
├── Synth
├── weka_files
│   └── results
└── xmlutils
```

In folder `dataOut/arff` we can find the arff files for training the models with extracted features and computed performance actions. Arff files inside `dataOut/arff_cleaned` are the files used in section 5 for the evaluation (divided by Onset, Energy and also "All" database files). In folder `dataOut/nmat` we can find a few Matlab workspaces with all the variables used in the computation.

In folder `extracted_midi` we can find transcribed MIDI files from the audio divided by musical piece (both performances of *Suite en la* are inside `extracted_midi/Suite_en_la` subfolder).

In folder `Figure` we can find diverse figures used in this written report.

In folder `guitar_in` we can find the hexaphonic recordings (a wav file for each string) for each performance.

In folder `Predictions` we can find csv files with the results of the models. First column indicates note number, second column reference value, third column predicted value and fourth column the error difference. Inside `Predictions/Midis` subfolder we can find a few re-constructed midis with the predicted values.

In folder `scores` we can find the scores of the musical pieces in four different formats.

In folder `Synth` we can find synthesised fragments used for the qualitative evaluation.

In folder `weka_files` we can find the experiment description both for the quantitative evaluation and for the feature selection. Inside `weka_files/results` we can find output files provided by Weka.

In folder `xmlutils` we can find a few utils needed in order to be able to read xml files.

Appendix C

On-line Survey

An On-line Survey for the quantitative evaluation was developed using BeaqleJS (browser based evaluation of audio quality and comparative listening environment) which provides a framework to create browser based listening tests and is purely based on open web standards like HTML5 and Javascript.

Server was set in a Github pages account, and can be found at `https://marcsiq2.github.io/`.

In Figure C.1 we can see the starting first page of the survey. This main page shows a few instructions in order to complete the survey as well as my personal contact information. By clicking "Start" the survey starts.

Survey consists of four different listening tests, each one looking like Figure C.2. Three different excerpts are given to the user who can play, pause, rewind and listen to them as many times as he wants. After listening to all of them he is asked to rate *How much "human" are the audios?* from Bad to Excellent by using a slider for each one, so no numerical value is given by the user. By navigating with "Previous Test" and "Next test" he can go back and forth in order to change ratings if he needs to.

After completing the four different tests (12 excerpts in total) the user is presented

	intro1			intro2			middle			end		
	Perf	Pred	Score	Perf	Pred	Score	Perf	Pred	Score	Perf	Pred	Score
1	20	51	43	63	43	83	40	26	75	75	34	40
2	31	29	76	73	33	91	13	29	52	13	25	88
3	61	39	62	48	35	69	57	30	61	45	47	62
4	49	49	55	63	43	83	29	66	76	45	67	78
5	46	37	50	45	47	81	70	50	76	40	37	80
6	35	69	26	70	65	50	39	70	50	32	91	68
7	50	71	49	50	50	50	68	54	87	35	68	69
8	33	72	50	70	71	29	66	68	47	90	47	68
9	40	19	53	49	61	72	49	30	49	20	9	52
10	31	12	50	66	46	96	51	63	67	44	66	88
11	57	30	65	42	31	62	38	32	69	45	34	61
12	40	39	49	73	50	82	41	59	70	40	29	75

Table C.1: Full table results of the on-line survey.

with the "Submit" page (Figure C.3), were he can input his own name and email or write any comment about the test. Those fields are not mandatory. Not many comments were given and the most meaningful one is *I based my "humanness" mainly in the attack of the instrument, when the time between one attack and the next is really short, it feels robotic and hasty. I tried not to guide myself on synthesis but it is difficult. The best of my grades went to a somehow "slower" performance with more rallentando than the other versions.*.

In Table C.1 we can see the punctuation given to each excerpt by each one of the 12 users. This punctuation is directly mapped from the rating slider, being hard right 100 points and hard left 0 points.

In Figure C.4 we can see those punctuations merged by type (Performance, Prediction and Score synthesis) and in Figure C.5 punctuations are divided and plotted by test number.

In Figure C.6 and as a curiosity we can see a runtime plot for each one of the four different tests, being *intro_1* the test with higher average runtime.

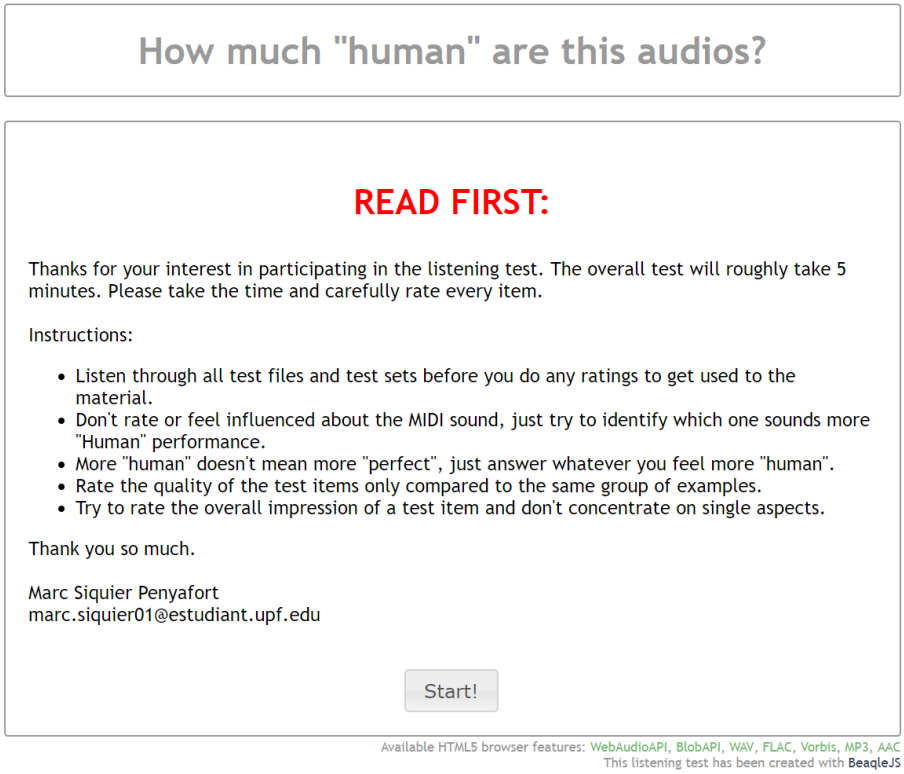

Figure C.1: On-line survey instructions.

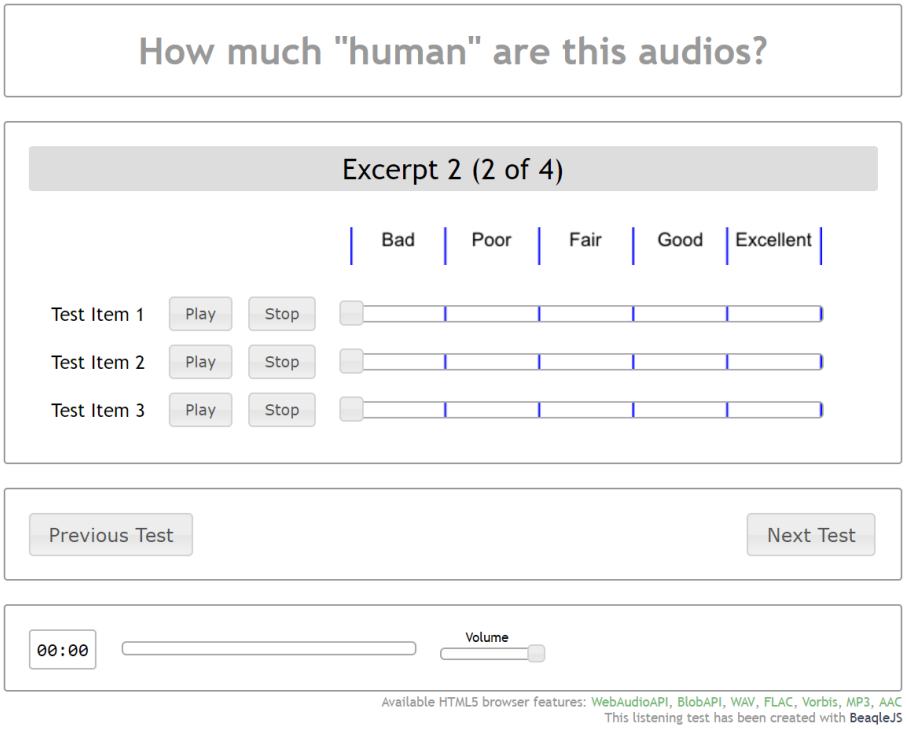

Figure C.2: On-line survey example of a Test set.

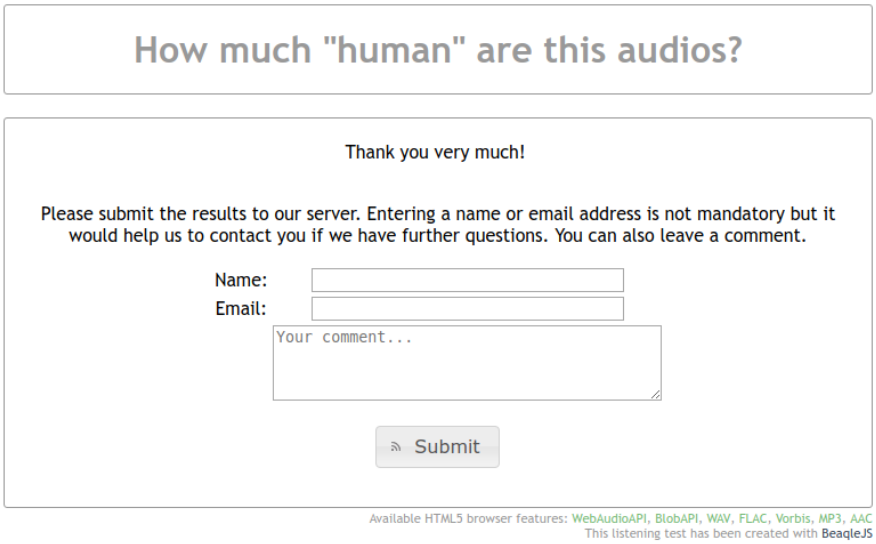

Figure C.3: On-line survey submit page.

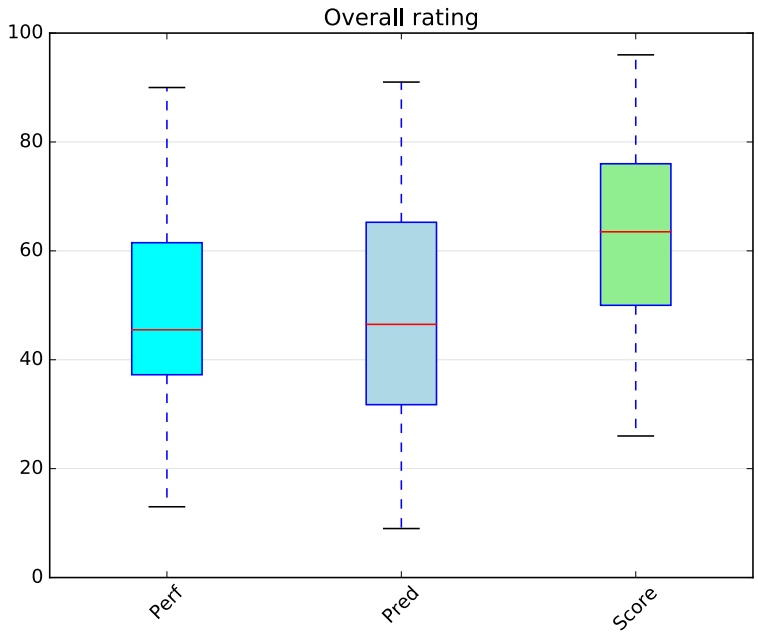

Figure C.4: Results of the on-line survey with performance, predicted and straight score synthesised midis.

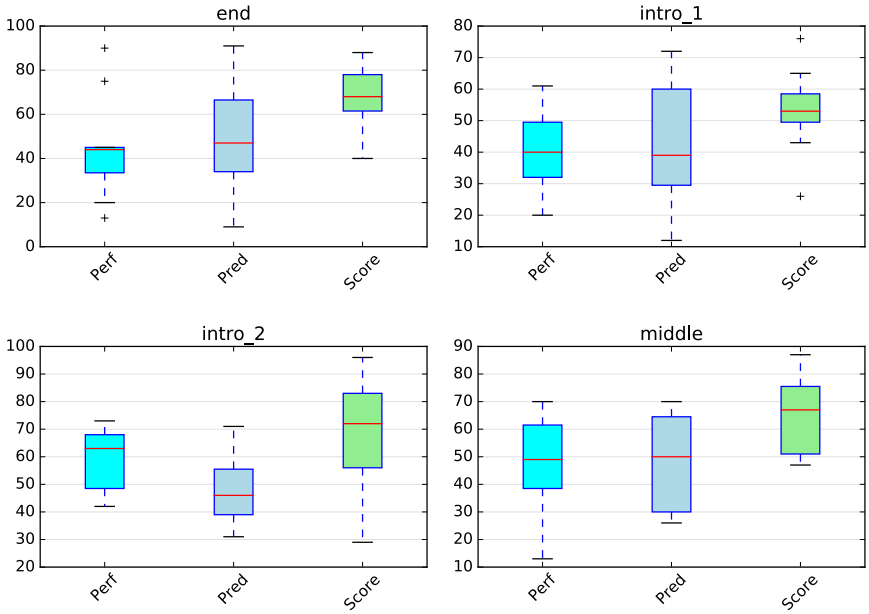

Figure C.5: Results per test of the on-line survey with performance, predicted and straight score synthesised midis.

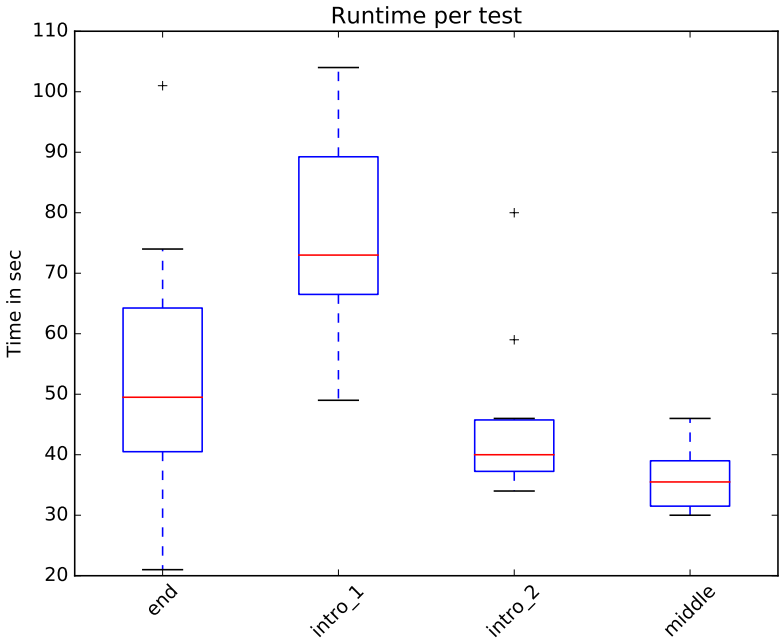

Figure C.6: Runtime for each test of the on-line survey.