



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Head Office: Università degli Studi di Padova
Department of Information Engineering

Ph.D. COURSE IN: Information Engineering
CURRICULUM: Information and Communication Technologies (ICT)

**An innovative computer-based model
for the generation of expressive lead guitar performances**

Coordinator: Prof. Fabio Vandin

Supervisor: Prof. Sergio Canazza

Ph.D. student : Pierluigi Bontempi

*I dedicate this thesis to my children and my parents.
It turns out I fell in between two wonderful generations.*

Contents

Abstract	1
Sommario	3
Acknowledgements	5
1 Computational Expressive Music Performance and lead electric guitar playing	7
1.1 Information engineering and Computational Expressive Music Performance: an introduction	7
1.2 Expressiveness in musical performance	8
1.3 The specific case of the lead electric guitar	11
1.3.1 Historical notes and peculiarities of the instrument	11
1.3.2 Lead electric guitar playing	12
1.4 Understanding performative expressiveness in contemporary popular music: interviews with top producers and musicians	14
1.4.1 Profiles of the interviewees	14
1.4.2 Toward a definition of the nature of performative expressiveness in popular music	16
1.4.3 Verbal indications of the expressive intention	18
1.4.4 Non-verbal indications of the expressive intention	18
1.4.5 Intentional and unintentional expressiveness	19
1.5 An historical introduction and the state-of-the-art in Computational Expressive Music Performance	19
1.5.1 Relation between expressiveness and structure of the musical piece . .	20
1.5.2 Local expressiveness	22
1.5.3 Relationship between emotional intention and expressive parameters . .	25
1.5.4 Relationship between sensorial experiences and expressive parameters .	27
1.5.5 Identification and modeling of the performative styles and of physical and psychological limits of real musicians	28
1.5.6 Conductor systems	30
1.5.7 Contributions of this work to the literature	31

2	Mechanical music and performative expressiveness	33
2.1	Mechanical music	33
2.2	A brief history and operating principles of the player piano	34
2.3	Are metronomic rolls expressive?	35
2.3.1	Dataset	36
2.3.2	Algorithms and implementations	37
2.3.3	The DBSCAN-based solution	39
2.3.4	The mathematical peak detection and analysis solution	41
2.3.5	Final considerations	42
3	Modeling of fingering choices	45
3.1	The fingering problem	45
3.2	Review of the scientific literature on automatic fingering generation	48
3.3	A new, more comprehensive and flexible approach to automatic fingering generation	50
3.3.1	High-level description of the system	50
3.3.2	Implementation	53
3.4	Ongoing development of the system	60
4	Modeling of articulations and expressive techniques	61
4.1	Defining the distribution of articulations and expressive techniques: a statistical approach	61
4.2	The rule-based system for automatic techniques insertion	63
4.2.1	Bending	64
4.2.2	Hammer-on	65
4.2.3	Pull-off	66
4.2.4	Legato slide	67
4.2.5	Vibrato	68
5	Unintentional expressiveness due to the biomechanical dimension: a Machine Learning-based modeling approach	71
5.1	Modeling unintentional expressiveness	71
5.2	The dataset	72
5.3	Descriptive statistics	78
5.3.1	Onsets' time deviation	78
5.3.2	Position and string change, versus preceding rest	79
5.3.3	Correlation matrix	80

5.4	Predictive models	81
5.4.1	Onsets	84
5.4.2	Offsets (preceding rest)	85
5.4.3	Velocity	86
6	Control of intentional expressiveness	91
6.1	Conductor systems	91
6.2	Loudness control	92
6.3	Onsets timing control	93
6.4	The end-to-end implementation of the expressive model	93
6.4.1	Overall structure	93
6.4.2	The GUI	96
6.4.3	The output MIDI File	97
6.5	Sound-based control	97
7	The next step: audio rendering with innovative virtual instruments	101
7.1	Digital Audio Workstations, virtual instruments and expressivity	102
7.1.1	Tools commonly available in VIs	102
7.1.2	Multiple articulations-based VIs	103
7.2	Expressiveness at a symbolic level or at the level of generation and manipulation of the audio signal?	104
7.3	Why MIDI 1?	107
8	Conclusion	111
8.1	Thesis path, objectives achieved, and possible future developments	111
8.2	Potential fields of application	112
8.2.1	Music Production	112
8.2.2	Education	114
8.2.3	Automatic music generation	114
8.2.4	Automatic generation of guitar tablatures from MIDI materials	114
8.3	Multidisciplinary and interdisciplinary research and computer engineering	115
	Appendix	117
	Scientific publications	149
	Activities in support of research	151
	Activities for the Third Mission	153
	Thesis correlations	157
	Popular publications	159

Abstract

Performative musical expressiveness can be attributed to the manipulation, more or less evident, of parameters associated with the macro categories of timing, dynamics, and timbre. The purpose of an expressive performance may vary depending on the specific musician and the sociocultural and stylistic context in which the performance takes place. Among the objectives commonly identified in the literature there are the expression (and/or induction in the listener) of emotions, the clarification of the piece's formal structure, the sonic rendering of concepts and sensory perceptions, and adherence to a specific stylistic current. Additionally, it is evident for some musicians that they imprint their performances with distinctive expressive traits that make them recognizable to the listener and that transcend the aforementioned objectives. The field of computational musical expressiveness aims to develop automatic models that are ideally capable of generating virtual musical performances endowed with expressiveness, similar to what occurs with human musicians. From the information engineering perspective, computational performative musical expressiveness has historically involved the fields of affective computing (of which it has also been considered a subfield), artificial intelligence, human-computer interaction, biomechanical simulation, and robotics. Although these fields are undoubtedly at the heart of computational musical expressiveness, this research area is also inherently highly multidisciplinary, requiring expertise at least also in psychological and musicological areas. The first automatic expressive models date back to the 1980s. Since then, numerous solutions have followed, based on rules, statistical approaches, or, in recent years, Machine and Deep Learning.

The stylistic reference domain has been, at least in a largely predominant manner, that of European art music from the 17th, 18th, and 19th centuries. This can be partially explained by a general academic cultural bias oriented towards music commonly referred to as "classical". Another reason Euro-classical music has historically been a protagonist in the field of computational expressiveness is that it is based on the score/performance dualism, thus allowing a precise analysis of the performance's expressive deviations compared to the score's prescriptive dimension. The musical instrument most often referred to has been the piano. Besides being one of the most representative instruments of the Western musical tradition of recent centuries, this involves a significantly smaller number of variables associated with the expressive dimension (and therefore it is more easily modelable) than other instruments. The pianist essentially has only the dynamic control of the notes (given by the speed with which the keys are struck), the control of the moments when each key is pressed or released, and the use of the pedals.

This research work, on the other hand, addresses the stylistic domain of contemporary popular music, specifically lead electric guitar parts, a field so far explored only very partially. This has firstly required understanding what "expressive performance" means in the reference

context. Also the instrument's physical characteristics, the performative techniques used by guitarists, and the biomechanical limits or constraints that may arise in the interaction between performer and guitar had to be analyzed. The following step was an in-depth review of the scientific literature on computational musical expressiveness, also to understand which of the previous works could serve as inspiration, possibly with specific modifications, in the new application context. The development of the expressive model was thus based on the use of optimization techniques for the automatic generation of the basic virtual fingering, on a rule-based system for the automatic insertion of articulations and expressive techniques, on a Machine/Deep Learning approach for the simulation of timing and dynamics deviations primarily due to involuntary biomechanical and psycho-perceptive components, and finally on the user's introduction of indications regarding the deliberate expressive component (specifically concerning loudness and positioning of the notes relative to the beat). The various modules of the model, based on an input melody in MIDI format, generate a detailed MIDI description of the guitar performance, which can then be sonified through a virtual instrument.

The potential applications are numerous and include, in particular, the use of the model in the educational and music production sectors.

Sommario

L'espressività musicale performativa può essere ricondotta alla manipolazione, più o meno evidente, di parametri associabili alle macrocategorie di tempo, dinamica e timbro. Il fine di una performance espressiva può variare a seconda dello specifico musicista e del contesto socioculturale e stilistico in cui si colloca l'esecuzione. Tra gli obiettivi comunemente individuati in letteratura figurano l'espressione (e/o l'induzione nell'ascoltatore) di emozioni, il chiarimento della struttura formale del brano, la resa sonora di concetti e percezioni sensoriali, l'adesione ad una determinata corrente stilistica. Appare inoltre rilevante in alcuni musicisti la tendenza a imprimere nelle proprie performance tratti espressivi peculiari, che li rendono riconoscibili all'ascolto e che trascendono gli obiettivi sopra elencati. Il campo dell'espressività musicale computazionale si occupa di sviluppare modelli automatici idealmente in grado di generare performance musicali virtuali dotate di espressività, analogamente a quanto avviene nel caso di musicisti umani. Dal punto di vista dell'ingegneria informatica, l'espressività musicale performativa computazionale ha storicamente coinvolto i settori dell'affective computing (di cui è stata anche considerata un sottocampo), dell'intelligenza artificiale, dell'interazione uomo-macchina, della simulazione biomeccanica, della robotica. Sebbene tali ambiti siano indubbiamente collocati al cuore dell'espressività musicale computazionale, il campo di ricerca in oggetto è per sua natura fortemente multidisciplinare, e richiede competenze quantomeno anche di area psicologica e musicologica. I primi modelli espressivi automatici risalgono agli Anni '80 del secolo scorso. Da allora si sono succedute numerose soluzioni basate su regole, approcci statistici o, negli ultimi anni, Machine e Deep Learning.

L'ambito stilistico di riferimento è stato, quantomeno in parte largamente preponderante, quello della musica colta europea dei secoli XVII, XVIII e XIX. Ciò è parzialmente spiegabile con un generico bias culturale accademico orientato verso la musica comunemente indicata come "classica". Un altro motivo per cui la musica eurocolta è stata storicamente protagonista nel campo dell'espressività computazionale è che si basa sul dualismo partitura/esecuzione, e permette quindi un'analisi puntuale delle deviazioni espressive della performance rispetto alla dimensione prescrittiva dello spartito. Lo strumento musicale d'elezione è stato invece il pianoforte. Questo, oltre ad essere uno degli strumenti più rappresentativi della tradizione musicale occidentale degli ultimi secoli, comporta una quantità di variabili associabili alla dimensione espressiva notevolmente più contenuta (e quindi più facilmente modellabile) rispetto ad altri strumenti. Il pianista ha infatti a disposizione sostanzialmente solo il controllo dinamico delle note (dato dalla velocità con cui i tasti vengono colpiti), il controllo degli istanti in cui ciascun tasto viene premuto o rilasciato e l'utilizzo dei pedali.

Questo lavoro di ricerca si rivolge invece all'ambito stilistico della popular music, e nello specifico alle parti soliste di chitarra elettrica, un campo esplorato sinora solo in maniera

estremamente parziale. Ciò ha determinato innanzitutto la necessità di comprendere cosa significhi “performance espressiva” nel contesto di riferimento. Si sono dovute poi analizzare le caratteristiche fisiche dello strumento, le tecniche performative utilizzate dai musicisti e i limiti o vincoli biomeccanici che possono sorgere nell’interazione tra performer e chitarra. Il passo successivo è stata un’approfondita revisione della letteratura scientifica relativa all’espressività musicale computazionale, anche al fine di comprendere quali dei lavori precedenti avrebbero potuto essere di ispirazione, ed eventualmente con quali modifiche, nel nuovo contesto applicativo. Lo sviluppo del modello espressivo si è quindi basato sull’utilizzo di tecniche di ottimizzazione per la generazione automatica della diteggiatura di base, su un sistema basato su regole per l’inserimento automatico di articolazioni e tecniche espressive, su un approccio in Machine/Deep Learning per la simulazione delle deviazioni espressive di tempo e dinamica dovute primariamente alla componente biomeccanica involontaria e infine sull’introduzione da parte dell’utente di indicazioni circa la componente espressiva volontaria (riguardanti nello specifico l’intensità e il posizionamento delle note rispetto al beat). I diversi moduli del modello, sulla base in ingresso di una melodia in formato MIDI, arrivano a generare una descrizione dettagliata della performance chitarristica, che può poi essere sonorizzata tramite uno strumento virtuale.

Le ricadute potenziali possono essere numerose, e comprendono in particolare l’utilizzo del modello nei settori educativo e della produzione musicale.

Acknowledgements

A journey lasting three years inevitably calls for numerous acknowledgments. Many people have shared this path with me, whether in a scientific, professional, friendly, or family context. I will thank them in this order, not as a ranking of importance. Moreover, when traveling together, as in this case, the boundaries between scientific activity, work, and friendship often blur.

My first thanks go to Sergio Canazza, my supervisor, who believed in me and supported me throughout this doctoral journey, from the project proposal to the present. I am also profoundly grateful to the wonderful people at the CSC for the shared moments, laughter, support, and scientific growth. Some have moved on to further their careers elsewhere, but once you join the CSC, you're always part of the family. In alphabetical order, I thank Valentina Burini, Filippo Carnovalini, Nadir Dalla Pozza, Giovanni De Poli, Alessandro Fiordelmondo, Andrea Franceschini, Roberta Bianca Luzietti, Nicolò Merendino, Giulio Pitteri, Niccolò Pretto, Antonio Rodà, Alessandro Russo, Matteo Spanio, Alvisè Vidolin, Anna Zuccante, and Giada Zuccolo. Many thanks to Francesco Silvestri for his willingness to participate in the committees for my admission to the third year of the PhD program and the presentation of the thesis to the external reviewers, and for his valuable advice and insights.

I want to give a special thanks to Alexandre D'Hooze, Daniele Manerba, and Pietro Zappalà, who are mentioned here as my coauthors. However, with Daniele and Pietro, I share much more than just our scientific collaborations.

I am grateful to the two reviewers of this thesis, Leandro Costalonga and Carlo Drioli, whose careful reading and numerous valuable comments enabled me to improve the initial text significantly.

Thanks to Aldo Orvieto for my involvement in the artistic and research project on the pieces by Gian Francesco Malipiero preserved on paper roll for player piano.

Thank you to Milva Badalotti and Claudio Vela, who have wholeheartedly encouraged and supported this journey without hesitation. A special thanks also to Roberta Licitra, who took on much of the work I would typically handle at the Department of Musicology and Cultural Heritage over the past three years.

Thanks to Erika Bertella, Matteo Biatta, Federica Iandico, Fabio Paterlini, Luana Savastano, Laura Taglietti, and Daniela Zani. Thanks to you, life is more beautiful.

Finally, thanks to my parents for their love and support, which have never been lacking, and to my children Angela and Giorgio, who make every moment spent with them a moment to remember.

Chapter 1

Computational Expressive Music Performance and lead electric guitar playing

Chapter introduction

When a musician plays a piece, s/he manipulates multiple parameters (related to tempo, intensity, and timbre) in order to make the performance expressive. The field of Computational Expressive Music Performance deals with creating computer models capable of automatically generating expressive virtual performances, usually by receiving the equivalent of the sheet of music for the human musician as input. This thesis is devoted to the creation of an innovative automatic expressive model dedicated to lead electric guitar in popular music. In this chapter, the definition of “performative expressiveness” in the specific field of interest is addressed, general information about the musical instrument under study is offered, and a review of relevant scientific literature is conducted.

1.1 Information engineering and Computational Expressive Music Performance: an introduction

The research field of Computational Expressive Music Performance (CEMP), the scope of this thesis, is concerned with developing computer models capable of generating virtual musical performances as comparable as possible with human ones [1]–[3]. Many sectors of information engineering have historically been involved in the field. Among the main ones stands out affective computing, a research area first introduced in 1997 by R. W. Picard [4]. Broadly

speaking, this field deals with giving computers the ability to recognize, express, and - in a sense - “have” emotions (the topic will be explored further, in relation to CEMP, in Section 1.5.3). More generally while, seven decades after the seminal *Proposal for the Dartmouth Summer Research Project on Artificial Intelligence* by J. McCarthy [5], an unambiguous and universally agreed definition of Artificial Intelligence (AI) still seems difficult to achieve (see [6], [7]), CEMP, at least in a significant part of its incarnations, particularly in recent times, is a full part of it. Another engineering field strongly involved in computational musical expressiveness is that of human-computer interaction, specifically in the area of so-called conductor systems (explored in more detail in Section 1.5.6 and Chapter 6). Finally, although to an at least quantitatively secondary extent, the CEMP field has involved the biomechanical study of musical performance and its computer simulation or replication through robotics (see, for example [8], [9], and Chapter 5).

Before operationally tackling the development of any automatic expressive model, it is, in any case, crucial to clarify what is meant in general and in the specific context of application by “performative expressiveness”. It is also necessary to investigate in detail the nature of the musical instrument of reference and the interaction between it and the musician. Finally, one must have a broad knowledge of previous scientific contributions in the field. This first Chapter is devoted to these topics.

1.2 Expressiveness in musical performance

When skilled musicians perform a piece, they typically avoid playing it in a purely mechanical manner, with perfectly steady timing and no variations in loudness or timbre. Instead, expert musicians often adjust various performance elements to create expressive and dynamic interpretations of the music:

[...] performers are able to use systematic variations in performance parameters to convey emotion and structure to listeners in a musically sensitive manner ([10] p. 64).

As will be explained in more detail in Chapter 5, such systematic variations can be observed in their phenomenological dimension, independently of the causes and mechanisms that lead to their occurrence, or one can try to separate the intentional (cognitive) dimension from the unintentional one (related to the biomechanical component and the non-conscious cognitive dimension).

With regard to Western classical music, expressiveness has often been defined in terms of deviations from what is prescribed in the score, at least since the systematic analyses presented in the seminal book *Psychology of Music*, by C. E. Seashore [11].

Also L. B. Meyer considered the performance as the space of freedom between the written page and its sound rendering [12]:

Qualitatively the performer's role is always the same; he is always an active creator, shaping and molding the abstract scheme furnished him by the composer [...]

In recent years, this conception of performative expressiveness continues to be commonly widespread, as verifiable in the excellent review of the CEMP field published in 2018 by C. E. Cancino-Chacón et al.:

When playing a piece, expert performers shape various parameters (tempo, timing, dynamics, intonation, articulation, etc.) in ways that are not prescribed by the notated score, in this way producing an expressive rendition that brings out dramatic, affective, and emotional qualities that may engage and affect the listeners. ([2] p. 1)

A different approach to the definition of expressiveness in music performance was that proposed by B. H. Repp, in which the deviation in expressive parameters is observed not with respect to the score but to a standard, “normal” performance of the piece [13]. Another possible perspective is the one based on the complex and multifaceted concept of gesture, described by R. I. Godøy and M. Leman [14].

Before continuing with the discussion, it should be highlighted that the focus of this thesis is on a different stylistic and socio-cultural sphere from the one most commonly investigated in research on (computational) performative expressiveness, namely popular music (reference is made to the common tripartition between classical, folk and popular music). More specifically, it was decided to address the peculiar case of lead electric guitar. There are many reasons for this, starting with the fact that I myself studied and taught electric guitar for many years. The lead electric guitar is also particularly complicated and fascinating to model on an expressive level since, as will become apparent in the following chapters, it grants the performer a broad spectrum of possibilities linked to the nature of the instrument and the numerous playing techniques available. Moreover, the academic attention given to this area has historically been relatively modest, thus leaving ample unexplored space for research.

Although all the definitions of performative expressiveness reported above are relevant in this specific field of interest, none seems - at least by itself - capable of completely clarifying what performative musical expressiveness is in the specific field investigated here.

In popular music, there is often no explicit score. Even when the performance is based on prescriptive written notation forms, these rarely reach the level of detail typical of classical scores (for example, lead sheets, an essential form of notation that describes the fundamental

elements of the song, are commonly used [15]). The clear distinction between composer and performer cannot be taken for granted in the popular field. Parts can be improvised, can be created and memorized by the musician without using any formal notation, they can also arise from the interaction between engineers, producers and musicians [16]. Furthermore, the rise of computer-assisted music production has increasingly blurred the lines between traditional roles in popular music creation. The distinctions between composer, performer, engineer, and producer are now merging, leading to a more integrated approach to music creation [17]. As a result, the concept of authorship in pop is more fluid and multifaceted than clearly defined and unambiguous [18]. It goes without saying that a definition of performative expressiveness based on the relationship between the written part and performance is out of focus in this context. According to musicologist Vincenzo Caporaletti, it is misleading to even try to understand popular music through the cognitive approach he describes as ‘visual’, typical of classical music. Instead, popular music would fall among the ‘audio-tactile’ ones, based instead on a direct relationship between auditory perception and the sense of movement, with a particular relevance of the bodily dimension and technological mediation (recording practices instead of sheet music) [19].

Moreover, even definitions not based on the score/performance dualism show obvious limitations in the field of popular music. On the one hand, it is generally difficult, if not senseless, to try to identify a reference performance, to be used as a yardstick for evaluating other performances of the piece. On the other hand, the abstract concept of gesture, while certainly applicable to defining performative expressiveness in popular music, is inherently broad. To be truly meaningful or explanatory, it requires further specification and contextualization.

This does not mean, of course, that there is no expressivity or research in this direction on the part of popular musicians, but rather that, before delving into the actual implementation of computational expressive models, profound analysis and deep reflection on the very foundations of performative expressiveness in popular music are required.

Added to this knot to be tackled is that of the chosen instrument of investigation, the (lead) electric guitar. In most cases, in fact, the scientific literature relating to CEMP has referred to the piano, while the contributions dedicated to the electric guitar are minimal. Even independently of this, however, it is believed that an understanding of the general characteristics of the instrument of interest is essential to properly exploring its expressive potential and actual use.

In the following sections of this introductory Chapter, therefore, the electric guitar will be introduced from the points of view of its history, physical characteristics and most common associated performance techniques. This will be followed by a summary of a series of interviews conducted with musicians, music producers, and educators regarding the nature and

characteristics of performative expressiveness in popular music. Finally, the existing literature relating to CEMP will be analysed, evaluating its applicability or, in any case, its interest with respect to the specific field being researched here.

1.3 The specific case of the lead electric guitar

1.3.1 Historical notes and peculiarities of the instrument

The electric guitar, the instrument around which this thesis revolves, originated as a twentieth-century evolution of the modern acoustic or classical guitars [20]. It shares some of its main characteristics with the latter, namely the fact of being a plucked string instrument, usually equipped with six strings tuned as E2 (82,407 Hz)-A2-D3-G3-B3-E4.

Etymologically, the term *guitar* can be traced back to the Greek instrument *kithara*, of which vascular representations dating back to the centuries between the seventh and fifth BC have reached us [21]. However, the *kithara* belongs to the family of yoke lutes, or lyres, which have only partial affinities with the modern guitar. More relevant to the later development of the guitar was the oud, an instrument of Middle Eastern origin that was also a predecessor of the European lute, of which there is evidence from at least the 10th century AD [22]. During the Renaissance, the Spanish vihuela, a close precursor of the modern guitar, emerged [23]. The Baroque period saw the five-course guitar (with paired strings) becoming popular in Europe, particularly in Spain and Italy. Between the second half of the 18th and the early 19th century, the six-string guitar became standardized [24]. The 20th century saw the rise of the steel-string acoustic guitar, which became a staple in folk and popular music.

The electric guitar was born between the 1920s and 1930s of the 20th century, in the widespread context of the attempt, using electric instruments, to amplify the volume of the acoustic guitar. The paternity of the invention is disputed between the historic manufacturers Fender, Gibson, Bigsby, and Rickenbacker. However, there is general agreement as to which was the first solid body electric guitar to be mass-produced, namely the Fender Telecaster (from 1951, still in production today) [20].

The essential construction components of electric guitars (see Figure 1.1 for an example) are [25]:

- **Body**

Electric guitars usually have wooden solid bodies (the body of the instrument has no cavities or resonance chamber). There may also present, rarely, semi-hollow bodies;

- **Neck**

The necks of electric guitars are commonly made of maple or mahogany. They can be

bolted or glued to the body. The fretboard, on which the metal keys are placed, is often (but not always, particularly in the case of maple) carved from a separate piece of wood from the neck, and is often made of rosewood, maple, or ebony. The neck profile, scale length (the distance between the two string resting points, typically the nut and the bridge), and fret count can vary, as the number and - sometimes - tuning of the strings;

- **Pickups**

The pickups are electromagnetic devices that convert string vibrations into electrical signals;

- **Bridge**

The bridge anchors the strings to the body. It can be fixed (as in the typical Gibson Les Paul *Tune-o-Matic* or Fender Telecaster *Hardtail* designs) or floating (as in the standard Fender Stratocaster *Tremolo* bridge, or in the Floyd Rose solution). Floating bridges allow for pitch bending and vibrato effects;

- **Controls**

Electric guitars feature various controls, the most important and common being volume and tone knobs, and pickup selector.



Figure 1.1: A Fender Stratocaster Road Worn 50s (picture by irish10567 from Little Falls, NJ, USA, CC BY 2.0 - <https://creativecommons.org/licenses/by/2.0>, via Wikimedia Commons)

1.3.2 Lead electric guitar playing

Generally speaking, the guitar can produce notes through the use of open (not fingered) strings or by the combination of string, fret, and finger of the fretting hand. In most cases, the same notes

can be generated in several different ways, i.e. with different string/fret/finger combinations and/or with the use of open strings. This applies to the performance of accompaniment parts as well as melodies and lead parts.

From the 1960s in particular, with the rise of the so-called *guitar gods* (in the beginning, among many others, Eric Clapton, and Jimi Hendrix), the lead use of the electric guitar has played an essential role in the development of the instrument's history and popular perception [26]. The search for new virtuosic techniques and the refinement of pre-existing ones led then to a generation of musicians who, especially between the 1980s and 1990s, laid the cornerstones of performative expressiveness and virtuosity in lead electric guitar (among many others Eddie Van Halen, Yngwie Malmsteen, and Steve Vai). Today, lead electric guitar playing involves a range of techniques that cannot be ignored in the development of dedicated computational expressive models. A broad division can be made between articulations (ways in which a note can be sounded) and expressive techniques (used for the precise purpose of introducing expressiveness into the performance) [27]. Clearly, there may be at least partial overlap between the two categorizations. The most frequently used techniques, the ones which were taken into account in the development of the computational model that is the main subject of this thesis, are listed below ¹.

Articulations

- *Standard picking*: the string, open or fingered, is picked with the use of a plectrum to be set into vibration;
- *hammer-on*: this technique involves using a finger from the fretting hand to hit the string from above, making it vibrate without being picked;
- *pull-off*: a finger of the fretting hand “pulls” the string, producing a note lower than the previous one without the need to pick the string.

Expressive techniques

- *Bending*: it is a technique in which the player increases the pitch of a string-fret combination by pushing up or pulling down, perpendicular to the keyboard, the string with one or more fingers. A similar effect, even on open strings, can be achieved by using a floating bridge (in this case, it is also possible to lower the pitch of the note);
- *vibrato*: it consists of rapidly bending and releasing a string to rhythmically vary the pitch. It can be done with the fingers of the fretting hand or with the lever of a floating bridge;

¹a video in which I present the techniques cited above is available on the Sound and Music Computing (SMC) 2024 conference website: https://smcnetwork.org/smc2024/videos/SMC2024_video_id54.mp4, last accessed: 2024/11/22.

- *legato slide*: this technique involves sliding a finger along the string to a different fret between two notes, without picking the string to play the second one.

With regard to the physical and biomechanical constraints that arise during performance, at least the following must be taken into account (as will be seen in more detail in Chapter 3):

- the time required to move the fretting hand along (change of position) or across (change of string) the neck between contiguous notes;
- the spread between the fingers playing adjacent notes and, in general, the opening of the fretting hand in relation to the index finger.

1.4 Understanding performative expressiveness in contemporary popular music: interviews with top producers and musicians

1.4.1 Profiles of the interviewees

Given the substantial absence of systematic studies on expressiveness in popular music performances, it was decided to conduct a series of interviews with musicians, producers, and disseminators of primary importance on a global level on the topic under investigation. From these interviews, the expressive traits to be taken into account in the development of the computational model become apparent. The themes and reflections addressed also made it possible to assess, on the basis of a solid general framework, the relevance and potential role of existing research work on computational expressive performance, but aimed at different stylistic domains or musical instruments.

Before presenting what emerged from the interviews and was relevant to the realization of the computational model ², the interviewees will be introduced briefly:

Michael Arrom

Keyboardist, producer, educator. Adjunct instructor at USC Thornton School of Music, he is an in-demand session player and touring musician. He worked with artists such as Steve Vai, Paul Jackson, Jr, Diane Warren, Hunter Hayes, Sofia Carson, Madison Beer, and Noah Cyrus. His work as a musician or producer has been heard on CBS, NBC, ABC, Fox, and HBO networks.

²A more in-depth account of the interviews, also involving topics not strictly related to the development of the computational model that is the subject of this thesis, is being reviewed in the Journal of the International Association for the Study of Popular Music (IASPM), see page 149.

Hayden Maringer

Guitarist, producer, engineer. As a guitarist, he played with artists such as Jennifer Lopez, Demi Lovato, The Band Perry, TLC, and many more. He composed and produced music heard in *The Titan Games* (NBC), *World of Dance* (NBC), *The Scorpion King: Quest for Power* (Universal), the 2021 Tokyo Olympics, the 2023 MLB World Series, and Thursday Night Football (these last three with his band, Evaride). He performed on television programs such as the American Country Music Awards, Billboard Awards, American Idol, and Late Night with Jimmy Fallon. He served as music director on world tours of various artists, including Bebe Rexha.

Dennis Moody

Producer, engineer. His main stylistic areas are jazz, pop/rock, R&B and soul, world, and Motown. Among many others, he worked with Steve Gadd, Mike Stern, the Yellowjackets, Aerosmith, Journey, Iggy Pop, Paul Anka, Freddie Washington, Diana Ross, and Sly Stone. He also worked on film soundtracks. As a live concert engineer, he has mixed in the most important American and European venues. He published two books about drums recording and studio building.

Bobby Owsinski

Producer, engineer, disseminator. He worked with recording artists and on commercial, television, and motion picture projects. He took part in surround and DVD productions for artists such as Jimi Hendrix, the Who, Neil Young, Iron Maiden, and the Ramones. He is a leading popularizer in the field of contemporary music production, with twenty-five published books, journalistic interventions and the landmark podcast *Inner Circle* to his credit.

Susan Rogers

Producer, engineer, educator. She worked as a staff engineer for Prince between 1983 and 1988. Other credits include diverse artists such as David Byrne, Robben Ford, Jeff Black, Tricky, Michael Penn, Toad the Wet Sprocket, and Tevin Campbell. She holds a doctorate in psychology and is now the director of the Berklee Music Perception and Cognition Laboratory. Her book *This is What It Sounds Like: What the Music You Love Says About You*, co-written with Ogi Ogas, was published by W. W. Norton in 2022.

Mike Senior

Recording and mixing engineer, disseminator. He was in-house engineer at Great Linford Manor Studios (Milton Keynes, England), and worked with Wet Wet Wet, The Charlatans,

Reef, Therapy, and Nigel Kennedy. Since 1999, he has been part of the editorial department of Sound On Sound magazine. He published reference books in the fields of audio recording and mixing. He founded Cambridge Music Technology, offering music production services, studio training, and music-technology educational resources.

Ed Stasium

Producer and engineer. He worked, among many others, with the Ramones, Talking Heads, Mick Jagger, Living Colour, and the Empty Hearts. He also worked on movie soundtracks such as Rock 'n' Roll High School, Parasite, and Scarred. He was part of the staff of Venture Sound Studios (Somerville, New Jersey, U.S.A.), Le Studio Morin Heights (Quebec, Canada), and Power Station Studios (New York, U.S.A.).

Daniele Vantaggio

Producer, sound designer, sound engineer, educator. Also known as Seismal D, he produces melodic techno with major labels including Armada, EMI, mau5trap, and in official collaborations with deadmau5. As a producer, he works with international artists and DJs on millions of streaming tracks. He has collaborated on film music productions for SKY and RAI (the Italian public radio and television company) Cinema. He's a brand ambassador for several audio hardware and software manufacturers, including Steinberg and Arturia. He's a Pearson certified trainer and teaches sound design and music production.

Carl Verheyen

Guitarist, singer, educator. Best known for being the guitarist of Supertramp, he is an esteemed solo artist and an in-demand session player. His credits include David Benoit, Bee Gees, Chad Wackerman, Dolly Parton, Victor Feldman, Richard Elliot, and Stanley Clarke. He also played guitar on film soundtracks and TV shows. He has been an adjunct instructor of jazz guitar at the USC Thornton School of Music, has taught numerous seminars at the Musicians Institute (MI) in Hollywood, California (U.S.A.), and published two instructional books.

1.4.2 Toward a definition of the nature of performative expressiveness in popular music

All the interviews started with a general question about what performative expressiveness means in popular music. Technically speaking, the interviewees emphasized two key aspects that are easy to measure: the timing of notes' onsets, especially in relation to the beat, and the changes in intensity, or dynamics. The reason why these parameters are modulated varies depending on

the respondent. Some emphasized the romantic vision of self-expression (Rogers, Stasium, Verheyen, Maringer, Moody), while others highlighted two particularly interesting themes: stylistic coherence (Owsinski), and the interaction between band members (Rogers).

In the former regard, Bobby Owsinski observed:

[...] a jazz feel would be different than a blues feel, which would be different than a rock feel, which would be different than a pop feel. And most of that has to do (with) where the emphasis is placed on the beat. So for instance, in jazz, the emphasis is on top of the beat; it's a little ahead or right on it. In blues, it's behind, and sometimes very behind (the beat). In rock it's behind, and in pop it's closer to right on top of the beat. So that's a big part of it right there. So as you're learning in that particular idiom, then you're learning where to place the beat as well. [...] we're talking anywhere between two and ten milliseconds, depending on the tempo.

Susan Rogers emphasized the gestural dimension of performative musical expressiveness. Talking about that and musicians' interaction, she said:

Musicians tend to have an innate awareness of how their gestures fit into the whole composition and how they might adjust their expressivity to help the listener pay attention to the most important elements. What the most important element is totally depends on the style of music. [...] If a performer is consistently ahead of the beat, consistently, let's say it's rhythm guitar out in front of the beat, what that's doing is it allows that snare to be a little bit behind the beat. So there's elasticity, the push of the rhythm guitar, the pull of the snare, and it's in the orders of milliseconds, but still it stretches time a little bit. And this lets the singer either be really excited out in front there, in front of that snare drum or hang back with the snare drum and let that rhythm guitar go forward. When they're making a record in the recording studio, producers are in the control room listening for that, and they might suggest to the performers "can you put your snare a little bit behind the beat? Push your high hat, [...] pull back on the snare, lay back on that ride cymbal. Guitarist, a little bit in front of the beat, in front of the beat. Bass player, be tight, tight, tight, tight, tight with kicking. [...] So we're listening for the whole expression and where our spotlight of attention goes. That's why musicians employ voice leading, (it) is to help direct the spotlight of attention to the main melody line. So they deliberately make those gestures be out in front of the other gestures. Yes, it feels just a little bit rushed, and by a little bit, I'm talking about milliseconds. Now, if that happens erratically, you say that's a bad drummer. It's not a good group. But if it happens consistently, if that drummer can do that on every measure, every beat, that's when you've got a great drummer.

Mike Senior and Michael Arrom observed how performative expressiveness is able to make the playing of the specific musician unique and recognizable.

1.4.3 Verbal indications of the expressive intention

During the interviews, we also discussed the expressive verbal indications that are most commonly given by artistic producers to musicians in the recording studio, a situation that can be used as a model for the user/virtual performer relationship to be implemented in computational expressive systems. It emerged that, most often, such indications have to do with energy or - again - positioning relative to the beat. The first category includes “dig in” (Rogers, Owsinski - meaning with more energy, more aggressive), “relax” (Owsinski - with less energy, basically the opposite of “dig in”), “burning/smoking/with fire” (Moody - meaning very energetic). Regarding timing, it emerged in particular “lay back” (Arrom, Owsinski, Verheyen, Maringer - playing a little behind the beat). Other indications were “airily” (Senior - giving a bit more breath, making the sound a little bit lighter), “bouncy” (Senior - kind of rhythmically staccato or happy), “darker”, “lighter” (Verheyen), “funky”, “driving”, “swing”, “sparse”, “thick” (Arrom), “delicate”, “thin”, “heavy” (Vantaggio).

It is interesting to observe how, despite the fact that the interviewees showed a general consensus that the emotional terms commonly used in the classical sphere to describe performative expressiveness (cheerful, sad, melancholic, etc.) can also be employed in the popular sphere, there is in fact no trace of them among the verbal indications most commonly used, at least in their opinion and experience. As already noted, time positioning in relation to the beat and general dynamics (forte/piano) are of particular importance. Some indications, such as “airily”, “lighter”, “darker”, “thicker”, and “thin”, seem to refer to a timbre dimension that can only partially be traced back to the actual performance (equally - if not more - important in this respect are the instrumentation used and its settings). In contrast, indications such as “sparse” or “bouncy” appear more related to the compositional dimension than to the performative one.

1.4.4 Non-verbal indications of the expressive intention

Respondents were also asked whether or not non-verbal (or indirect verbal) indicators are commonly used in music production to communicate the desired expressive intention to the performer. The relationship between visual, auditory, or tactile perceptions and musical expressiveness is currently the subject of scientific study (see, e.g., [28], [29]). The answers indicated that one of the most common ways to evoke a specific expressive feel in a musical performance is by describing a particular context or showing (potentially moving) pictures. Susan Rogers talked about “painting a picture for a story”. This story might involve the weather,

such as a sunny day or ominous clouds gathering. Colors can also represent different expressive approaches. For example, Bobby Owsinski uses “make it more red” to mean “make it more intense,” while blue indicates to him the opposite. Carl Verheyen recalled being directed to “play more orange” or “make it greener.” Ed Stasium recounted Dale Ashby describing Laura Nyro’s use of colors to convey her musical vision, saying things like “make this more purple” or “more blue,” which sometimes made her intentions unclear. As Mike Senior pointed out, using words of this kind to describe music can be problematic, because listeners may interpret them differently, leading to a lack of consensus about the associations between colors and expressive intentions.

Hayden Maringer noted that specific expressive intentions are often communicated nonverbally through voice, humming, or “beatboxing,” and can also arise from real-time interactions between musicians, who guide each other through their instruments. This observation will be particularly relevant in Chapter 6. Also geometrical shapes were mentioned by respondents as possible unique and highly personal communication tools, though these may not always clearly convey the intended expression. Generally speaking, if, on the one hand, as Susan Rogers highlighted, the auditory experience cannot be fully understood in isolation from other senses, on the other hand, trying to convey expressive intentions through non-verbal or indirect verbal means appears to be potentially problematic and subject to misinterpretation.

1.4.5 Intentional and unintentional expressiveness

Finally, in the interviews, a theme never emerged that I consider fundamental, namely the distinction between intentional expressiveness and the introduction of performative deviations due to the biomechanical and psycho-perceptive limitations of the musician. Moreover, it seems reasonable to assume that these limitations of the musician may also contribute to his or her characteristic expressive traits. Even in the scientific literature on computational performative musical expressiveness, as will be seen in the next section, this issue is rarely considered.

1.5 An historical introduction and the state-of-the-art in Computational Expressive Music Performance

Since the 1980s, there has been growing interest in Expressive Music Performance (EMP) and its computational modeling. This research field has two main objectives: understanding human musical interpretation and automatically generating expressive performances. Various approaches have been proposed, including rule-based, statistical, Machine Learning, and Deep Learning methods, most of which focus on classical music, especially piano pieces. The

particular attention paid to the sphere of classical music can certainly be explained, on the one hand, by a general cultural bias towards this context. Perhaps even more relevant, as seen earlier, is the separation, typical of cultured music, between composer and performer. Sections 1.5.1 to 1.5.6 review the existing scientific literature on computational musical expressivity deemed relevant to the object of study of this thesis³. The materials have been organized and categorized not on the basis of the technology used (an excellent review article that moves in that direction is [2]) but on the basis of the high-level objective(s) that the existing computational systems purport to model. The macro-areas into which the reviewed scientific literature has been divided are:

- relation between expressiveness and structure of the musical piece;
- local expressiveness;
- relationship between emotional intention and expressive parameters;
- relationship between sensorial experiences and expressive parameters;
- identification and modeling of the performative styles of real musicians;
- identification of physical and psychological limits of the performer;
- conductor systems.

This subdivision is functional to enhance general clarity and understanding, even though the musical parameters being examined, such as timing, dynamics, and timbre, may be relevant across all areas. The difference lies in the rationale guiding each approach to study or modify these specific parameters.

1.5.1 Relation between expressiveness and structure of the musical piece

In academic reflection on music, structural analysis has historically held particular relevance. A broad definition of it is that proposed by I. D. Bent [30]:

[...] analysis may be said to include the interpretation of structures in music, together with their resolution into relatively simpler constituent elements, and the investigation of the relevant functions of those elements.

Music can be analyzed in terms of its form or structure across various levels of abstraction. At the highest level, this includes the sections of a piece, such as those in the traditional sonata form, which is divided into introduction, exposition, development, recapitulation, and coda. In popular music, we can have sections like introduction, verse, bridge, and chorus. On a

³A more in-depth treatment of the topic, including related scientific literature that is not fully relevant to the objectives of this thesis, has been published in the journal *Multimodal Technologies and Interaction*, see page 149.

more detailed level, we can identify structural elements - moving from the simplest to the most complex - such as motifs [31], phrases [32], and periods [33].

This is as valid in classical as in popular music (albeit with due differences).

In research about (computational) performative expressiveness in classical music, quite often the clarification of the piece structure has been considered one of the determining factors [34]–[36]. In fact, numerous studies in the literature focus on modeling expressive parameters based on structural analysis.

N.P.M. Todd, a key scholar in the field, examined how tempo variations relate to phrase structure in tonal music [37]. In his approach, the piece is divided into nested, hierarchically organized time spans. The model represents the structure of the piece by slowing down at structural endings, with the degree of slowing varying according to the hierarchical significance of the syntactic break. In later works, Todd incorporated the computational modeling of rubato in relation to music structure, and dynamics, based on the assumption that there is a correlation between speed and intensity (the faster, the louder) [38], [39]. Despite subsequent publications highlighting significant limitations of Todd’s model [40], [41], it remains a cornerstone in the development of research on computational musical expressiveness.

Other significant works that have based performative expressiveness on the piece structure are those of G. Grindlay and D. Helmbold [42] (in which a Hierarchical Hidden Markov Model - HHMM - was used to extract statistical data about the relation between the piece structure and the associated performances, with particular regard to time deviations), and of G. Widmer and A. Tobudic [43], [44] (rule-based). All of these studies distinguish between phrase-level and note-level expressive patterns that, combined together, can generate predictions about complex expression curves for new pieces.

Another approach to *structure-based* expressiveness is that relying on the construction and resolution of tension [12], [45]. C. Cancino-Chacón and M. Grachten presented a computational study examining the role of tonal tension in predicting expressive performances of classical piano music, using a bidirectional Long Short-Term Memory (LSTM) Recurrent Neural Network [46]. To compute the dynamics of tonal tension they relied on the previous work by D. Herremans and E. Chew, who used pitch classes, chords, and keys to derive variations in tension during the piece of music [47].

A particular case of use of E. Narmour’s implication/realization model [45] is that described by T. Mizutani and S. Sasaki, where it is exploited to analyze the relationship between musical structure and expressive dimension, in the broader context of a musical style transfer system based on Deep Learning [48].

One of the most significant lines of research in CEMP is the KTH rule system for musical performance [49]. This system is largely based on the *analysis-by-synthesis* approach, in which

the rules are obtained from dialogue with human experts [50]. Additionally, the system employs a complementary *analysis-by-measure* approach, where rules are created based on objective data retrieved from actual performances. These weighted rules influence various parameters, including timing, dynamics, and articulation. They are categorized into Phrasing, Micro-level timing, Metrical patterns and grooves, Articulation, Tonal tension, Intonation, Ensemble timing, and Performance noise, thus covering different levels of the musical piece and the performance. Specifically, regarding the structural dimension, the KTH system proposes rules for creating phrase arch-like tempo and loudness contours, as well as for inserting a *ritardando* at the end of the piece.

Examples of computational models based completely or partially on the structure of the piece could continue. The reader is invited to refer to [51] for a more in-depth analysis.

In the interviews reported in Section 1.4, however, a particular link between song structure and expressive performance in popular music never emerged. This could be due to a widespread tendency not to rationally elaborate an expressive component that may not be necessarily absent, but it seems much more likely that, as popular music is almost always played and recorded in group contexts, and increasingly on click, the game of expressiveness is played more at the level of local micro-deviations than of high-level formal elements (it is certainly more feasible to mark these through agogic and dynamic variations in the context of, say, a solo piano piece, unencumbered by other musicians and the click). The structural dimension was, therefore, not taken into account in the development of the expressive model described in the following chapters. Nevertheless, it was considered useful to present the topic here as it is relevant to Chapter 2, and may also be significant for possible future developments of the model.

1.5.2 Local expressiveness

Musical expressiveness can be observed and modeled not only on a broad scale but also locally, note-wise, or in relation to small groupings of notes. This local perspective allows for a detailed analysis of how individual notes and small clusters of notes contribute to the overall expressiveness of a musical performance.

The seminal studies conducted by A. Gabrielsson, among many others, are fundamental to this perspective [50], [52]–[54]. In Gabrielsson’s approach, an analysis of real performances, with a meticulous examination of relevant performance variables, makes it possible to detect systematic variations (SYVARs) that are related to some type of normative standard. These variations, or deviations from mechanical regularity, can vary according to the specific context, such as the type of music, the performer, or even the specific piece being performed. Gabrielsson’s investigations delved into rhythmic micro-structures, such as sequences involving a half note followed by a quarter note, or a dotted chrome followed by a sixteenth note,

among other patterns. He also explored deviations at half-measure and full-measure levels, identifying significant regularities, at least within the same musical context. These systematic variations highlight how performers introduce subtle deviations to convey expressiveness, which are often perceived as natural or stylistically appropriate by listeners. Furthermore, Gabrielsson’s research has shown that these deviations are not random but rather systematic and context-dependent. For example, in classical music, a performer might consistently lengthen the first note of a phrase to emphasize its importance, while in jazz, a performer might play behind the beat to create a laid-back feel. These performance practices underscore the importance of understanding musical expressiveness as a nuanced and context-sensitive phenomenon.

Gabrielsson’s observations seem to be particularly relevant in the case of popular music. Significant, for instance, is the reminder of the importance of positioning with respect to the beat in certain stylistic contexts, an argument which also emerged strongly in Section 1.4. The tendency to model the onsets of notes forming specific short rhythmic structures consistently across their various occurrences also offers a relevant field of investigation. At the current stage of development of the model that is the subject of this thesis, as will be seen in the following Chapters, the local variations of the expressive parameters are derived from the biomechanical component and intentional expressive choices, but there is nothing to prevent the system from being expanded in the future to include a systematic analysis closer to that proposed by Gabrielsson, thanks also to the dataset described in Section 5.2 and the larger dataset under construction described among the publications under preparation, at page 150.

Many of the rules of the KTH system [49] also address the local expressive micro-pattern area. For example, the duration contrast rule states “Shorten relatively short notes and lengthen relatively long notes”, while the double duration rule says “Decrease duration ratio for two notes with a nominal value of 2:1”.

Shifting the focus to data-driven studies, it appears that these are well suited to the computational calculation of expressiveness on a local scale while offering little robustness over extended time spans. By the way, the problem is relevant not only in the field of computational expressiveness, but also in the related field of automatic composition [55], also briefly touched by this thesis in Section 3.4.

K. Teramura et al. [56] tried a Gaussian Process (GP) regression approach, incorporating a Bayesian Committee Machine. The model learns the relationship between musical scores and actual performance data using a range of input features:

- position features: these include the musical meter, whether the note is on the first beat, and whether it’s in the highest (melody) part;
- pitch features: these describe the relative pitch of the note, differences in pitch from previous and subsequent notes, and average pitch comparisons;

- duration features: these detail the note’s duration, and ratios of durations with preceding and following notes, as well as average duration of the notes in the previous and following bars divided by the average duration of the notes in the target one;
- dynamics features: these capture the dynamics at the measure level and for the individual target notes.

The model showed the ability to predict dynamics, attack time, and release time of the notes during the performances and is of great relevance to this thesis.

A relatively similar approach is that of the YQX system, by S. Flossmann, M. Grachten, and G. Widmer [57]. This employs a Bayesian model trained on a corpus of human piano performances to learn and predict expressive elements like timing, dynamics, and articulation. Input parameters include features such as rhythmic context, duration ratio, pitch interval, and melodic contour, based on E. Narmour’s implication-realization model [45]. The results indicate that YQX can produce performances that are musically coherent and comparable to the performance of an average student (although far inferior to the performances of first-class pianists).

Okumura et al. proposed a method for modeling deviations from the musical score using binary tree-based clustering to classify the context of the notes [58], [59]. The parameters taken into consideration include notes’ pitch, duration value, rhythmic position, onset, offset, gate time (defined as “*The quotient of the time taken to depress the key in the performance divided by its length on the score. If both are exactly the same, the deviation of gate time is 1*”, p. 534), dynamics, and tempo. Although these are not fully overlapping approaches, as will be seen in Chapter 5, a tree-based Machine Learning approach was also used to develop the model that is the subject of this thesis.

Moulieras and Pachet explored the use of Maximum Entropy models, built to capture and mimic both unary and pairwise correlations between variables, to generate expressive performances [60]. The input parameters include metrical position, onset deviation, duration deviation, and loudness of the notes. The models were trained on a dataset of jazz, pop, and Latin jazz performances by a professional pianist. Quantitative validation showed that the generated sequences closely matched the reference performances, especially for styles like swing, with high predictive accuracy. A listening test revealed that people generally preferred the virtual performances generated by the MaxEnt model over those with no expression or random expression, and sometimes rated them almost as highly as human performances. This study is particularly relevant for the purposes of this thesis, as it emphasizes that, in the reference styles mentioned, performative expressiveness is to be sought more at the level of local textures than over extended arcs of time. It also shows a certain dependence between the modulation of expressive parameters and musical genre.

S. Oore et al. used an LSTM-based recurrent neural network, trained on a dataset comprising 1400 classical piano performances recorded in MIDI format, to predict both the notes and their expressive nuances, thus integrating composition and performance generation [61]. The model considers note timing, duration, and dynamics (velocity). The generated performances were generally well-received and noted for their realistic expressiveness and musical coherence. Despite some limitations in the long-term structure, the model excelled in producing performances with strong local expressive qualities, often perceived as human-like. Once again, a data-driven solution showed better results on a local basis, more than on long-term structures. An LSTM-based approach was also tested in the development of the model object of this thesis, as described in Chapter 5.

1.5.3 Relationship between emotional intention and expressive parameters

The emotional dimension as a determining component of expressive musical performance in popular music did not emerge significantly during the interviews summarized in Section 1.4. In particular, no terms attributable to the emotional context were mentioned among those most commonly used in the recording studio to communicate the expressive intention to the performer. In the expressive model described in the following chapters, the emotional dimension was therefore not taken into account. Nonetheless, most of the interviewees agreed that emotional categories can be profitably used to describe popular music pieces, understood as the combination of composition/arrangement and performance/production and post-production. Moreover, following further investigation, the emotional component could represent a reference for possible future developments. It was therefore considered pertinent to include a section devoted to this side of CEMP as well.

The relevance of the emotional component in determining people's interest in music appears to be dominant [62]. The research field that investigates the relationship between music and emotion attracted particular interest in recent decades. Excellent general introductions to the subject are those by P. N. Juslin and P. Laukka [63], and by T. Eerola and J. K. Vuoskoski [64].

The first distinction to be made is between emotions expressed by the music and identified by the listener, and music-induced emotions, which are felt by the listener. According to A. Gabrielsson, there is not necessarily a positive correlation between perceived and felt emotions in music [65]. This indicates that what a listener perceives as the emotional content of a piece of music may not correspond with what they actually feel while listening. Also Juslin observed that the emotions perceived and expressed in music can differ, adding that music does not always evoke an emotional response in the listener [62].

P. Evans and E. Schubert further examined the relationships between the emotional qualities

attributed to musical materials, referred to as expressed emotion or *external locus of emotion*, and the subjective emotional response to music, referred to as felt emotion or *internal locus*. They argue that the simple hypothesis of equality between these two types of emotions is overly simplistic, suggesting that the relationship is more complex than previously assumed [66].

According to T. Eerola [64], there are four main models of representation and understanding of the relationship between music and emotions, which are:

- discrete model: this model suggests that all emotions can be derived from a limited set of basic emotions, typically including fear, anger, disgust, sadness, and happiness;
- dimensional model: often traced back to J. Russell's circumplex model [67], still relevant today. It represents emotions as a mixture of the core dimensions of valence and arousal in a bi-dimensional space;
- miscellaneous models: these are based on a collection of concepts such as preference, similarity, and tension;
- music-specific models: these focus on emotions directly relevant to music, contrasting with the other more general-purpose approaches that may not be fully suited to the music field.

Numerous music features have been linked in past studies to expressions of discrete emotions. For example, fast tempo, major mode, simple and consonant harmony, and ascending pitch are positively correlated with happiness [62]. A. Gabrielsson published seminal papers on the relationship between parameters such as timing, dynamics, intonation, and expressed emotion [68], [69].

S.R. Livingstone et al. presented CMERS (Computational Emotion Rule System), a rule-based system for real-time control of musical emotion by modifying score and performance features such as pitch, loudness, and articulation [70].

R. Bresin and A. Friberg explored the connection between emotion and music by analyzing seven musical variables: tempo, sound level, articulation, phrasing, register, timbre, and attack speed [71]. In the experiments, participants used a gesture controller to adjust these variables in real time to express different emotions (neutral, happy, scary, peaceful, sad) across four musical scores. The findings revealed that participants adjusted the musical variables in ways consistent with previous research, identifying specific mean values and ranges for each emotion, regardless of the musical score.

T. Eerola et al. examined how primary musical cues contribute to emotional expression in music. The investigated variables were tempo, sound level, articulation, phrasing, register, and timbre. Using stimuli created with the Director Musices software (a software based on the

KTH System [49], [72] and the Vienna Symphonic Library⁴), the participants were asked to rate the emotional content of various musical excerpts. The findings demonstrate the significant influence of these musical variables on emotional communication, revealing how they can be systematically adjusted to evoke distinct emotional responses, and shedding light on the intricate interplay between these elements in music.

As anticipated, these studies could provide the basis for future extensions of the computational model presented in this thesis.

1.5.4 Relationship between sensorial experiences and expressive parameters

Another possible observation point of performative musical expressiveness is that which uses physical sensorial perceptions as the reference dimension. S. Canazza et al. [73] recorded multiple distinct renditions of a passage from W.A. Mozart’s Clarinet Concerto K622, asking the performer to play it once in a scholastic, *normal* way, and the other times trying to express the adjectives *light*, *heavy*, *soft*, *hard*, *bright*, and *dark*. Specific sonological parameters were measured, focusing on aspects such as timing (linked to changes in amplitude and duration) and frequency (timbre variations). The primary objective was to determine which parameters, and how, are modified when the performers alter their expressive approach. The analysis demonstrated how performers adapt timbre and timing parameters to reflect their expressive intentions. A synthesized performance, based on data previously collected in the real performances, validated the effectiveness of this method. This research laid the foundations for the subsequent CaRo system [74]–[77]), that will be addressed in more detail in Section 1.5.6.

Adjectives such as *hard*, *light*, *heavy*, or *soft* can also be found in the pDM system, a real-time sequencer integrated with the KTH rule system, with which a virtual performance can be created mapping these high-level indications to rules [78].

The association between sensory or physical experiences and expressive performance parameters could be of primary importance in the development of CEMP systems oriented toward popular music. Within pop music production teams, these kinds of linguistic parallels seem to be frequently used [79], and might be more easily interpreted by producers, composers and musicians and better convey expressive intentions than other terminology families, more established in the classical sphere. Analog or similar terms to those mentioned above also emerged in the interviews summarised in Section 1.4. The main performative parameter involved by most of these indicators seems to be dynamics. Restricting oneself to the actual performance (excluding instrumentation and gear settings from the analysis), also the timbre of the instrument can be traced in part to the dynamics. In the specific case of the guitar, timbre

⁴<https://www.vsl.co.at>, last accessed: 2024/11/22.

also depends on the choice of fingering and the performance techniques used, two parameters managed by the computational model and presented in Chapters 3 and 4.

1.5.5 Identification and modeling of the performative styles and of physical and psychological limits of real musicians

One of the aims of the computational model that is the subject of this thesis is to imitate, at least partially, the performative peculiarities of specific musicians. The scientific literature attributable to this field is, therefore, of particular relevance here. Specifically, the approach followed (as described in the following Chapters) is dual. On the one hand, unintentional variations in expressive parameters, likely to be attributable to biomechanical and psychic constraints or limitations, are modeled. To do this, a Machine Learning-based approach is employed, trained on a real-world performance dataset (see Chapter 5). On the other hand, intentional expressivity is then introduced, implemented in the model in the form of indications entered into the system by the user (see Chapter 6).

As L.L. Costalonga et al. noted [8], there has been limited research focused on the implications of biomechanical constraints and internal processes of musicians in the development of expressive music performance models. In their work, with a focus on guitar playing, they identified two primary approaches to modeling expressive performance: one that focuses on the outcome of the performance (*Psycho-Acoustics-Based modeling*) and another that delves into the internal processes of the performer (*Performer-Based modeling*). The study used a combination of ergonomic and biomechanical analysis to understand the physical demands on guitarists. This included measuring the force exerted by the fingers on the strings, which varied across different frets and strings due to factors like string tension and guitar setup. The research highlighted the importance of these physical aspects, showing that they significantly impact the quality and clarity of the notes produced. The research emphasized the often-overlooked importance of physical manipulation in musical performance. By examining the biomechanics of guitar playing, the authors demonstrated that these physical constraints were crucial in shaping the sound and expressiveness of the performance. The findings underscored the necessity of incorporating biomechanical considerations into developing expressive music performance models (see also L. Costalonga (2023) [80]). By doing so, researchers and developers could create more accurate and realistic representations of music performance, which could be beneficial for educational purposes, performance assessment, and the development of new musical technologies.

I believe that this position, valid in itself in any context, has a particularly pronounced importance in the field of popular music, where, as already observed, in most cases the performer's degrees of freedom are much more restricted than what happens in performances

of classical music for solo instrument.

The GERM model, developed by P. Juslin, A. Friberg, and A. Bresin [10] is another significant contribution in CEMP, and it also takes into consideration the constraints and limits of the performer. The model, in itself, seeks to integrate different approaches from the literature into a single computational framework. The acronym GERM stands for Generative rules, Emotional expression, Random variations, and Motion principles. The latter two components, random variations and motion principles, are particularly relevant to this discussion. The model acknowledges that human musical performances are inherently stochastic, with variations introduced by the limitations of human perceptual and motor skills. These include motor delay noise, which does not depend on the inter-onset interval (IOI) of notes (the time between the onsets of two adjacent notes), and time-keeper variance, which varies according to the IOIs. The model simulates random variations by applying white noise to the onset times and loudness of each note (motor delay noise) and using filtered white noise to achieve a $1/f$ frequency response for time-keeper noise, with variability based on the IOI. The model also considers physiological constraints that arise at the intersection of the human body and the physical demands of playing a musical instrument.

Although the goal is not generally the construction of expressive models, also studies on motor control, hand dexterity, and timing precision in musicians should not be overlooked [81]. These studies often aim to prevent injuries [82] or to analyze optimal fingerings from a biomechanical perspective [83], [84]. This last field will be explored extensively in Chapter 3.

Other work on modeling real musicians has focused on what L.L. Costalonga et al. described as *Psycho-Acoustics-Based modeling*. They are, therefore, less relevant to the current state of the expressive model presented here but could provide inspiration for future developments of the same, potentially following procedures similar to those described in Chapter 5. From this perspective, it is considered appropriate to cite at least - among others - the works linked to the KTH model ([49], [85], [86]). The contributions of E. Stamatatos and G. Widmer [87] and of R. Ramirez et al. [88], were dedicated, respectively, to the automatic identification of piano and saxophone performers. A. Tobudic and G. Widmer explored the use of relational instance-based learning (RIBL) to replicate the expressive styles of real pianists [89]. Using the DISTALL algorithm, they analyzed recordings of six pianists performing sections of Mozart's sonatas, focusing on beat-level timing and dynamics. They modeled these expressive traits with second-degree polynomials to capture the nuances of each pianist's style, including inherent random variations and motion principles. The study successfully demonstrated that the system could accurately replicate and identify the unique styles of pianists, highlighting the potential of RIBL in musical expression analysis and education. S.I. Giraldo and R. Ramirez developed a

machine learning system aimed at discovering performative rules and modeling expressive jazz guitar performances [90], [91].

1.5.6 Conductor systems

Solutions for computational music expressiveness that incorporate meaningful user interactions are known as conductor systems. In these systems, users can make adjustments to the expressive rendering of a musical piece either asynchronously or in real time. Two commonly emphasized aspects in conductor systems research include enabling users without specific musical training to engage with music, with a focus on creativity [77], and allowing users to concentrate on the expressive aspects of the music without being concerned with the technical challenges of playing musical instruments [92].

More generally, conductor systems take part of the definition of the expressive features of the virtual performance away from the automatic component of the model and return it to the human operator. This may or may not be desirable, depending on the scientific goals and intended uses of expressive models. With regard to the one presented in this thesis, it was decided to implement the automatic calculation of expressive deviations only with regard to the unintentional expressive component (given by psychic/perceptual and biomechanical limitations and constraints - see Chapter 5), and to intervene instead on deviations related to intentional expressivity on the basis of user input (see Chapter 6). This is to recreate, in a way, the real situation that occurs when a session man records in a studio under the guidance of a producer.

The association between conductor systems and traditional orchestra conducting inspired some of the earliest research in this area. In M.V. Mathews and J. Lawson's work, a radio baton capable of controlling expressive parameters of a synthesizer (timing and dynamics) through the dedicated software Conductor was realized [92]–[94]. Another example of work involving a virtual orchestra conducting system is that by E. Lee et al. [95], which is based on the recognition of baton gestures and associated time-stretching on the audiovisual recording of a performing orchestra. A more recent implementation of virtual orchestral conducting was proposed by T. Baba et al. [96]. J. Langner and W. Goebel described a system for the real-time representation of performances on a two-dimensional graph, where tempo (bpm) is plotted on the x-axis and loudness on the y-axis [97]. In this system, a dot moves synchronously with the sound along the two axes. The analyzed materials were piano performances played on a Bösendorfer computer-controlled grand piano (SE290). Timing information was extracted first from MIDI data and subsequently directly from audio, while loudness data were obtained from the audio recordings of the performances. The dot's trajectory, which gradually fades away leaving a visible tail, visually represents the two most important performance parameters: tempo and loudness. The term *performance worm* was coined for this kind of visualization [98], and it

has been referenced like that in other scientific publications (see [99]). In [100], S. Dixon et al. revisited the metaphor of the performance worm, inverting it to allow the user to control tempo and loudness in real time. This control can be achieved through hand movements using a digital theremin (air worm) or with a computer mouse (mouse worm). In the same study, the air tapper and mouse tapper systems were also introduced. These are based on the more traditional control of metronomic tempo, either through hand gestures or the use of a mouse.

Although these systems aim to directly control the agogics and dynamics of the musical flow, and thus do not fully coincide with the 'conducting' objectives of the model proposed in this thesis, they are of interest in relation to the study of the possible modes of interaction between user and machine, a theme that will be explored in more detail in Chapter 6).

A different and highly relevant take on conductor solutions was that of the CaRo system [74]–[76]. The concept is based on the idea that there are two primary sources of musical expression: the structure of the piece (see Section 1.5.1) and the expressive intention, associatable with sensorial experiences (e.g., bright, dark, hard; see Section 1.5.4). By asking professional musicians to perform the same melody in a neutral, scholastic manner and with a specific set of expressive intentions, it is possible to estimate how performance parameters (e.g., intensity, legato, attack duration, brightness) are influenced by the musician's expressive intention. For each performance parameter and expressive intention, two metrics were extracted: k (associated with the mean value) and m (representing the range of values, affecting variance). Factor loadings from factor analyses (using a two-dimensional solution) were then used as coordinates to describe the expressive parameters of performances in a two-dimensional space. By reversing the process, this two-dimensional space can be used to control the expressive intention of neutral performances. The CaRo system can operate on both audio signals and MIDI data. Years after its initial release, the CaRo system was adapted to function in a Web 2.0 environment [77]. The CaRo system is in some ways closer to the mode and logic of user interaction targeted by this thesis.

1.5.7 Contributions of this work to the literature

This thesis aims to make an original contribution to the study of computational expressiveness within a number of specific areas of interest, research and application:

- Section 1.4 proposes an analysis of the nature of performative musical expressiveness in popular music from the point of view of musicians and producers through a series of interviews with world-leading professionals. Such a reflection was not previously present in the literature;
- automatic fingering generation: as will be explained in detail in Chapter 3, I consider

automatic fingering generation an essential junction for the creation of expressive guitar models. Although the topic has been dealt with previously, a more in-depth and flexible approach is offered here than in previous solutions;

- use of expressive techniques modeling: as will be explained in detail in Chapter 4 the possibility in lead guitar of generating notes in many different ways, and of enriching them with expressive techniques, makes it necessary to model the performer's choices in this regard as well. This is a subject that has been dealt with in an extremely partial and marginal manner in the literature and which has instead been explored in depth in this work;
- creation of a specific dataset: as explained in detail in Chapter 5, it was necessary to create an original dataset of guitar performances without intentional expressiveness in order to carry out training and evaluation of predictive models;
- also in Chapter 5, a series of Machine and Deep Learning predictive solutions were compared to see which offered the best results in predicting expressive parameters;
- in Chapter 6, a reflection on the development of innovative solutions for user interaction in non-fully automatic expression models is proposed. In particular, the hypothesis of non-verbal voice control is introduced;
- finally, an end-to-end model is proposed for generating expressive MIDI Files from quantized MIDI melodies with fixed velocity, geared specifically to lead guitar.

Chapter 2

Mechanical music and performative expressiveness

Chapter introduction

The automatic reproduction of music encoded in symbolic form was not born with digital technology; instead, it has its roots in mechanical musical instruments, which have been widespread since the Middle Ages. This chapter presents a work dedicated to the automatic search, in perforated paper rolls for player piano, of passages intentionally made expressive in the manual manufacturing of the roll itself. Although the topic is not a constituent part of the expressive model for guitar, the investigation - carried out with computer engineering tools - is aimed at an area that can be considered an ancestor of modern systems for the automatic generation of expressiveness. The results proved to be of particular value and interest also from a musicological point of view.

2.1 Mechanical music

The study of computational musical expressiveness is strongly linked to the symbolic representation of music. This, in a digital context, requires encoding in specific formats, including, for example, MIDI (an acronym for Musical Instrument Digital Interface) [101], MusicXML [102], or tokens [103]. However, the realization of symbolic music representations intended for machines originated well before the digital era, with the media destined for mechanical musical instruments.

The first genuinely mechanical instruments, capable of playing music automatically without requiring continuous intervention from a musician, emerged and became widespread during the Middle Ages and the Renaissance [104]. Carillons, in particular, were of significant

importance. These instruments consist of a series of bells automatically played using a drum or cylinder mechanism equipped with pegs or pins, which lift hammers to strike the bells. The earliest known carillons date back to the 14th century and were predominantly developed in the Netherlands and Belgium. Music boxes, originating in the 18th century, produce melodies using a cylinder or disc to engage a series of metal tines. Mechanical street organs, which gained popularity from the late 18th to the early 20th century, are automated musical instruments used in public spaces like streets and fairs. They are commonly based on barrels with pins or cardboards, which control the airflow through pipes to produce sound. The end of the 19th century then saw the birth and spread of player pianos, whose operation is based on perforated paper rolls.

If, on the one hand, for obvious reasons, an automatic computational approach to the generation of expressiveness cannot be detected in this context, on the other hand, it appeared interesting to investigate at least whether, in the supports manually manufactured, there was an interest in some form of expressive rendering, or whether more simply an attempt was made to reproduce the content of the corresponding score as faithfully as possible (or whether, when a score was not present, durations and possibly intensities or other parameters were encoded in a rigid manner or not).

A case study of particular interest turned out to be that of the player piano. The so-called metronomic rolls (i.e. made at the desk, not based on an actual performance), commonly considered to be a mere translation of the reference score, in fact revealed an approach that was at least partially interested in expressive sound performance as well [105].

2.2 A brief history and operating principles of the player piano

Player pianos emerged as an evolution of earlier mechanical instruments, such as street organs (see Section 2.1). Unlike their predecessors, they used lighter, narrower paper rolls wrapped around a cylinder. These rolls featured holes that, when aligned with a device called *tracker bar*, allowed air to pass through and activate the piano's keys via a pneumatic system. The position and length of the holes determine the pitch, time position, and duration of the notes, effectively encoding these parameters [106]–[108] (see Figure 2.1).

There are two primary types of player piano rolls: metronomic and reproducing. Metronomic rolls are created basically by *translating* the musical score directly onto the paper perforations, ensuring a strict relationship between the score's notes and the holes' lengths. These rolls are often considered less expressive, as they are commonly believed to strictly follow the reference score [105]. In contrast, reproducing rolls capture the nuances of a pianist's live performance, including - to a certain degree - dynamics and tempo variations, providing a more

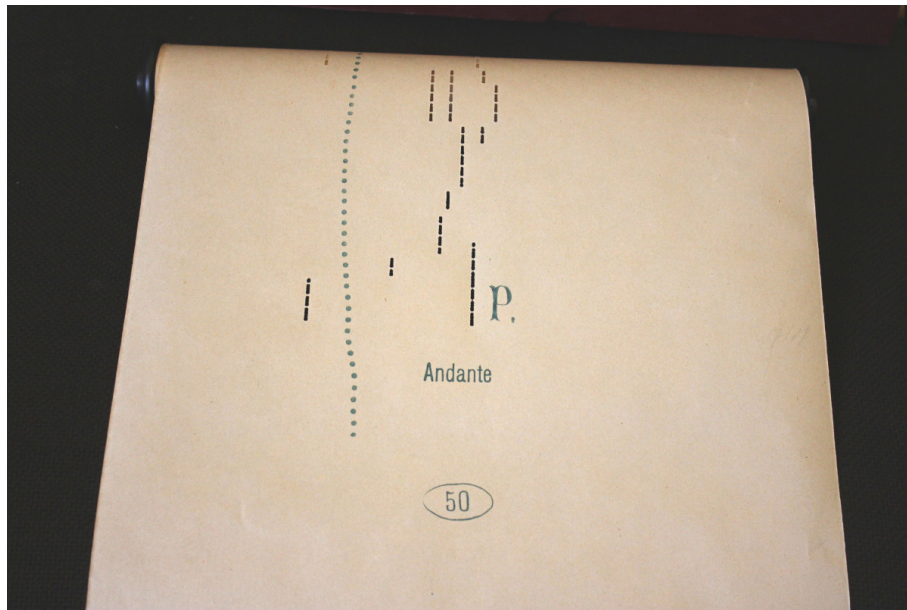


Figure 2.1: A roll for player piano. The circled number 50 indicates the expected roll flow speed of 50 tenths of a foot per minute.

lifelike rendition of the music [109].

2.3 Are metronomic rolls expressive?

The research work described below arose from the collaboration between the Centro di Sonologia Computazionale [110], [111] and the Department of Musicology and Cultural Heritage of the University of Pavia (Italy). The Pavia Department hosts one of the most relevant collections of piano rolls in Europe and the world (as of June 2024, there are over 8500 pieces, constantly growing)¹. A special roll scanner and software for converting scans into MIDI format (MIDI File, first version of the protocol ²) are also in use at this Department. Both were developed by AMMILab, the technology division of the Associazione Italiana Musica Meccanica³ (see Figure 2.2).

The question that was attempted to be answered, with the aid of computational tools, was: "were the metronomic rolls really an exact copy of the reference scores, or, at least in some cases, did the roll creators introduce variations for expressive purposes?". The dataset, the algorithm used, its implementation, and the results obtained are briefly described below. For more detailed information on the work, please refer to P. Bontempi, P. Zappalà, and S. Canazza (2024) [105].

¹<https://pianorolls.unipv.it>, last visited: June 17, 2024

²<https://midi.org/standard-midi-files-specification>, last visited: June 17, 2024

³<https://sites.google.com/view/ammi-lab/home>, last visited: June 17, 2024



Figure 2.2: The player piano rolls room of the Department of Musicology and Cultural Heritage. A player piano is visible on the left, and on the right is the roll scanner.

2.3.1 Dataset

A dataset consisting of around 700 rolls was extracted from the scanned rolls collection based on these criteria:

- the rolls were chosen from among those produced by the Italian manufacturers Aurora & Apollo, First (Fabbrica Italiana Rulli Traforati) and Fratelli Cigna;
- the rolls should have no obvious defects, nor should any problems have arisen during digitization and conversion to MIDI format;
- the rolls had to have a single flow speed indication, valid from the beginning to the end of the roll (see Figure 2.1);
- during the conversion to MIDI format of the roll, the operator should not have made any manual agogic changes (the MIDI conversion software allows, if wanted, for manual interventions on the agogic of the generated MIDI File).

The analysis and automatic search for expressive passages were performed on the Standard MIDI Files generated from the scans of the selected rolls. These MIDI Files follow the specifications of MIDI version 1, and are either type 0 or type 1. The MIDI File encoding software always uses the values of 500,000 and 96, respectively, for the *Set Tempo* meta message and the *Pulses Per Quarter Note* (PPQN) parameter. This implies that the temporal resolution (the time covered by 1 MIDI tick) is of about 5.2 ms ($500,000 \text{ ms} / 96$), which is adequate

to represent the roll and reveal possible expressive deviations. Since the value of SetTempo is fixed and unrelated to the speed at which the roll flows, it is not possible to deduce from it the real metronomic speed in Beats Per Minute (BPM) of the musical pieces. In the same way, it is not possible, even knowing the value of PPQN, to associate a certain number of MIDI ticks *a priori* with the musical duration values (crotchets, quavers, etc.). On a practical level, there is a correspondence between MIDI ticks and absolute time values, rather than having a correlation between MIDI ticks and symbolic musical duration values, as the MIDI standard should instead provide. This made the implementation of the automatic search algorithm for expressive passages slightly more complicated, as described in the following paragraphs.

2.3.2 Algorithms and implementations

Metronomic rolls codify in the holes the pitch and start/end of the notes, as well as - in some cases - the use of the sustain pedal and the accentuation of the right or left half of the keyboard (to balance melody and accompaniment; the split point is generally located around the E3 at 329 Hz). This information is unambiguously and automatically interpreted by the player piano. Besides, there is generally a line (continuous or dotted) printed on the roll, used by the performer as a reference to move during the performance a lever dedicated to the sound volume to be generated. Given the approximate nature of this dynamic indication, also linked to the fact that the line offers a general reference, not specific for each note, somehow similar to MIDI Control Changes (CC) 7 and 39, it was decided to focus attention on the timing of the notes, to evaluate whether in relation to it the rolls followed the associated scores precisely or whether a certain freedom of expressive deviation was instead foreseen.

Since note durations in music are encoded in proportional terms, and metronomic rolls are basically transpositions of musical scores, it is natural to expect a high number of occurrences of the Inter Onset Intervals (the time between the beginnings of two music events - IOIs) corresponding to the rhythmic values used in the piece (e.g. minims, crotchets, quavers and so on). In a quantized MIDI file, all the IOIs associated with the same rhythmic value would correspond exactly to the same number of MIDI ticks. This is not the case when MIDI conversions of player piano rolls are taken into consideration: rather than individual ones, ranges of values that are more or less wide depending on the case have to be taken into account here, due to the accuracy limitations inherent in the medium (Figure 2.3). Furthermore, the use of a fixed SetTempo value makes it impossible on this dataset to assume correspondences between the number of MIDI ticks and musical duration values before analyzing the actual content of the MIDI File.

The basic idea of the approach followed is that by identifying IOIs located outside these tolerance intervals but sufficiently close to one of the peaks corresponding to the duration values

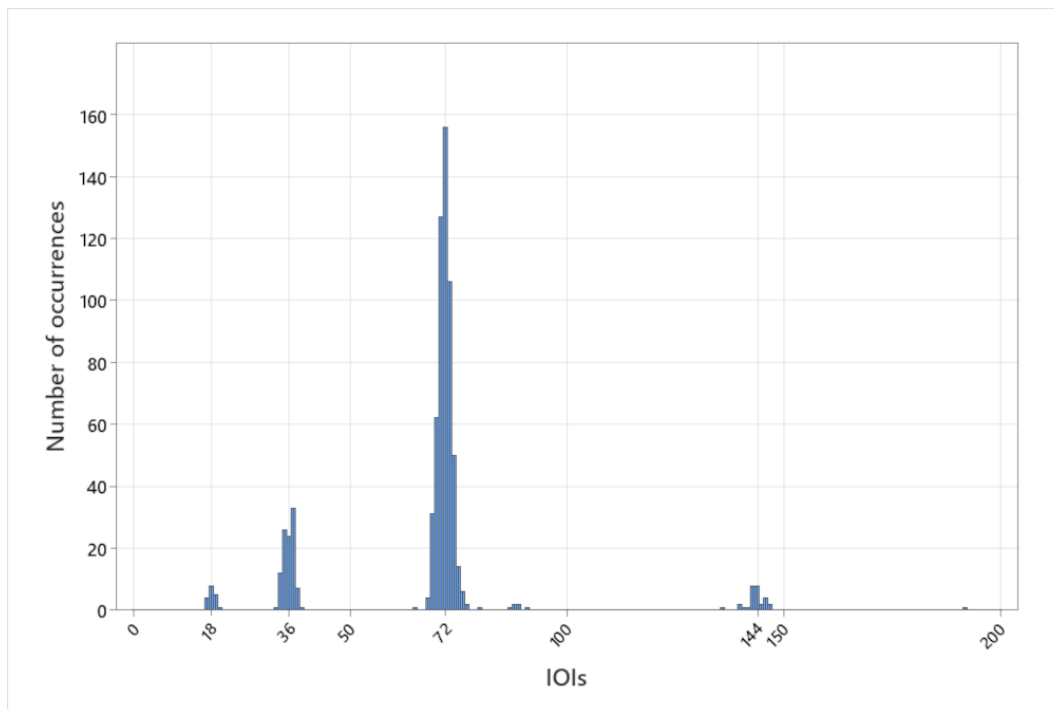


Figure 2.3: The distribution of the IOIs (in MIDI ticks) in the roll First 0115.

used in the song to potentially be expressive variations thereof, it becomes computationally possible to identify passages with a high probability of being expressive deviations of the score, intentionally inserted during the production of the roll.

At the implementation level, two different software solutions were developed to identify possible expressive passages (both declined in a user-customizable version dedicated to the analysis of single files and in a non-customizable one dedicated to mass processing), one based on unsupervised clustering, using the DBSCAN algorithm [112], and the other based on mathematical peaks recognition and analysis⁴.

It must be added that, although beat tracking algorithms can be applied to MIDI Files or their audio renderings to identify agogic variations, this approach was not considered the optimal solution for the area of study under consideration, which is why different solutions (introduced above and described in detail in the following paragraphs) were sought. The main reasons for not using beat-tracking algorithms to search for expressive passages are:

- even the most advanced beat tracking systems are susceptible to a significant degree of error [113];
- beat intervals in player piano rolls are subject to continuous fluctuations, which can

⁴All the original code described or cited in this chapter is available at https://github.com/pierluigi-bontempi/PhD_thesis/tree/main/Chapter_2_Mechanical_music_and_performative_expressiveness.

complicate the differentiation between intentional expressive agogic variations and unintended oscillations resulting from the inherent limitations and production practices of the medium;

- music transcribed for player piano rolls often features short notes used as appoggiaturas or ornaments, which can confuse the tracking algorithm;
- beat tracking systems may not provide detailed insights into what occurs within each beat, potentially overlooking subtle expressive timing deviations.

2.3.3 The DBSCAN-based solution

The first software developed was based on the well-known DBSCAN unsupervised clustering algorithm. This algorithm was chosen because:

- it can be used on one-dimension data (in this case, an ordered array of IOIs, measured in MIDI ticks);
- it is not constrained to a predetermined number of clusters;
- it is suitable for detecting outliers.

The programming language chosen was Python, with the use of the additional libraries Mido [114], NumPy, Pandas, scikit-learn, and the Python statistics module.

First, the MIDI file is parsed and information about each note's start and end times is extracted. The total duration of the notes is also calculated. When a note starts at a maximum distance of 6 MIDI ticks from the beginning of the next one, it is considered to be in sync with it. This way, simultaneous notes (chords, for example) are treated as single rhythmic events. It is also obviated the potential problem of correctly recognizing appoggiaturas or ornaments, which generally consist of short notes placed at a minimum temporal distance before the more structurally significant ones. This yields a linear sequence of relevant rhythmic events with associated IOIs.

Since it was not possible to identify fixed DBSCAN parameters that would give appreciable results over the data set used (of 50 MIDI Files randomly selected among the rolls scans that met the requirements described in Section 2.3.1), and since that even within the same roll the parameters often had to be changed to correctly identify different clusters, a recursive application of the algorithm was opted for, with parameters being changed at each iteration. The starting parameters (that allow for simple, but rarely sufficiently homogeneous, clustering) and subsequent recursive changes (potentially generating increasingly homogeneous clusters) to be made have been identified empirically. To be accepted, a cluster must comply with predefined parameters of standard deviation, range of values, and coefficient of variation, which, on the

one hand, guarantee internal cohesion but, on the other hand, allow for the variability of values not attributable to expressive intentions to be taken into account.

At each iteration, when the algorithm detects one or more valid clusters, it removes them from the dataframe. Given that, in some cases, it has been observed that shorter notes generate excessive numbers of outliers and that in no case were these outliers considered significant to listening, at each iteration of the algorithm the notes shorter than a threshold that gradually increases, up to a maximum of 28 MIDI ticks, are also removed. The process ends when, during an iteration, all IOIs are inserted into valid clusters or marked as outliers, or after a predetermined maximum number of iterations. Rhythmic events remaining within the dataframe are considered outliers, and a new iteration of the DBSCAN identifies areas with a high density of them, potentially affected by intentional expressive timing deviations. The flowchart of the algorithm, complete with detailed parameters, is represented in Figure 2.4.

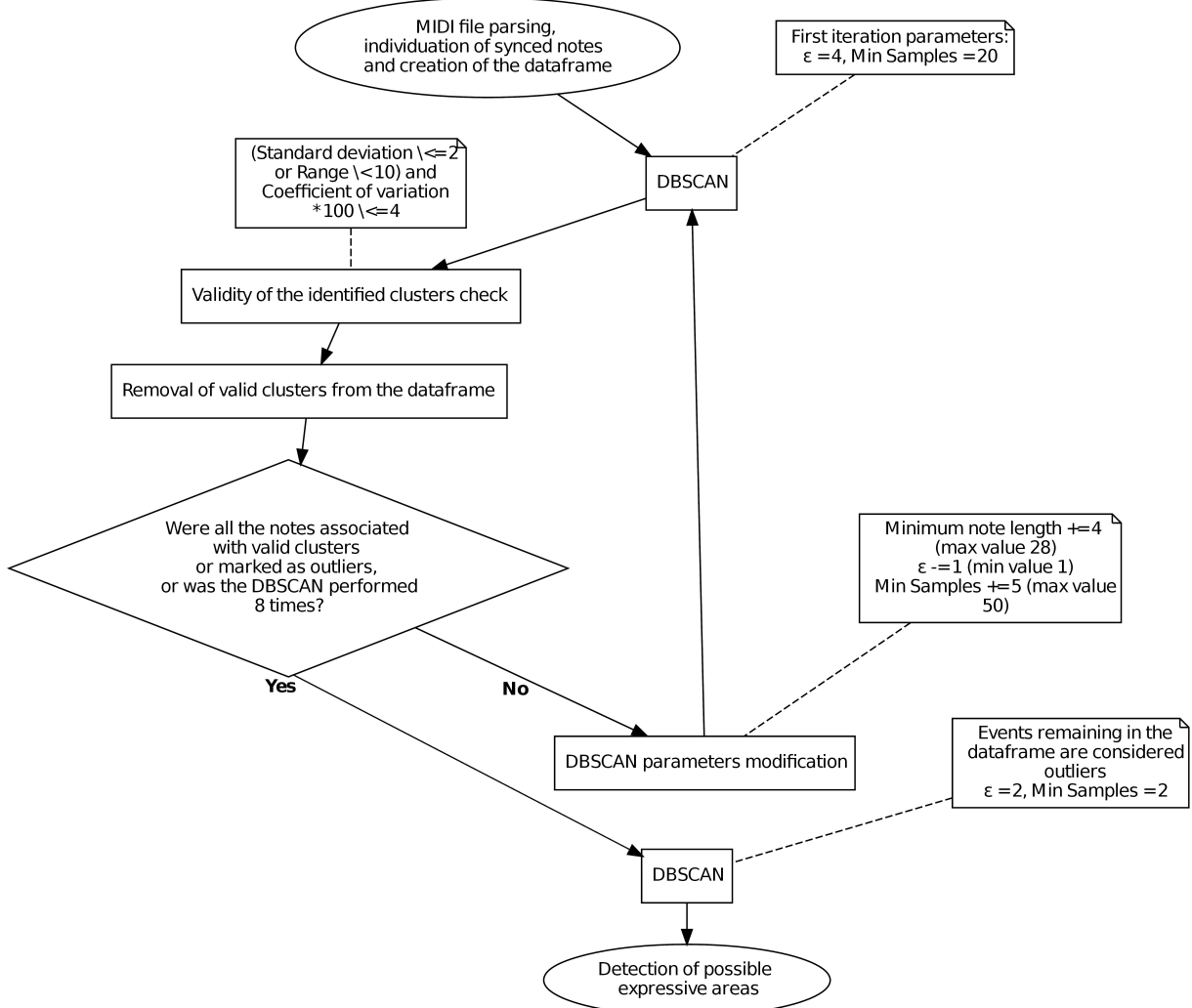


Figure 2.4: Flowchart describing the DBSCAN-based algorithm.

As observed above, this first software solution was tested on a subset of the overall dataset, consisting of 50 rolls. In eight cases, the software did not find any potentially expressive area. This was always confirmed by listening. In all the other 42 cases, the software highlighted at least one potential expressive area, but only in 6 cases the effective presence of apparently intentional expressive passages was confirmed by listening. In relation to the rolls judged expressive by listening, in 4 cases all the areas of interest were detected correctly (at least a portion of each expressive area was highlighted), while in the last two cases part of the expressive passages was identified, and part was missed.

While this first solution offered encouraging results, the high number of false positives prompted the evaluation of an algorithm capable of starting from the same idea, but not based simply on unsupervised clustering.

2.3.4 The mathematical peak detection and analysis solution

Since the DBSCAN-based implementation had proved sensitive but lacking in specificity, another approach was evaluated, which - although starting from the same basic idea - would allow a more precise classification of duration deviations from the peaks identified in the IOIs distribution. The software was written in Python, with the use of the libraries Mido, NumPy, Pandas, and the package for scientific computing SciPy.

In this second implementation (the one described in [105]), after arranging all IOIs in an ordered array (excluding the shortest rhythm events and those within a minimum distance from the following one, similarly to what happens in the DBSCAN implementation), significant peaks corresponding to the rhythmic values used in the piece are identified using SciPy's `signal.find_peaks` function. This operates on a one-dimensional array (in this case, the ordered collected IOIs), detecting local maxima by comparing adjacent values. In addition to the IOI array, specific parameters were set: *height* at 10, *threshold* at *None*, and *distance* at 5. These parameter values were determined through experimental testing on an initial dataset of 107 rolls, refining the settings until a configuration that worked well in most cases was established. At this point, *Virtual peaks* are added, representing twice the detected MIDI tick peaks values, as long as they do not fall within the range of existing peaks. This step ensures that the longer, less frequent notes, which are part of the "non-expressive" duration values, are also considered.

Each IOI is then given a score based on its distance in MIDI ticks from the nearest peak. Each peak governs an interval extending from half to one and a half times its MIDI tick value. Outside this interval, no score is calculated, as attributing the IOI to a deviation from the nearest peak would be highly questionable. However, experimental results indicate that only a residual number of IOIs fall into this category.

To reflect the varying significance of deviations based on the reference peak, the scores

are normalized by dividing them by the nearest peak's tick value. For ease of interpretation, the score is then multiplied by 100. For example, an IOI of 200 MIDI ticks that is 15 MIDI ticks away from the nearest detected peak of 185 MIDI ticks results in a score calculated as $\text{abs}(200-185)/185 \times 100 = 8$.

Potentially expressive areas are identified by analyzing the cumulative score across groupings of contiguous notes.

Similar to the peak detection function, the optimal values for score threshold and note grouping were determined experimentally based on the initial dataset. In most cases, threshold and grouping values of 80 and 3, respectively, yield good results. In some instances, the threshold needed to be lowered to better align with expressive passages. However, this adjustment was significant only when the detected peaks were numerous and closely spaced. This typically occurs when the roll is divided into multiple sections, each with its specific tempo (even if this is not explicitly indicated).

In the initial set of 107 rolls, upon listening, the authors had determined that 16 contained expressive passages, while 91 did not have any significant ones. Of these 91 rolls, the software correctly identified the absence of expressive passages in 48 cases but generated at least one false positive in the others (proving significantly less prone to false positives than the DBSCAN-based implementation). Out of the 40 expressive passages identified by listening, the software successfully detected 34, with 19 of these detections occurring with a score threshold of 80 and a grouping of 3, and the remaining 15 requiring a lower threshold.

The application of the batch version of the software to the rest of the dataset (for a total of around 700 rolls), with the threshold set at 80 and grouping at 3, allowed us to identify 24 additional expressive passages spread across 17 rolls quickly. In the batch version of the software, rolls that generated more than six peaks (excluding the *virtual* ones) were automatically excluded from the analysis to avoid data distorted by the probable presence of multiple musical sections at different tempos during the roll.

2.3.5 Final considerations

This study demonstrated that metronomic rolls for player piano are not simply mechanical transcriptions of musical scores. Instead, it has been shown that, in some cases, the perforations in the paper were made taking into account the performative expressiveness that would derive from the reproduction of the roll. These specific supports for mechanical musical instruments can, therefore, be considered the ancestors of modern research efforts in the direction of computational musical expressiveness, albeit with the obvious difference that in the piano rolls the expressiveness was inserted manually, and not through automatic systems. Resuming the discourse on the objectives that musical performative expressiveness may have, it should be

noted how, in the case of the rolls, the agogic variations detected have been almost entirely attributable to the intention of clarifying the structure of the piece of music 1.5.1, according to a logic similar to that described by N.P.M. Todd (concerning the slowing down to mark the closing of sections [37]). The findings of this study are also significant for the musicological investigation of historical performance practice since they demonstrate that metronomic rolls for player piano can indirectly provide valuable insights into the performance practices of the early twentieth century.

These studies on player piano rolls also led to a collaboration with the Padua Conservatory of Music in an important project of rediscovery and artistic reworking of musical pieces composed at the beginning of the 20th century by Gian Francesco Malipiero and preserved on a roll. This activity is described in detail among the activities for the Third Mission, at page 155.

Chapter 3

Modeling of fingering choices

Chapter introduction

The first task of the lead guitar expressive model concerns the automatic generation of the fingering (the definition of which string is used to produce each note, and in the case of a fretted string at which fret and with which finger of the fretting hand). This is essential, as on fretted string instruments, such as the guitar, in most cases notes can be played on at least two different strings and with different fret/finger combinations. Although the note's fundamental frequency remains the same, its execution on different strings and frets involves significant timbral variations and different use of the fretting hand, which could lead to relevant variations in performance. Both components have to be taken into account in the development of the expressive model.

3.1 The fingering problem

If in keyboard instruments, such as the piano, there is only one way to play each note (each pitch corresponds to one and only one key), the situation is decidedly more complicated in fretted string instruments. The notes included in the range of the instrument can be played in one or more ways, taking advantage of the available fret/string combinations and the open strings (when the string is made to resonate without being pressed against the keyboard) [27] ¹.

In the case of the guitar in standard tuning, there are six strings, tuned from the lowest (the sixth) to the highest (the first) as E2 (82 Hz) - A2 (110 Hz) - D3 (146 Hz) - G3 (196 Hz) - B3 (246 Hz) - E4 (329 Hz). Metallic frets are then arranged on the keyboard of the instrument, which allows the pitch of the notes produced on the string to be raised by pressing it slightly

¹For simplicity, notes generated through natural or artificial harmonics are not taken into consideration here, nor they were in any of the other scientific contributions considered.

in the space between two frets, shortening the portion of the string that vibrates. For each fret, moving in the direction of the body of the instrument, the pitch of the note generated rises by a semitone (in physical terms, a frequency equal to that at the previous fret, multiplied by $\sqrt[12]{2}$, is reached). Assuming a guitar equipped with 24 frets in standard tuning, we obtain the distribution of pitches observable in Figure 3.1.



Figure 3.1: The distribution of pitches on a 6-string guitar in standard E2-E4 tuning. Some pitches (such as the F2 highlighted in purple and the D6 in green) can be played only on a single open string or with a single string/fret combination, while others (such as the G4 highlighted in orange) can be executed in several different ways.

Moreover, the strings can be pressed against the fingerboard with different fingers of the fretting hand. The index, middle, ring, and little finger are typically used for this purpose. The same note, therefore, may not only be played on several string/fret combinations and on open strings but, in the case of fretted notes, also with different fingers of the fretting hand. Those are commonly numbered, index to pinkie, from 1 to 4, as shown in Figure 3.2

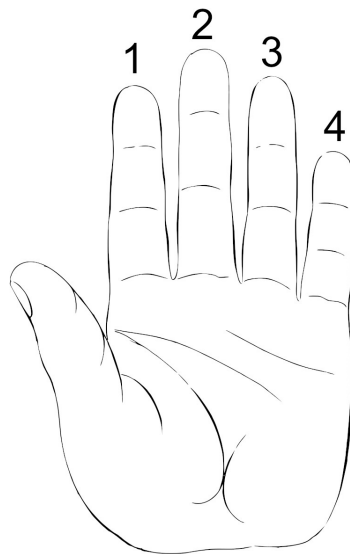


Figure 3.2: Numbering of the fingers of the fretting hand.

The choice of the open string or string/fret/finger combination to be used for each note is taken by the musician - rationalizing it or not - on the basis of biomechanical and expressive evaluations.

Biomechanical constraints and preferences may refer to:

- fretting hand displacement along (parallel to the long side) the keyboard, should this movement be required (change of position ² between two notes);
- fretting hand displacement across (parallel to the short side) the keyboard, should this movement be required (change of string between two notes);
- the spread between each pair of fingers of the hand (finger spread), or between each finger and the index finger of the hand (hand spread);
- the area of the fingerboard considered the most comfortable (the middle of the neck, typically between the fifth and twelfth positions, seems to be generally preferred, as noted in J. Cournut et al. (2021) [115], and in H. Heijink and R. G. J. Meulenbroek (2002) [116] in relation to classical guitar).

Expressiveness-related considerations may include:

- the use of open strings;
- the preference for using certain strings instead of others (for example, playing the same note on a thicker, lower string will produce a different timbre than playing it on a thinner, higher string. Additionally, the position on the fretboard can affect the decay and harmonic content of the note [117]).

I consider automatic fingering generation to be a necessary component of any expressive model dedicated to the guitar, obviously only if the fingering is not given as input (as in the case of the model presented here). This is because a significant part of the expressiveness of the instrument is linked to the choice between open or fretted strings and to that of the strings to be used. It must also be taken into account that, if the goal is to achieve a realistic virtual performance, the sequence of open strings and/or string/fret/finger combinations must follow a realistic path from the point of view of the human performer.

²The position of the fretting hand is obtained from that of the index finger. If this is placed at the fifth fret, the hand is in fifth position, etc.

3.2 Review of the scientific literature on automatic fingering generation

The topic of automatic generation of fingerings for fretted string instruments, specifically guitar, is not new in the literature but is not particularly well-trodden either.

One of the earliest contributions, also particularly relevant in relation to subsequent studies, is that of S. I. Sayegh and M. F. Tenorio [117], [118]. In their approach, the automatic fingering generation problem is solved by addressing it as a Shortest Path Problem (SPP) within a weighted, layered directed graph (referred to as a *Viterbi network*). Each layer represents a note in the sequence, and the nodes within each layer correspond to the possible fingerings for that note. The transitions between nodes are weighted according to biomechanical difficulty and musical consistency. The basic idea is to penalize transitions that require more effort (movements of the fretting hand along or across the fretboard), or that may result in a less uniform sound (the system tends to favor the use of the same string used for the previous note).

The influence of Sayegh and Tenorio's approach is evident in the subsequent work of D. Radicioni et al. [119]. There, a directed acyclic graph (DAG) is generated from the musical score, with each vertex representing a possible fingering for a note, and edges between vertices representing transitions between fingerings. Weights are assigned to the edges of the graph based on the estimated difficulty of transitions between fingerings. This accounts for horizontal movements along the fretboard, vertical shifts across the strings, and the spread between fingers. The authors grounded their estimation of transitions complexity on H. Heijink and R. G. J Meulenbroek (2002) [116]. A characteristic trait of this work is that the musical piece is manually segmented into phrases in the pre-processing stage. Individual graphs are then constructed separately for each phrase, and the optimal path is computed within each segment. After that, the system connects these segments to ensure smooth transitions between them.

Also organized around the division of the melody into phrases, but in this case based on timing, with rests used as boundaries, is the contribution by M. Miura et al. [120]. Their system focuses on generating guitar tablatures optimized for beginner players. The system minimizes wrist movement along the guitar neck, aiming to keep the hand as stable as possible. Reducing lateral displacements of the fretting hand increases the likelihood (depending on the melody) of string jumps or more pronounced spreads between the fingers. The authors took the latter into account, but considered it less problematic for a beginner than fretting hand movements along the fretboard. Interestingly, using a rule-based system, the system also includes specific performance techniques (hammer-on, pull-off, chalking-up, half-chalking-up, slide) in the generated tablature. However, it is unclear whether the user's requests for such techniques are general (e.g., on their frequency in relation to the total number of notes) or whether they can be

declined note by note. In the case of two techniques (chalking-up, half chalking-up), it is not even clear what exactly the authors are referring to, although the descriptions of the associated rules suggest mordant variants.

D.R. Tuohy and W.D. Potter tackled the problem of automatic generation of guitar fingerings by relying on a Genetic Algorithm (GA) [121]. The proposed system is primarily intended to support the creation of valid fingerings for harmonic parts (chords sequences), but single-note passages also appear in the examples given. Their fitness function assesses the playability of the generated tablatures by analyzing hand movement (the total amount of lateral hand and finger movement along the fretboard) and hand manipulation (that estimates the difficulty of the fretting hand requested arrangement). Excessive hand movements or challenging positions (whose evaluation is based on heuristic methods) are penalized, while the use of open strings and reduced lateral hand movements are rewarded. The system does not include information about which finger should be used to play each note.

G. Hori et al. started from the consideration that if the left-hand form at any point depends solely on the previous form (which is their assumption), a guitar performance can be modeled using a hidden Markov model (HMM) [122]. In this model, the hidden states are the left-hand forms, and the output symbols correspond to the chords (the actual notes) played using those forms. Each hidden state (or left-hand form) corresponds to only one output symbol, but more than one hidden state can correspond to the same output symbol. The search for the most probable sequence of hidden states is described as a decoding problem, and it's solved through the Viterbi algorithm [123]. It is interesting to observe that the authors highlighted how crucial it is to take into account the time available between one fretting hand form and the next one, a theme that is taken up in a structured way in the model proposed in this thesis:

In reality, the choice of the left-hand form depends not only on the previous form but also on the time interval between the current chord and the previous chord.

When the time interval is very long, we can almost ignore the dependency on the previous form; that is, the time interval inhibits the dependency ([122], p. 265)

The effect of time intervals is considered by employing the “input-output HMM” introduced by Y. Bengio and P. Frasconi [124]. In summary, we then have, for each t -th chord, the observable variables x_t (the notes) and d_t (the time between t and $t-1$), and z_t (the hidden state, i.e., the fretting hand arrangement).

In a subsequent work by G. Hori and S. Sagayama, the authors focused on simplifying the most challenging movements required to play a part, rather than minimizing the overall difficulty of execution, considering this approach more functional for beginners [125]. The paper introduces a variant of the basic algorithm for multi-layered networks, called the *minimax Viterbi algorithm*, to address this problem. When applied to an HMM-based fingering decision

network, the algorithm seeks the sequence of hidden states that maximizes the minimum transition probability along the sequence (transition probabilities are high for easy moves and low for difficult ones).

T. H. Raboanary et al. proposed a system for automatically generating fingerings for 4-string bass guitar based on dynamic programming [126]. Estimating the difficulty of the transition from one note to the next is based on three parameters: the fingers used, the size and direction of the interval, and whether the movement requires a change in the fretting hand position. Basically, the system seeks to minimize hand movement and to favor combinations of fingers that are considered easier than others.

A. D’Hooge et al. presented a system somehow similar in goals to that described in G. Hori et al. (2013) [122]. The system employs a fully connected neural network to determine the most appropriate chord diagram using the input parameters, which include the current chord label and the preceding chord diagram. The model transforms chord labels and diagrams into vectors and is trained on chord pairs to enhance the probability of smooth transitions between chords [127].

3.3 A new, more comprehensive and flexible approach to automatic fingering generation

3.3.1 High-level description of the system

The development of an innovative approach to the automatic generation of guitar fingerings, in several respects more comprehensive and customizable than those previously found in the literature, was achieved through collaboration between the Centro di Sonologia Computazionale and the Universities of Brescia (Italy) and Lille (France). The system developed includes not only automatic fingering generation, but also automatic, statistically based, insertion of articulations and expressive techniques, a topic that will be covered in Chapter 4. By choice, we addressed specifically monophonic (one note at a time) lead guitar parts. This work was presented during the Sound and Music Computing Conference (SMC) 2024, which was held in Porto (Portugal) [27], and represents a relevant portion of what is described in this chapter³. The system described below refers to the complete, user-customizable version presented in Bontempi et al. (2024) [27]. In the end-to-end model implementation detailed in Chapter 6, the mathematical model was adopted without substantial changes; however, the option for user customization of parameters was removed to provide a more immediate, user-friendly software experience.

³The code for this chapter is available at https://github.com/pierluigi-bontempi/PhD_thesis/tree/main/Chapter_3_modeling_of_fingering_choices.

Before delving into the system from a technical and implementation perspective, it is considered useful to indicate how it proves to be more comprehensive and versatile than pre-existing solutions.

Fretting hand movements along and across the fretboard

As seen in Section 3.2, the minimization of the movements of the fretting hand along (position change, parameter *PC*) and across (string change, parameter *SC*) the fretboard has been one of the most sought after objectives in automatic fingering generation systems. This is also true in the solution presented here, but with two peculiarities:

- it is checked that the time available to perform the horizontal hand displacement (consisting of the IOI with respect to the previous note) is congruous with the magnitude of the displacement itself. To do this, user-customizable configuration data include the time in ms required to move the hand horizontally for each fret (for example, if a time of 10 ms is set for each fret covered by the movements of the fretting hand along the neck, to carry out a displacement of 5 frets it will be necessary to have an IOI greater than 50 ms);
- the user can freely assign different penalties (or weights) to the displacements of the fretting hand along and across the neck.

This has some important implications. First, fingerings that are in fact inexecutable, due to the lack of sufficient time to make the required fretting hand movements, are automatically discarded from possible solutions. Moreover, the required time associated with horizontal and vertical fretting hand movements can be customized to simulate musicians with different technical abilities. The possibility to customize penalty levels to be associated with string changes and movements along the keyboard, then, allows different fingerings to be generated depending on personal preferences. Finally, interest in segmenting the melody into phrases or subsections (as seen in some previous approaches) gets lost, as the system automatically searches for the optimal fingering, taking advantage of passages where there is more time to reposition the fretting hand, if beneficial.

Distance from the *handy position*

As noted above, there is a tendency among guitarists to prefer an intermediate fretboard area, roughly between the fifth and twelfth positions ([115], [116]), most likely for reasons of convenience. In fact, the frets are further apart in the lower positions, while in the higher ones they are closer together. Additionally, the guitar body may restrict easy access to the top of the fretboard. Moreover, the pressure required to produce clear, flawless notes varies depending

on the specific string and fret, often making the initial part of the fretboard more challenging to play:

In summary, the experiment shows that the force range required to produce clean notes on the particular guitar tested stayed between 0.204 and 0.897, with an average of 0.423 kgf; this was considerably higher than initial calculations based on string action, gauge and tension had suggested. The calculated average force required was around 0.223 kgf with [5th fret, 1st string] as the position that required the least amount of force (0.141 kgf) and [1st fret, 3rd string] as the position that required the greatest amount of force (0.438 kgf) ([8], p. 338).

In the system presented here it is possible to freely set the range of keys to be considered “comfortable”. For example, by setting this range between the fifth to the twelfth fret, fretting hand positions from the first to the fourth and above the twelfth are penalized, proportionally to the distance from the optimal area. The parameter “distance from the handy position” (DH) can be used to better approximate the fingering choices a human performer would make, depending on the instrument used and personal preference, but it can also be used, in special cases, to force the use of a specific keyboard area.

Use of open strings

The use of open strings can sometimes be advantageous in biomechanical terms, but it can also result in a timbral fracture compared to the use of fingered notes. Unlike the approach of D.R. Tuohy and W.D. Potter [121], we therefore did not think it correct, or at least complete and versatile, to always consider the use of open strings as a goal to be pursued (or, on the contrary, something to be avoided). At the same time, we think that the use of open strings, because of their possible timbral impact on performance, should not be ignored as a relevant fact in itself, as is the case in many of the previous scholarly contributions. We have therefore introduced into our system the possibility of penalizing the use of open strings, to an extent customizable by the user (parameter NO , from “Number of Open strings”).

Fretting hand position tracking and hand spread evaluation

For each fretted note y_i we have, in case it is fretted, associated string $s \in S$ with $S = \{1, 2, \dots, s_{max}\}$, fret $f \in F$ with $F = \{1, 2, \dots, f_{max}\}$, and finger $h \in H$ with $H = \{1, 2, 3, 4\}$. The fingering in this case can then be described as $y_i(s_i, f_i, h_i)$. In the case of open strings, we have likewise $s \in S$, but without fret and finger assignments. Formally, we can describe open strings as $o_i(s_i)$.

Automatic fingering generation systems in the literature are generally based on constructing valid concatenations between contiguous fingerings. Formally, $y_{i+1}(s_{i+1}, f_{i+1}, h_{i+1})$ is chosen

based on $y_i(s_i, f_i, h_i)$ (for simplicity's sake, all the possible combinations between open and fretted strings are not listed) ⁴.

Although this approach can undoubtedly work in substance, an important element is lost, namely, knowledge at each instant of the fretting hand position (i.e. index position) $p \in P$ with $P = \{1, 2, \dots, f_{max}\}$. Constant tracking of the hand position on the fingerboard is essential to correctly estimate movements along the neck (position change), and it is implemented in our system.

Fretting hand position is also used to evaluate the hand spread (parameter HS). Given index finger placement at an f fret, when the middle finger is used at the $f+1$ fret, ring finger at the $f+2$ fret, or little finger at the $f+3$ fret, there is a hand spread value of zero (all the fingers are in their basic position with respect to the index). When the middle, ring, or little finger is placed on frets farther apart than their basic position, the hand spread value corresponds to the number of extra frets. For example, given the index finger placed at the f fret, if the middle finger is used at the $f+2$ fret, the hand spread value is 1. Hand spread can be penalized to an extent settable by the user and in proportion to its magnitude. For biomechanical reasons, the system does not allow the middle and ring fingers to be used on consecutive notes, leaving empty keys between them:

$$y_i(s_i, f_i, h=2) \rightarrow \neg y_{i+1}(s_{i+1}, f \neq f+1, h=3)$$

The same melody played on the same strings, at the same frets, and with the same fingers can result in different hand use (with related different PC and HS), as exemplified in Figure 3.3.

The user can set the proportional relevance of the penalties related to position change, string change, distance from the handy position, use of open strings, and hand spread. Each of these parameters, if desired, can be set to zero so as not to penalize at all the specific associated element (position change, etc.).

3.3.2 Implementation

The optimization model

The fingering generation code was written in *Optimization Programming Language* (OPL) as a static and compact mathematical programming formulation. The model can be handled both within the *IBM ILOG CPLEX Optimization Studio* software, v20.1.0⁵ or - for example - directly

⁴Moreover, for practicality and consistency with the nature of the system described here, reference is made to a monophonic part.

⁵<https://www.ibm.com/products/ilog-cplex-optimization-studio>, last accessed: 2024/08/13.



HS = 1, V position

HS = 1, V position



HS = 0, VI position

HS = 0, VI position

Figure 3.3: In the tablature (the bottom staff), each line represents a guitar string from the sixth (thicker) at the bottom to the first (thinner) at the top. The numbers indicate at which fret the string should be fretted. In the first and second rows of images, the same notes are played with the same fingers (represented by the numbers next to the notes heads in the musical staff) and with the same string/fret combinations. What distinguishes the two solutions is the position change present in the second but not in the first fingering, leading to a hand spread equal to 1 in the first fingering and to 0 in the second.

in Python, using the IBM ILOG CPLEX Python API.

The required input data are:

- the number of the frets available on the guitar;
- the number of the strings available on the guitar;
- the range of the notes that can be played;
- the tuning of the guitar (or, in other words, the correspondences between open strings and notes);

- the note generated by each string/fret combination;
- the number of the notes of the melody;
- the sequence of the notes of the melody;
- the available time to play each note (the IOI between that note and the previous one, in milliseconds);
- the maximum spread of the fretting hand, with respect to each finger;
- the time needed to move the fretting hand along the neck;
- the range of frets of the handy interval;
- the penalties to be associated to PC, SC, HS, DH, NO.

As can be seen, part of the input data is related to the instrument configuration, part to the melody, and part to the user's configuration preferences.

The objective function is:

```

minimize
alpha_PC * d_PC * tot_PC
+ alpha_SC * d_SC * tot_SC
+ alpha_HS * d_HS * tot_HS
+ alpha_DH * d_DH * tot_DH
+ alpha_NO * d_DH * tot_NO;

```

Where the values of the:

- α_X parameters must be greater than 0 and sum to 1, and are used to define the relevance of each penalizing factor compared to the others;
- d_X parameters represent the level of penalty to be associated with each respective factor. No particular constraints are placed on the sum of the values, and they can be set to 0;
- tot_X parameters are correlated to the number and magnitude of occurrences of the penalizing factors within the generated fingering.

Both d_X and α_X parameters are user-settable. Although the same results could be achieved by using only one penalty weight for each factor (PC, SC, etc.), it was preferred to keep two separate ones for greater semantic clarity and system flexibility.

When the CPLEX software is used, given a .dat file containing all the input data needed, and the .mod file containing the mathematical model, it returns the string/fret/finger combination or the open string on which to play each note by identifying the optimal solution, which minimizes

the total penalization generated by position and string change, hand spread, distance from the handy position, and use of open strings. The same results can be obtained using the CPLEX Python API. An example of two different fingerings generated starting from the same melody, varying the weights to be associated with PC, SC and HS is presented in Figures 3.4 and 3.5.

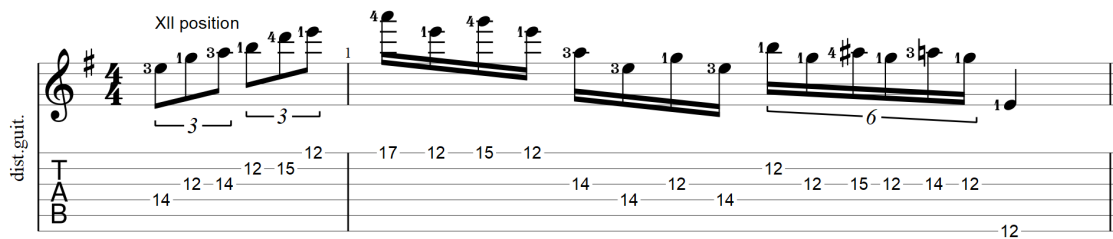


Figure 3.4: To obtain this fingering, the least penalized feature was the string change (SC). The hand spread (HS) was penalized twice as much as SC, while the position change (PC) was penalized twice as much as HS.

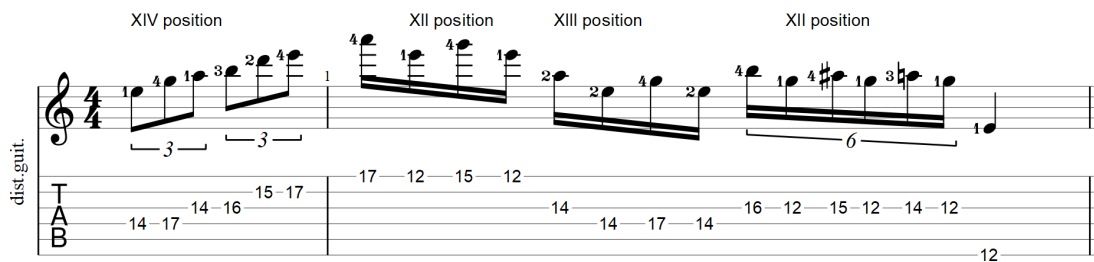


Figure 3.5: To obtain this fingering, the least penalized feature was HS. PC was penalized twice as much as HS. SC was penalized twice as much as PC.

Generation of the input data

Generation based on a Standard MidiFile

Input data can be manually encoded, automatically generated, or taken partially from a MIDI File. Regarding the last option, a Python script has been developed to parse MIDI files using the Mido library, extracting the necessary information and organizing it into well-structured data sets. The script is also capable of directly generating the .dat file for CPLEX, and it adds information about the instrument characteristics and optimization model configuration based on user preferences. This approach to obtaining input data is the reference one in relation to the final use of the expressive model, i.e. the situation in which, in an educational or musical production context, a virtual lead guitar performance must be obtained starting from a MIDI

melody, an almost universal format in the symbolic coding of music, and the reference one in music production digital environments.

Automatic generation of the melody

Also for the purpose of performing massive tests of the model described here and its evolution presented in Section 3.4, it was deemed useful to develop software capable of automatically generating plausible guitar melodic lines (along with instrument and optimization model configuration data), formatted appropriately for subsequent processing in CPLEX.

The field of computational creativity associated with automatic music composition is particularly vast, and even to recapitulate its major contributions would require space not commensurate with the goals of this thesis, which instead addresses computational expressive music performance. I therefore refer to the excellent review by F. Carnovalini and A. Rodà for a general presentation of the field [128]. Narrowing it to the solutions specifically aimed at lead guitar parts, the following should be mentioned. J. Biles utilized a genetic algorithm to generate realistic jazz solos [129]. The fitness function in his *GenJam* system assesses the quality of musical phrases by considering factors such as melodic shape, rhythmic variety, harmonic fit, and alignment with jazz improvisation norms. By incorporating both established musical rules and user feedback, the system guides the evolution of these phrases to produce jazz solos that are musically coherent and stylistically authentic. Over successive generations, this process refines the solos, leading to more sophisticated and pleasing musical results. McVicar et al. proposed *AutoLeadGuitar*, a probabilistic-based system designed to automatically generate guitar solos in tablature format, with input by the user of the key and the chord sequence. The model, trained on a dataset of over 11000 notes, integrates various guitar-specific techniques such as hammer-ons, pull-offs, and bends, and generates solos that maintain appropriate phrase boundaries, rhythms, and fretboard positions [130]. The system also highlights chord tones (notes used in the melodic line, that also belong to the underlying chord), and was the main inspiration for the original solution described below. N d. S. Cunha et al. used combinatorial optimization, specifically integer programming, to generate guitar solos [131]. The focus is on creating solos over a 12-bar blues chord progression by sequencing pre-defined licks in a way that optimizes musical coherence and, consequently, listener enjoyment. P. Sarmento et al. introduced *ShredGP*, a Transformer-based model designed to generate guitar tablatures in the style of specific iconic guitarists, such as David Gilmour, Jimi Hendrix, Steve Vai, and Yngwie Malmsteen [132]. By employing Deep Learning techniques, the system captures and reproduces the unique stylistic elements of these guitarists, including techniques such as hammer-on, pull-off, bending, or tapping.

In developing the original solution for the automatic creation of melodies (and all related

CPLEX data) based on which the guitar fingering are then generated, the following objectives were pursued:

- the melodic line and the rhythmic structure should be plausible, in musical terms, and reasonably similar to those commonly found in lead electric guitar parts;
- it should be possible to generate a large number of melodies that differ from each other, with a high variability of notes duration values in absolute terms (i.e., in ms, so that IOIs and thus the available time parameter used in the optimization model vary as well);
- there should be a random element that can generate, albeit occasionally, musical intervals that deviate significantly from those most commonly used;
- the reference instrument (in relation to the number of strings, number of frets, and tuning) should be randomly selectable from a shortlist of possibilities;
- for testing purposes, it should be possible to generate multiple complete instruction sets for CPLEX, keeping the melody and instrument configuration unchanged but varying user preferences.

First of all, the script selects, on a random basis, the reference instrument. At the moment, three options have been included, but more could easily be added if necessary:

- six-string electric guitar, standard tuning ($E2 > E4$);
- six-string electric guitar, drop-D tuning ($D2 > E4$);
- seven-string electric guitar, standard tuning with added low B ($B1 > E4$).

Also randomly selected from the available options are the tempo in beats per minute (bpm), the underlying chord (no harmonic progressions are provided, as they were not considered useful in this context), and the scale to be used, compatible with the chosen chord, as described in Table 3.1.

The sequence of duration values and the pitches of the notes that make up the melody are generated independently.

Regarding duration values, the system combines short predetermined modules, which represent common rhythmic sequences in MIDI ticks, based on a Markov chain. Each pre-built sequence corresponds to two bars in 4/4 and is associated with one of three levels of virtuousness. Each rhythm module also includes a connected Boolean array, which marks which notes in the series fall on the beat and which do not (whose usefulness will be clarified below). Once the first rhythmic module has been randomly selected, the Markov chain, based on a predefined

	Major scale (R-2-3-4-5-6-7)	Minor scale (R-2-b3-4-5-b6-b7)	Minor pentatonic scale (R-b3-4-5-b7)	Blues scale (R-b3-4-b5-5-b7)	Mixolydian scale (R-2-3-4-5-6-b7)
Major chord (R-3-5)	✓	✗	✗	✗	✓
Minor chord (R-b3-5)	✗	✓	✓	✓	✗
Dominant 7th chord (R-3-5-b7)	✓	✗	✗	✗	✓

Table 3.1: The chords and scales present in the automatic music generation script (easily expandable if necessary). For each chord or scale, the corresponding intervallic musical structure is indicated. After randomly selecting a root note (R) and a type of chord, the system randomly chooses one of the scales associated with the chord type (scale/chord associations are indicated with the symbol ✓ in the table), based on the same root.

transition matrix, concatenates a series of rhythmic modules, favoring (but not limited to) those belonging to the same category as the previous one. In this way, overall coherent but also varied rhythmic structures are obtained. The MIDI tick values are then converted to absolute values in ms based on the previously determined metronome tempo.

As for the choice of note pitches, that is also grounded on a Markov chain, but with some important adjustments. The basic transition matrix is based on the Platykurtic beta distribution⁶, centered on a pitch randomly chosen between the ones of the last note played, the previous or the following note in the selected scale. In this way, the pitches most likely to be selected are those close to the previous one, as commonly happens in guitar solos, but at the same time the risk of obtaining repetitions of the same note too often is mitigated. Also the normal and the double gamma⁷ distribution were tried, but with less convincing results. Before selecting the pitch of the following note, the transition matrix is altered if the note is on the downbeat, doubling the probabilities of chord tones compared to the other notes of the scale. In this way, the system tends to place notes on the beat that are fully consonant with the underlying chord, as an expert guitarist would do (the notes on the downbeat are the most exposed and significant). When the pitch theoretically chosen by the Markov chain is outside the range of the instrument, a random one is automatically selected, taking into account the time available to reach it (to avoid impracticable changes of position or string) and, also in this case, increasing the probability of obtaining a chord tone on the downbeat notes.

⁶https://docs.scipy.org/doc/scipy/tutorial/stats/continuous_beta.html#beta-distribution, last visited: 2024/08/15

⁷https://docs.scipy.org/doc/scipy/tutorial/stats/continuous_dgamma.html, last visited: 2024/08/15

In terms of user-settable optimization model parameters, the script generates consistent configurations and produces variants of each instruction set, modifying these preferences while keeping the melody and instrument fixed for testing purposes.

3.4 Ongoing development of the system

A second version of the system presented above is being developed. This is based on a multi-objective model, which also receives in input articulation and expressive techniques to apply to each note. A more detailed description of the project is available in the Section about publications under preparation, at page 150.

Chapter 4

Modeling of articulations and expressive techniques

Chapter introduction

In lead electric guitar, it is not only fingering that should be considered among the choices that directly impact the expressive output of the performance. Articulations (how a note is produced) and expressive techniques (intervening on the note, changing some of its parameters for *colorful* purposes) should also be considered. This chapter is dedicated to the expressive model module, which, given the melody and the basic fingering, automatically inserts articulations and expressive techniques (at a symbolic level) in order to obtain a more realistic, expressive, and engaging virtual performance.

4.1 Defining the distribution of articulations and expressive techniques: a statistical approach

In the sequence of steps to achieve the expressive computational performance, after the first module (described in Chapter 3), which is responsible for automatically generating the fingering of the melody, comes the one devoted to the automatic insertion of articulations and expressive techniques¹. These have been introduced in Section 1.3.2.

The first problem that the module in question must solve is to define the distribution of the techniques, that is, how to identify, for each note, whether or not to apply (where possible) each of the articulations and expressive techniques implemented. The issue is not one of immediate

¹As already reported, a video in which I present the techniques implemented in the model is available on the Sound and Music Computing (SMC) 2024 conference website: https://smcnetwork.org/smc2024/videos/SMC2024_video_id54.mp4, last accessed: 2024/11/22.

solution, nor is it unambiguous, as there is basically no compulsory correspondence between melodic line/fingering and performance techniques. Whether or not to use them, and if so in which contexts, depends primarily on the conventions of the musical genre and the preferences of the individual guitarist. The correlation between specific musicians and the frequency of use of each common technique was found in the aforementioned Sarmiento et al. (2023) [132]. Some techniques are also considered in Miura et al. (2004) [120], as described in Section 3.2, but, in general, the topic of lead electric guitar articulations and expressive techniques is well underrepresented in the literature, not only from the computational perspective, but also from the musicological and biomechanical ones (an interesting - non-computational - exception is D. R. Grimes (2014) [133], which explores the topic from a physical perspective). The problem has been addressed, in the case of the electric bass, in Goddard et al. (2017) [134], where the emphasis is placed on the creative dimension of the performative interpretation of a musical part. There, the transition from the musical symbolic input to its *enriched* output version (intended then to be played through a physical model) is grounded on case-based reasoning.

In our case, it was decided to tackle the problem, first of all, on a statistical basis, leaving the user with ample possibilities for customizing the module’s behavior. The statistical data on the distribution of articulations and expressive techniques on each string were obtained from the proprietary *MySongBook* dataset. This consists of over 2000 music transcriptions in standard notation and guitar tablature realized by professional musicians, encompassing various genres². The scores/tabs in the corpus are the work of expert transcribers, and thus - although fewer in number - less prone to inaccuracies and approximations than the transcriptions commonly found online and made by general users (a relevant dataset in this regard is DadaGP [135]). These transcriptions are created using *Guitar Pro* software, developed by Arobas Music³, primarily targeting guitarists. However, they often include also vocal parts or other instruments such as bass guitar or drums. Among the guitar parts, the lead sections were identified using the procedure described in Régnier et al. (2021) [136].

The occurrences of bending, hammer-on, pull-off, and vibrato techniques detected on each string were divided by the total number of notes per string, yielding the distinct frequency of each technique’s use per string, as represented in Figure 4.1.

This statistical distribution was used as a baseline for automatic input by the expressive model of articulations and expressive techniques. In any case, as noted earlier, given that the frequency of use of different techniques is also strongly related to musical genre and personal preferences, it was deemed appropriate to allow the user, when setting the script preferences, to alter the percentages measured over the entire *MySongBook* corpus as deemed appropriate, as described below. Likewise in relation to automatic fingering generation, what follows refers

²<https://www.guitar-pro.com/c/18-music-scores-tabs>, last accessed: 2024/08/18.

³<https://www.guitar-pro.com/>, last accessed: 2024/08/18.

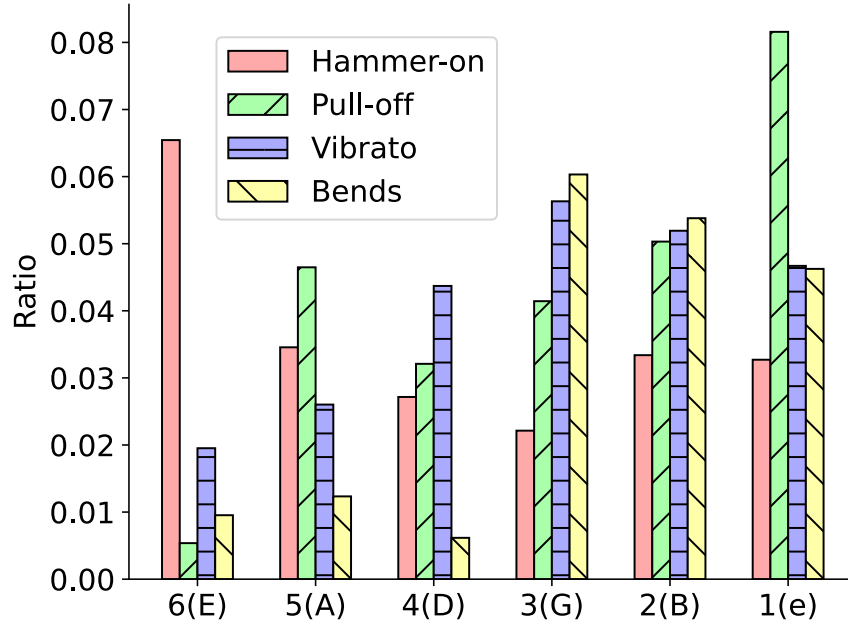


Figure 4.1: Ratio of the usage of the analyzed techniques to the total number of notes for each string. Data were collected for 6-string guitar transcriptions, the vast majority of the total. Strings are labeled according to standard guitar tuning, with the 6th string being the low “E” and the 1st string being the high “e” (image taken from Bontempi et al. (2024) [27]).

to the complete, user-customizable solution presented in Bontempi et al. (2024) [27]⁴. In the end-to-end model implementation detailed in Chapter 6, the same techniques insertion logic was adopted without substantial changes; however, the option for user customization of parameters was removed to provide a more immediate, user-friendly software experience.

4.2 The rule-based system for automatic techniques insertion

The performance techniques considered are bending, hammer-on, pull-off, legato slide, and vibrato. Standard picking is implicitly considered present when the note is not articulated through a hammer-on or pull-off and cannot be generated through the release of the bending present on the immediately preceding note. The inclusion of each technique in the melody follows the order in which those are presented here, also based on verifying compatibility with those previously entered. Every technique is then subject to a specific set of rules and logic, presented below.

⁴The code for this chapter is available at https://github.com/pierluigi-bontempi/PhD_thesis/tree/main/Chapter_4_modeling_of_articulations_and_expressive_techniques.

4.2.1 Bending

Upward bending is an expressive technique in which the guitarist pushes or pulls a string across the fretboard to change the pitch of a note (see Figure 4.2). The pitch is gradually raised by bending the string, creating a smooth and expressive transition to the target note.



Figure 4.2: A bend on the third string (colored in red for clarity).

Reverse bends are performed by starting with the bent string and then releasing it, returning to its resting position. This type of bending, less frequent than the upward one, was not implemented (see Figure 4.3).

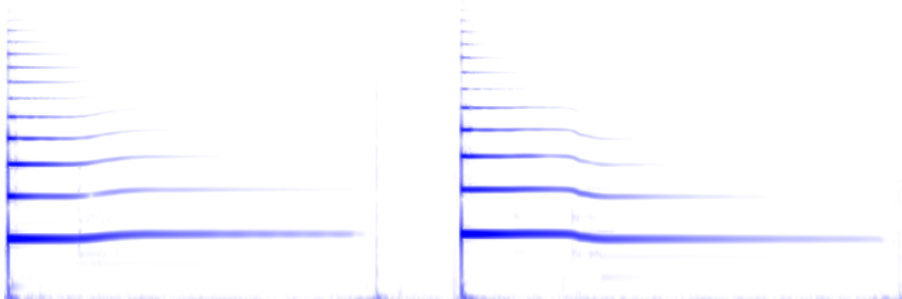


Figure 4.3: The spectrogram of an upward bend, followed by a downward one. Time is represented on the abscissa axis; frequencies are represented on the ordinate axis.

In guitar tablatures, bends are typically shown with an upward arrow next to the fret number, indicating bending amplitude in semitones or textually (*full* stands for two semitones, *half* for one semitone). In standard notation, the way of indicating bending varies with scoring software but often involves entering the starting note and the one reached by bending, distinguished by some graphical solution, as shown in Figure 4.4.

For the automatic application of bending, the algorithm goes in search of melodic configurations in which bending is most commonly used, identified through an in-depth analysis of the *MySongBook* corpus and reference lead guitar manuals (such as [137]–[139]).

Specifically, the system evaluates the insertion of bends in the presence of:

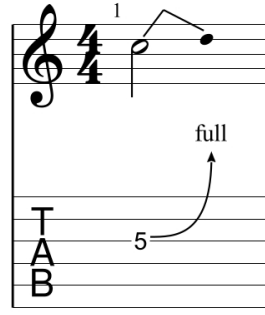


Figure 4.4: A D4 obtained through a whole-step (two semitones) bend on the fifth fret of the third string (without the bend, a C4 would instead be generated).

- two or more consecutive notes in unison;
- groups of three consecutive notes with first and third notes equal to each other and the second presenting higher or lower pitch (e.g., G4-A4-G4, where A4 is performed through a bend, keeping the string/fret/finger combination unchanged);
- two contiguous notes that can be played at the same fret of the same string, using a bend on one of them.

The input of the bends is also subject to a maximum value in semitones for each finger of the fretting hand, which the user can set. Biomechanical feasibility, based on the previously generated fingering, is also evaluated. Following the insertion of the bends, the fingering is changed, sometimes even changing the string on which the note is played, but taking care to always generate biomechanically viable solutions.

In the basic statistics, the presence of bending was found in 4,6% of the notes played on the first string, 5,4% of the notes on the second, 6% of the notes on the third, 0,6% of the notes on the fourth, 1,2% of the notes on the fifth, and 1% of the notes on the sixth. The user can increase or decrease these rates as desired. The basic or the modified statistics are used as targets by the technique insertion algorithm, presented as pseudo-code in the Appendix.

4.2.2 Hammer-on

Hammer-on is an articulation in which the note is obtained by quickly and firmly pressing down a finger of the fretting hand on a string, without picking the same. In most cases (those modeled here), the note articulated through the hammer-on follows another - lower - note played on the same string. This technique is commonly used in solos and riffs to create fast, fluidly connected note sequences.

In the model presented here, a note articulated using a hammer-on can follow a fretted note, as well as a note produced by picking an open string. Formally, given a fretted note $y_i(s_i, f_i, h_i)$,

a hammered note $y_{i+1}(s_i, f_{i+1} > f_i, h_{i+1} > h_i)$ can follow, as long as the maximum hand spread is respected. Alternatively, given a note played on an open string $o_i(s_i)$ an hammered one $y_{i+1}(s_i, f_{i+1}, h_{i+1})$ can follow.

In tablature and standard notation, the hammer-on is usually marked with an "H" and/or a curved line between the notes, as shown in Figure 4.5.

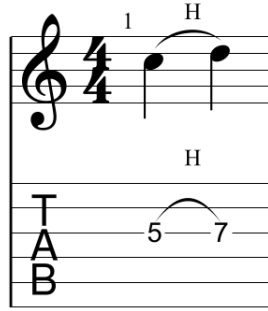


Figure 4.5: A D4 played on the seventh fret of the third string using a hammer-on.

In the basic statistics, the presence of hammer-on was found in 3,3% of the notes played on the first string, 3,3% of the notes on the second, 2,2% of the notes on the third, 2,7% of the notes on the fourth, 3,5% of the notes on the fifth, and 6,5% of the notes on the sixth. The user can increase or decrease these rates as desired. The basic or the modified statistics are used as targets by the technique insertion algorithm, presented as pseudo-code in the Appendix.

4.2.3 Pull-off

Pull-off is an articulation in which a note is produced without picking the string, similar to what happens with the hammer-on. The similarities end there, however, in that in pull-off, after a fretted note, instead of simply lifting the fretting finger off the string, the same is pulled slightly downward or upward. Through this movement, the string is essentially plucked, similar to what one might do with a finger of the other hand, and a note lower than the previous one is reached. This technique is commonly used in solos and riffs to create fast, fluidly connected note sequences.

In the model presented here, a note articulated using a pull-off can be both fretted or on an open string and can only follow a fretted note. Formally, given a fretted note $y_i(s_i, f_i, h_i)$, a fretted note $y_{i+1}(s_i, f_{i+1} < f_i, h_{i+1} < h_i)$ or a note on an open string $o_i(s_i)$ can follow when articulated through a pull-off (as long as the maximum hand spread is respected in the case of pull-off between fretted notes).

In standard notation and tablature, pull-offs are typically marked with a "P" between the notes and/or a curved line connecting them, as shown in Figure 4.6.

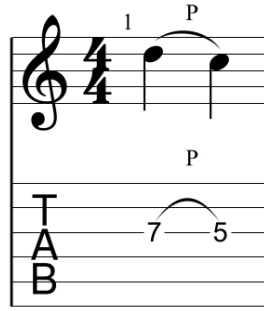


Figure 4.6: A C4 played on the fifth fret of the third string using a pull-off.

In the basic statistics, the presence of pull-off was found in 8,2% of the notes played on the first string, 5% of the notes on the second, 4,1% of the notes on the third, 3,2% of the notes on the fourth, 4,6% of the notes on the fifth, and 0,5% of the notes on the sixth. The user can increase or decrease these rates as desired. The basic or the modified statistics are used as targets by the technique insertion algorithm, presented as pseudo-code in the Appendix.

4.2.4 Legato slide

Legato slide is a technique that creates a smooth, connected transition between two notes. It involves playing a fretted note and then, without re-picking the string, sliding the same finger up or down the neck to another fret. The second note is sounded purely by the sliding motion, resulting in a seamless, flowing sound (see Figure 4.7). Formally, legato slide can be used only between two fretted notes $y_i(s_i, f_i, h_i)$ and $y_{i+1}(s_i, f_{i+1}, h_i)$.

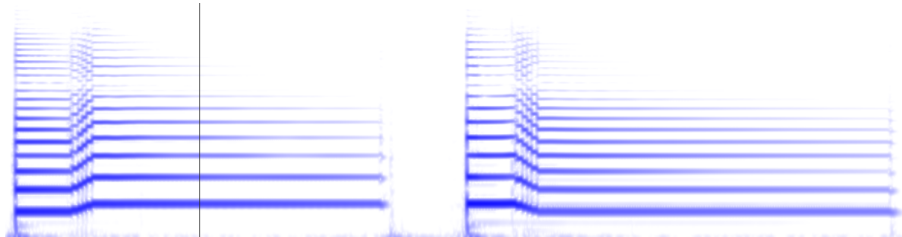


Figure 4.7: The spectrogram of an ascending legato slide, followed by a descending one. The presence of ‘steps’ is due to the frets, which impose a movement by semitone jumps.

Given that the techniques in the system presented here are applied based on the automatic fingering generated earlier and that a legato slide requires two consecutive notes played on the same string and with the same finger (a rather rare occurrence), it was deemed appropriate to simply ask the user if s/he would like to always insert a legato slide where possible without relying on statistical references as with the other techniques.

In standard notation and tablatures, a legato slide is typically represented by a diagonal line connecting the two notes involved, combined with a curved line above them and sometimes the

textual indication *sl.*, as shown in Figure 4.8.

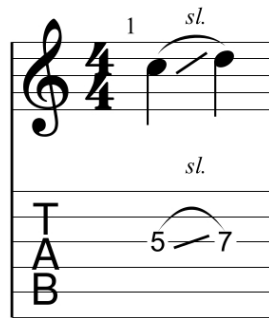


Figure 4.8: A legato slide between a C4 and a D4, played on the fifth and seventh fret of the third string.

4.2.5 Vibrato

Vibrato is an expressive technique commonly used in stringed instruments to rhythmically vary the pitch by swinging it. In electric guitar lead playing, it is usually performed by rapidly bending and releasing the string using one or more fingers of the fretting hand (the case used here as reference), but it can also be done with the use of the floating bridge, if present (see Figure 4.9).

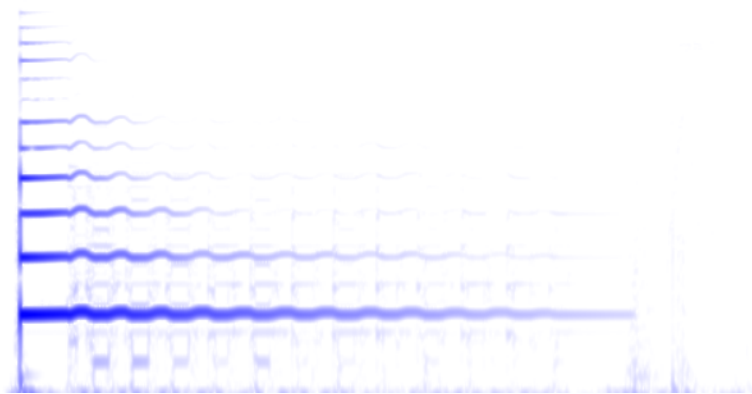


Figure 4.9: The spectrogram of a vibrato.

Vibrato is less common on the low strings, as observable in Figure 4.1, but its use is also related to note duration: it is easy to perform it on longer notes, to which it can lend more interest and movement, too, making them less static.

Given two note durations in ms t_1 and t_2 (customizable by the user), with t_1 corresponding to the minimum duration value for applying the vibrato and $t_2 > t_1$ used to identify the notes for which the application of vibrato is highly probable, the system applies the expressive technique firstly to the fretted notes whose duration exceeds t_2 , and subsequently, if needed, to those that

exceed t_1 but not t_2 , based on the statistical data retrieved from the *MySongBook* corpus and/or user preferences.

Vibrato is typically represented in standard notation and tablature by a wavy line, or sometimes a series of short, wavy lines (~), placed above or next to the note that should present the vibrato effect, as shown in Figure 4.10.

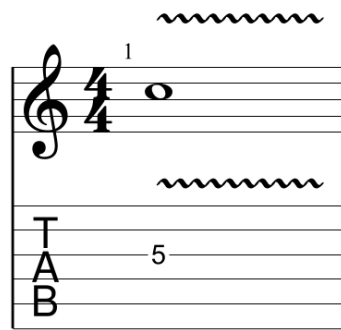


Figure 4.10: A vibrato applied to a C4, played on the fifth fret of the third string.

In the basic statistics, the presence of vibrato was found in 4,7% of the notes played on the first string, 5,2% of the notes on the second, 5,6% of the notes on the third, 4,4% of the notes on the fourth, 2,6% of the notes on the fifth, and 2% of the notes on the sixth. The user can increase or decrease these rates as desired and ask the system to always apply vibrato to each fretted note with a duration greater than t_1 , regardless of the statistical data. Except for the last case scenario, the basic or the modified statistics are used as targets by the technique insertion algorithm, presented as pseudo-code in the Appendix.

Chapter 5

Unintentional expressiveness due to the biomechanical dimension: a Machine Learning-based modeling approach

Chapter introduction

The previous chapters described how, starting from a simple sequence of notes, a plausible fingering from the point of view of the human guitarist is generated and then enriched with the most common articulations and expressive techniques. In other words, the system autonomously makes the choices a guitarist usually takes when s/he has to perform a lead part. The next step is to model what happens during the actual performance. This chapter addresses unintentional expressiveness, i.e., the deviations in terms of timing and loudness that are introduced unconsciously by the musician and which are linked to the biomechanical (which movements are performed) and psychological components, as well as to peculiar (and unaware) performative traits of the performer. To do this, a dataset of *inexpressive* guitar performances was created, complete with fingerings and techniques used. The same was then used to train predictive modules based on Machine or Deep Learning.

5.1 Modeling unintentional expressiveness

The approach to musical performative expressiveness followed in this thesis is based on the distinction between two different dimensions: on the one hand, non-intentional expressiveness, on the other, the intentional one (this topic was introduced in Section 1.4.5). Both act on the same parameters (primarily timing and loudness), but they do so from different perspectives. Considering that the main application area imagined for the expressive model is that of music

production, even on a practical level it seems focused and justified to separate the unintentional manipulation of the expressive parameters (managed autonomously by the model) from the intentional one (based on the instructions of the user, similarly to what happens in a recording studio between a musician and an artistic producer).

As observed in L.L. Costalonga et al. (2019) [8], there are basically two primary approaches to modeling expressive performance: one that focuses on the outcome of the performance (*Psycho-Acoustics-Based modeling*) and another that delves into the internal processes of the performer (*Performer-Based modeling*). The first is undoubtedly the one widely predominant in the literature. The model presented in this thesis aims to consider both dimensions while keeping them separate. A similar concept is also expressed in Canazza et al. (1997) [140], where performances defined as ‘normal’ (and thus not intentionally expressive) are contrasted with performances of the same musical extract characterized by different expressive intentions (hard, soft, heavy, light, bright, dark).

As noted in Section 1.5.2, data-driven approaches seem to be particularly suitable for modeling local expressivity (i.e., relative to small groups of contiguous notes). Also in the solution presented here, approaches of this type are used to predict timing and loudness deviations. Specifically, the predictive capabilities of Decision Trees, Random Forests, Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) solutions were tried. All the regressors were fed with data on current note properties, fingering, techniques used, and timing and loudness deviations of the previous note (Decision Tree, Random Forest, one time step RNN and LSTM) or notes (four time steps RNN and LSTM). The different algorithms were tested to verify and compare their performance and predictive capacity. In particular, I wanted to verify whether a fully transparent algorithm such as the Random Forest one, with input data limited to the current and main features of the previous note, could obtain results comparable to the RNN and LSTM-based solutions¹. The results are presented in Section 5.4.

5.2 The dataset

Several guitar performance datasets are available and can be used for scientific purposes.

In Su et al. (2014) [141] a dataset containing 6,580 audio clips and 11,928 individual notes dedicated to electric guitar lead techniques recognition is described. This covers seven distinct playing techniques: normal, muting, vibrato, pull-off, hammer-on, sliding, and bending. The recordings were made by a professional guitarist using high-quality equipment. To reflect a variety of musical genres — such as rock, metal, funk, and country — the dataset includes

¹The code and data related to this chapter are available at https://github.com/pierluigi-bontempi/PhD_thesis/tree/main/Chapter_5_Unintentional_expressiveness_due_to_the_biomechanical_dimension.

multiple guitar sounds with varying levels of distortion, reverb, delay, and chorus. The recordings are designed to highlight the subtle sound characteristics of each playing technique in the context of single notes, not melodies or polyphonic parts.

Also the dataset described in Stein et al. (2010) [142] is based on single notes (or chords) with varying audio effects, and it is used to train a system for the automatic detection of audio effects.

In Xi et al. (2018) [143] GuitarSet is presented. That comprehensive dataset features high-quality polyphonic guitar recordings paired with detailed annotations and metadata. The dataset includes separate recordings of each string, thanks to the use of a hexaphonic pickup. The collection comprises various musical excerpts played on an acoustic guitar, complete with time-aligned annotations detailing string and fret positions, beat/downbeat tracking, chord estimation, and playing style.

In Chen et al. (2022) [144] a new dataset named EGDB is introduced. This includes transcriptions of the electric guitar performances of 240 tablatures, rendered with different tones. The dataset is designed to benchmark the performance of automatic guitar transcription systems.

GAPS, described in Riley et al. (2024) [145], is a large dataset of real guitar audio recordings. It comprises 14 hours of audio, associated aligned scores, aligned MidiFiles, and performance videos.

In Comunità et al. (2021) [146], a dataset derived from that presented in Stein et al. (2010) applying to the unprocessed recordings 13 overdrive, distortion, and fuzz plug-ins software emulations is presented. The dataset aims to support the development of an automatic guitar effect recognition system based on Convolutional Neural Networks (CNNs). It contains about 250 hours of processed electric guitar samples, divided into four sub-datasets with both monophonic and polyphonic samples.

The multimodal dataset discussed in Mitsou et al. (2024) [147] includes 549 samples of audio and video, capturing nine distinct playing techniques such as alternate picking, vibrato, hammer-on, pull-off, etc. The recordings were made in a realistic home studio setup with various guitars and amplifier simulations, enhancing the diversity of the data. The collection also includes the corresponding transcriptions.

Unfortunately, none of the above-mentioned datasets possesses all the necessary characteristics for implementing the unintentional expressivity module introduced in Section 5.1. These are datasets created to train automatic transcription systems or retrieve useful information in that context. Moreover, the materials are often polyphonic or based on single isolated notes. The reference dataset needed here to train Machine or Deep Learning solutions, on the other hand, had to be:

- related to lead electric guitar parts (not single notes), monophonic and with annotation not

only of the musical features and fingering but also of the articulations used (specifically, hammer-on, pull-off, and bending);

- calibrated so as to represent the variables seen in Chapters 3 and 4 adequately, i.e., position change, string change, hand spread, IOI;
- associated with intentionally inexpressive performances, i.e., in which the musician is asked to play as neutrally as possible, without adducing any intentional expressive traits, in order to be able to separate the contribution given by the unintentional component from the one due to the intentional one, in the deviations of the expressive parameters. Using datasets of real performances, in which the musician is not placed under any constraints but is instead left free to express her/himself, separating later the unintentional from the intentional component (which includes, for example, the gestures described by Susan Rogers in Section 1.4.2) would become extremely complicated, if not impossible.

I therefore decided to build an *ad hoc* dataset. This consists of 10 exercises (reported in standard notation and tablature in the Appendix), each of which includes a minimum of 57 and a maximum of 320 notes, each played at four different metronomic tempos (60/80/100/120 bpm), for a total of over 7000 notes. The exercises, even the longest, are mostly based on repeated patterns applied to the chromatic, the minor pentatonic, or the natural minor scale, so as to be easily memorized by the performer. This is a valuable element, particularly should one wish to expand the dataset (as in this case, see the Section about scientific papers under preparation, at page 150) to include recordings of the same materials made by other guitarists. I made the recordings myself, using a high-end, hand-made six-strings/24 frets electric guitar in standard tuning connected to a Focusrite Clarett+ 8Pre sound card, with direct hardware monitoring active, which allows the input signals to be sent to the speakers with negligible latency. The Digital Audio Workstation (DAW) used was Steinberg Cubase Pro 13. The signal was recorded at a sampling rate of 48 kHz and 24-bit resolution in lossless WAV format. A headroom of approximately 12 dBFS was maintained during the recordings, and a mild limiting was applied to uniform the same.

The extraction of features from digital audio was carried out using the Celemony Melodyne Assistant plugin, version 5.4.1², see Figure 5.1), which identifies onset and offset, loudness and pitch of each note of monophonic musical parts, and allows the subsequent generation of MIDI parts based on that data. Although there is no shortage in the literature of examples of automatic transcription systems for guitar parts (see, among others, [148]–[151]), it was preferred to proceed in this way due to the versatility and particularly clear visual feedback of Melodyne and Cubase, which allows intervening easily if corrections are necessary. In particular, it was found that while Melodyne is extremely accurate in detecting note onsets, it tends to postpone note

²<https://www.celemony.com/en/melodyne/what-is-melodyne>, last accessed: 2024/11/19

closure beyond their actual end. To correct this error, the offsets were automatically calculated on the basis of the signal level dropping below -24 dBFS for a time of at least 30ms.

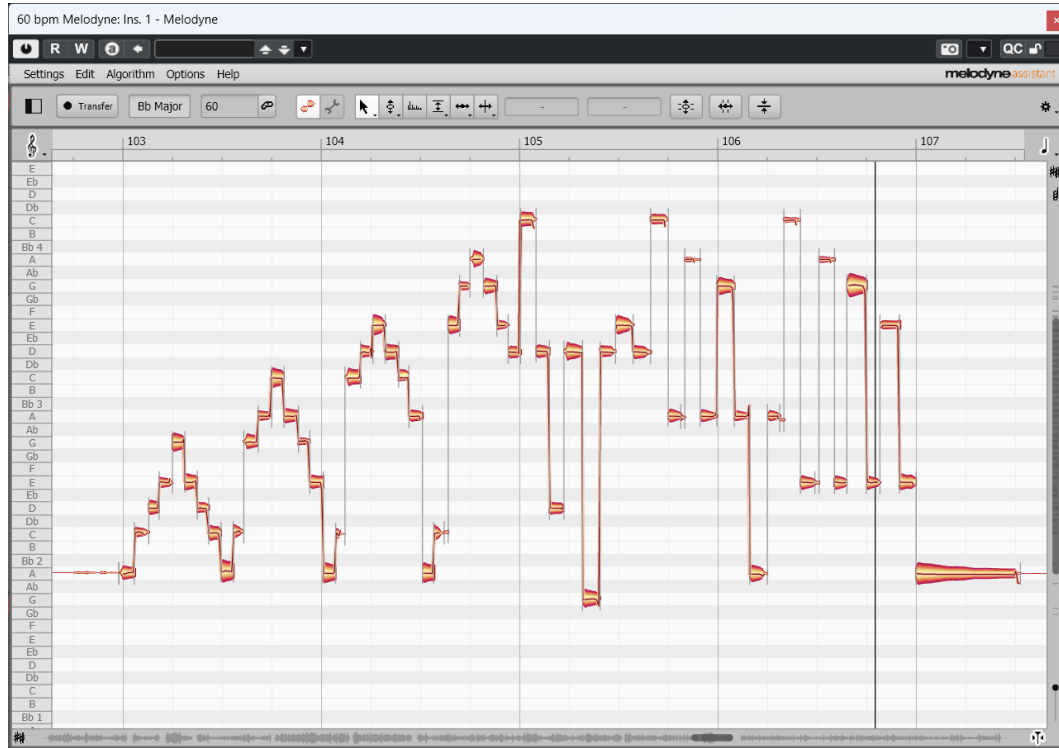


Figure 5.1: Celemony Melodyne Assistant. The automatically identified notes, represented as colored bubbles on a piano roll, are visible.

The generated MIDI parts were then exported as Standard MIDI Files. For export, a PPQN value of 960 was set, which leads to a temporal resolution of approximately 1 ms for the parts recorded at 60 bpm (60 seconds / 60 beats per minute / 960) and half ms for those recorded at 120 bpm. The dataset is, therefore, composed of the audio recordings, their conversion into MIDI Files, and the data extracted from the MIDI Files in CSV format.

In addition to the data on the temporal positioning of onset and offset of the notes with respect to the start of the part in MIDI ticks, and MIDI velocity (obtained through the procedure described above, `t_dt_note_on`, `t_dt_note_off`, and `velocity`, respectively), the dataset, when completed, included the following features:

- **bpm**: the metronomic tempo of execution, in beats per minute, of the exercise of which the note is part (60, 80, 100, or 120);
- **exercise**: the exercise of which the note is part (from 1 to 10);
- **ex_note_num**: the progressive number associated with each note of each exercise (each exercise starts from 1);

- **start-stop**: label indicating whether a note is the first (start), last (stop), or is an intermediate note (continue) within each exercise;
- **note_num**: MIDI note number;
- **note_name**: note name (for example C4);
- **onbeat**: label indicating whether the note falls on the beat (1) or not (0);
- **t_dt_note_on_ref**: total delta time in MIDI ticks of the onset of the note, from the beginning of the MIDI part, with reference to the ideal position, as indicated in the score;
- **t_dt_note_off_ref**: total delta time in MIDI ticks of the offset of the note, from the beginning of the MIDI part, with reference to the ideal position, as derived from the score;
- **note_lng_ref**: length of the note, in MIDI ticks, with reference to the ideal length, as derived from the score;
- **IOI_ref**: Inter Onset Interval (distance from the previous note onset), in MIDI ticks, with reference to the ideal value, as derived from the score;
- **prec_rest_ref**: length of the rest preceding the note, in MIDI ticks (≥ 0), as derived from the score;
- **string**: number of the string used to play the note;
- **fret**: number of the fret used to play the note (0 in case of open string);
- **finger**: number of the string used to play the note (from 1 - index - to 4 - pinkie);
- **position**: position of the fretting hand (≥ 1);
- **PC**: position change made to reach the note (frets covered by the position change). It can take on positive integer values (displacement in the direction of the guitar body) or negative (displacement in the direction of the headstock);
- **SC**: string change made to reach the note (strings covered by the string change). It can take on positive integer values (displacement in the direction of the thicker strings) or negative (displacement in the direction of thinner strings);
- **HS**: hand spread;
- **hammer-on**: boolean variable (possible values 0 or 1) indicating whether a hammer-on is applied to the current note;
- **pull-off**: boolean variable (possible values 0 or 1) indicating whether a pull-off is applied to the current note
- **bending**: indicates the presence (value > 0) and the amplitude in semitones of the bending applied to the current note;

- **dt_note_on**: delta time (temporal distance) in MIDI ticks of the onset of the note from the last MIDI event, with reference to the actual performance;
- **t_dt_note_on**: total delta time in MIDI ticks of the onset of the note, from the beginning of the MIDI part, with reference to the actual performance;
- **dt_note_off**: delta time in MIDI ticks of the offset of the note from the last MIDI event, with reference to the actual performance;
- **t_dt_note_off**: total delta time (temporal distance) in MIDI ticks of the offset of the note, from the beginning of the MIDI part, with reference to the actual performance;
- **note_lng**: length of the note, in MIDI ticks, with reference to the actual performance;
- **IOI**: Inter Onset Interval (distance from the previous note onset), in MIDI ticks, with reference to the actual performance;
- **prec_rest**: length of the rest preceding the note, in MIDI ticks (≥ 0), with reference to the actual performance;
- **velocity**: MIDI velocity of the current note, with reference to the actual performance.

The following features were then calculated directly in Python, based on the data available in the dataset, and were added to the related Pandas dataframe:

- **t_dt_note_on_ms**: same as **t_dt_note_on**, but converted in milliseconds;
- **t_dt_note_off_ms**: same as **t_dt_note_off**, but converted in milliseconds;
- **IOI_ms**: same as **IOI**, but converted in milliseconds;
- **prec_rest_ms**: same as **prec_rest**, but converted in milliseconds;
- **delta_onset_ms**: the difference between the actual and ideal total delta time of the onset of the current note, in milliseconds;
- **delta_onset_ms-1**: same as **delta_onset_ms**, but related to the previous note, in milliseconds;
- **delta_onset**: the difference between the actual and ideal total delta time of the onset of the current note, in MIDI ticks;
- **delta_onset-1**: same as **delta_onset_ms**, but related to the previous note, in MIDI ticks;
- **delta_offset_ms**: the difference between the actual and ideal total delta time of the offset of the current note;
- **velocity-1**: same as **velocity**, but related to the previous note;
- **PC-abs**: same as **PC**, but in absolute value;

- **SC-abs:** same as SC, but in absolute value;

A small portion of the features were not actually used for the purposes of descriptive statistics and the creation of predictive models described below. Still, it was nevertheless considered appropriate to include them in the dataset in view of possible future developments.

5.3 Descriptive statistics

5.3.1 Onsets' time deviation

The temporal position of the onsets of the notes can undoubtedly be considered among the main parameters relating to a musical performance. In the Pandas dataframe, the corresponding features are `delta_onset_ms` and `delta_onset`. The first measures the time deviation of the onsets with respect to the *ideal, perfect* performance in milliseconds, while the second does the same in MIDI ticks. We, therefore, have a measurement of time deviations in absolute terms (ms) and one in relative terms (MIDI ticks).

In figure 5.2, it can be seen how, even when varying the metronomic time, the distribution of time deviations remains relatively constant in absolute terms. Specifically, the main parameters are as follows:

- **60 bpm**
Mean: -2 ms; Median: -2 ms; Standard Deviation: 23 ms
- **80 bpm**
Mean: 7 ms; Median: 7 ms; Standard Deviation: 20 ms
- **100 bpm**
Mean: 0 ms; Median: 0 ms; Standard Deviation: 20 ms
- **120 bpm**
Mean: 7 ms; Median: 6 ms; Standard Deviation: 26 ms

On the other hand, in figure 5.3 it can be seen how, as the metronomic tempo increases, deviations in MIDI ticks tend to increase in significance, and particularly in standard deviation. Specifically, the main parameters are as follows:

- **60 bpm**
Mean: -2 ticks; Median: -2 ticks; Standard Deviation: 22 ticks
- **80 bpm**
Mean: 9 ticks; Median: 9 ticks; Standard Deviation: 26 ticks

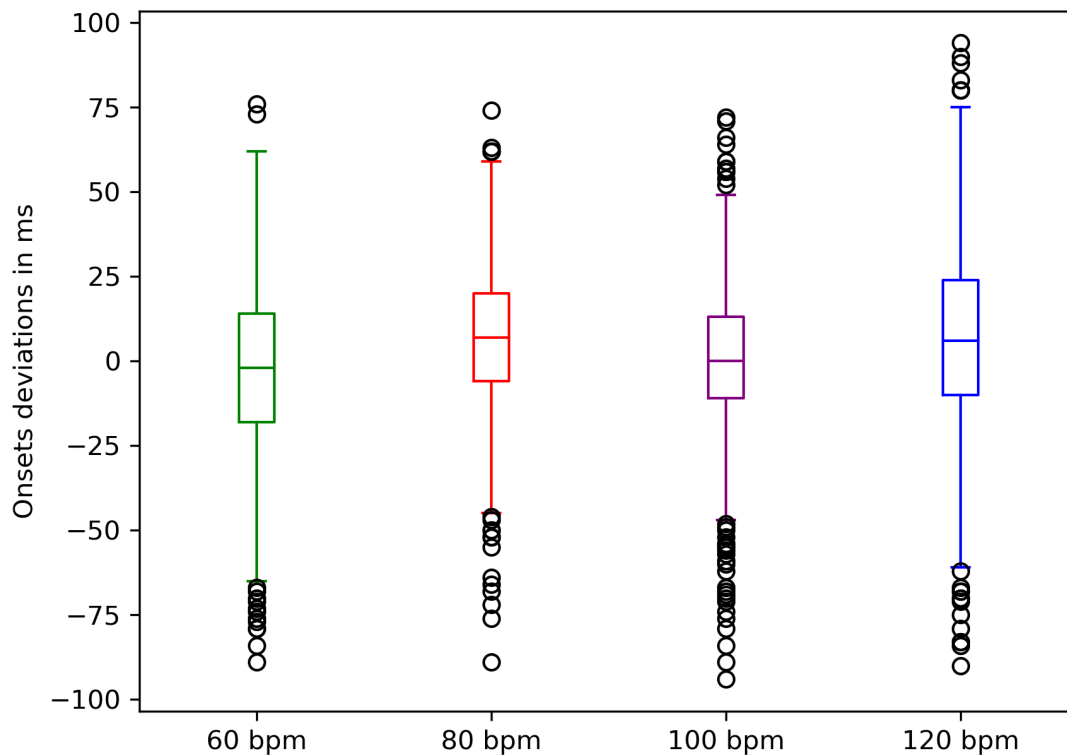


Figure 5.2: Boxplots representing the distribution of time deviations of note onsets, in ms, for each metronomic execution tempo of the exercises.

- **100 bpm**

Mean: 0 ticks; Median: 0 ticks; Standard Deviation: 32 ticks

- **120 bpm**

Mean: 14 ticks; Median: 12 ticks; Standard Deviation: 51 ticks

With this in mind, I decided to include time values in ms instead of MIDI ticks in the next steps of analyzing and testing predictive models.

5.3.2 Position and string change, versus preceding rest

Another element that I decided to investigate on a descriptive level was the relationship between changes in position or string and the extent of any pause preceding the note observed. In other words, I tried to understand if and to what extent the movements along and across the neck of the fretting hand force the musician to interrupt a note before its theoretical conclusion, in order to reach the next note.

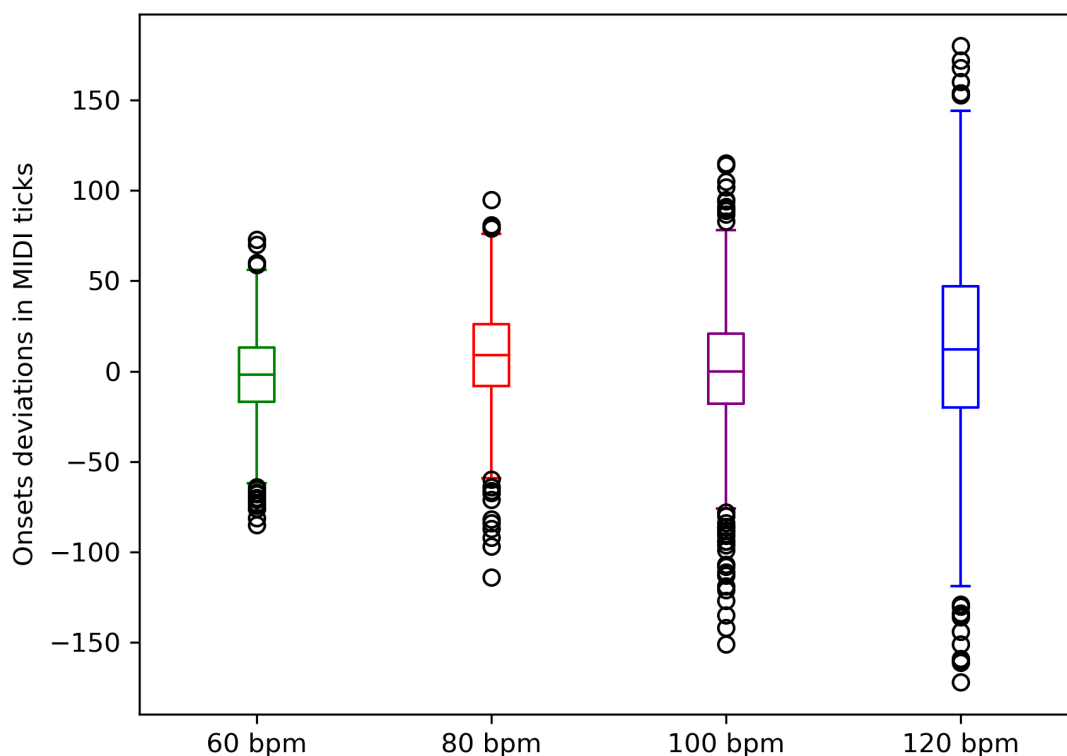


Figure 5.3: Boxplots representing the distribution of time deviations of note onsets, in MIDI ticks, for each metronomic execution tempo of the exercises.

In Figure 5.4 the relationship between position changes and any pause preceding the note is described graphically. The data found are quite similar for shifts in the direction of the neck or in the opposite one, and a general tendency to increase the duration of the pause preceding the note in step with the extent of the fretting hand shift can be observed.

In Figure 5.5 the relationship between string changes and any pause preceding the note is described graphically. Movements across the neck downwards (toward the thinner strings) have a greater influence on the need to close the previous note earlier. Movements in the opposite direction, on the other hand, tend to have less significant influence, with a general tendency to increase the dispersion of the data as the magnitude of the displacement increases.

5.3.3 Correlation matrix

Before constructing and testing predictive models, I finally developed a correlation matrix to highlight any linear relationships between the variables (see Fig. 5.6).

The most obvious and significant correlation is between `delta_onset_ms-1` and

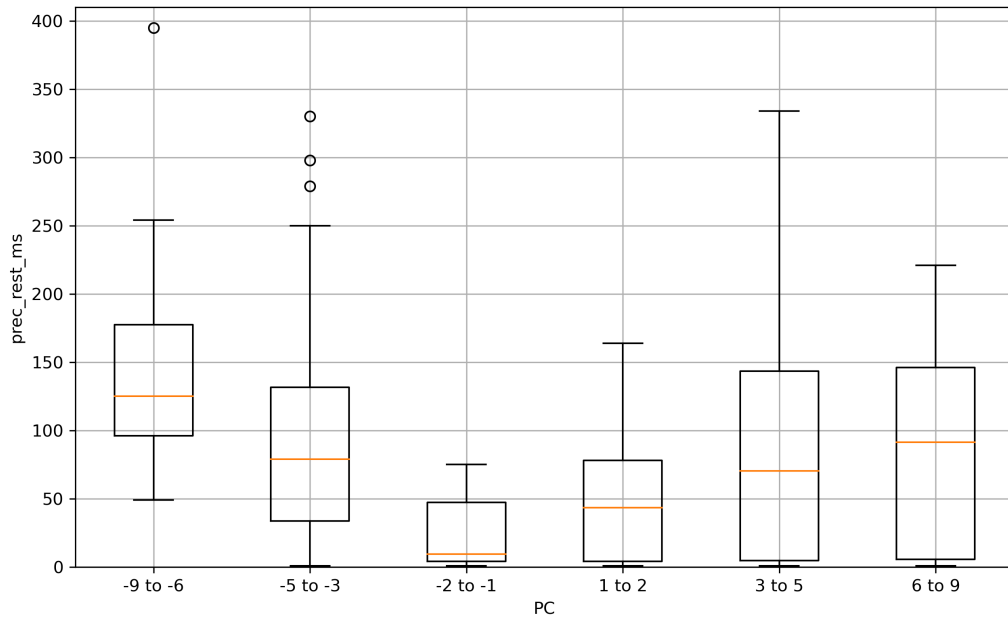


Figure 5.4: Boxplots representing the size of the pause preceding the note in relation to position changes.

delta_onset_ms, but numerous other direct or inverse correlations of modest but still potentially significant magnitude can be observed.

5.4 Predictive models

In order to evaluate the best solution to include in the expressive model for the prediction of timing and loudness deviations, I tested the predictive ability of six different solutions, each of which utilizes the following features: note_numb, onbeat, PC, SC, HS, hammer-on, pull-off, bending, velocity-1, IOI_ms, delta_onset_ms, delta_onset_ms-1. Random states and seeds were set for reproducibility. More specifically, to predict onset (delta_onset_ms), preceding rest (prec_rest_ms), and velocity deviations, I tried:

- a Decision Tree regression model. The dataset was shuffled and divided into training (80%) and test (20%) sets. The Decision Tree has a maximum depth of 10. The model's performance was evaluated using Mean Squared Error (MSE) and R-squared on both the training and test sets, and a graphical comparison of the real versus predicted values was generated for the first 100 points in the test dataset;
- a Random Forest regression model. The dataset was shuffled and divided into training

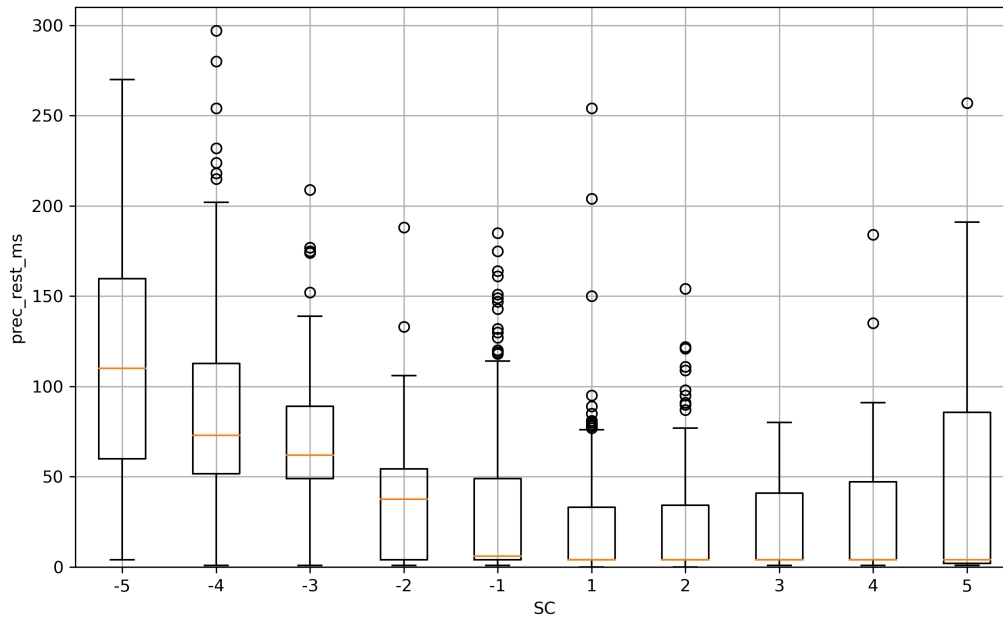


Figure 5.5: Boxplots representing the size of the pause preceding the note in relation to string changes.

(80%) and test (20%) sets. The model's performance was evaluated using Mean Squared Error (MSE) and R-squared on both the training and test sets, and a graphical comparison of the real versus predicted values was generated for the first 100 points in the test dataset;

- an RNN regression model. The dataset was shuffled and divided into training (80%) and test (20%) sets. The model consists of one RNN layer with 40 units, trained for a maximum of 500 epochs with early stopping after five epochs of no improvement to prevent overfitting. Only one time step is used. The model's performance was evaluated using Mean Squared Error (MSE) and R-squared on both the training and test sets, and a graphical comparison of the real versus predicted values was generated for the first 100 points in the test dataset;
- an RNN regression model similar to the previous one, but with the inclusion of four time steps;
- an LSTM regression model. The dataset was shuffled and divided into training (80%) and test (20%) sets. The model consists of one LSTM layer with 40 units, trained for a maximum of 500 epochs with early stopping after five epochs of no improvement to prevent overfitting. Only one time step is used. The model's performance was evaluated using Mean Squared Error (MSE) and R-squared on both the training and test sets, and a

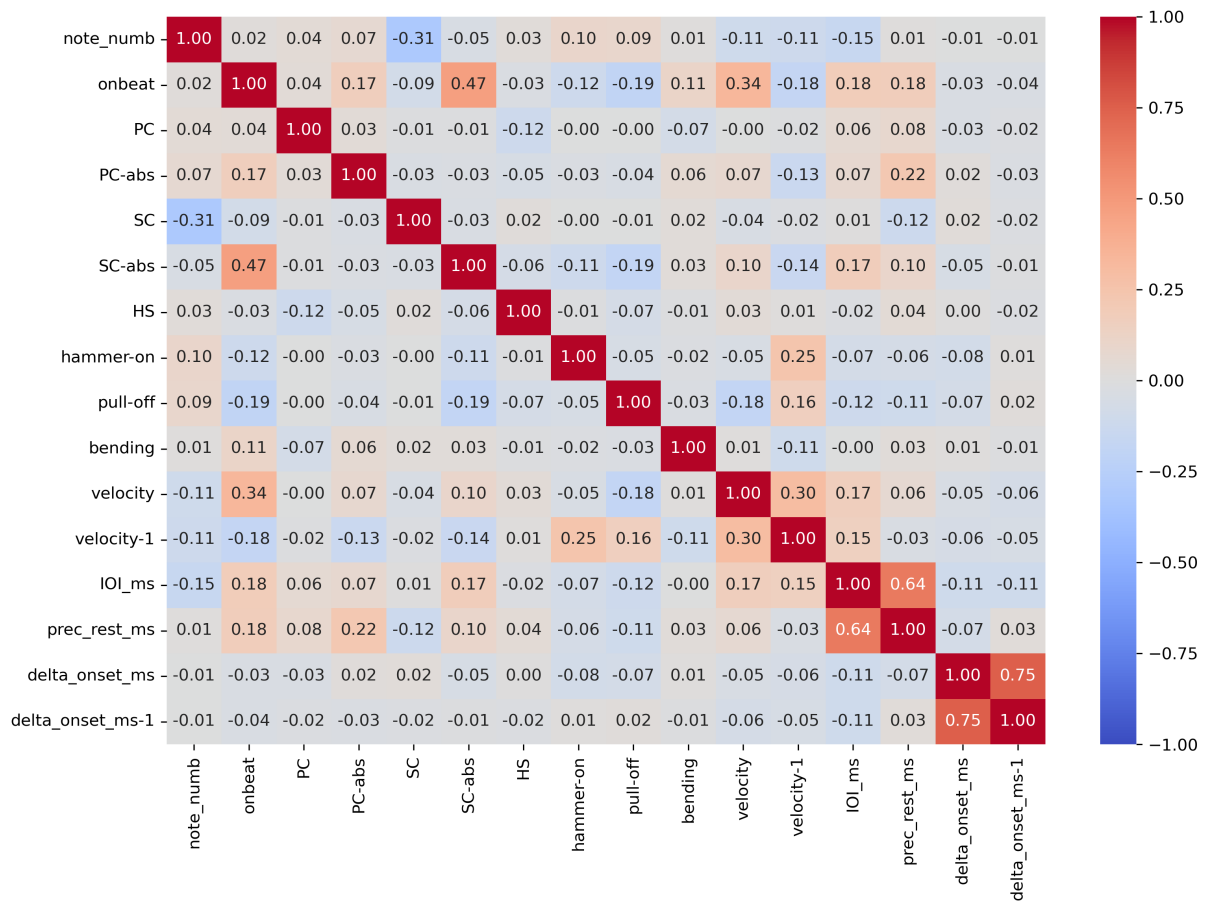


Figure 5.6: The correlation matrix.

graphical comparison of the real versus predicted values was generated for the first 100 points in the test dataset;

- an LSTM regression model similar to the previous one, but with the inclusion of four time steps.

I used both RNN and LSTM models with a single time step and with four time steps to evaluate whether there were any advantages to considering past observations in the sequence. In RNN and LSTM models I experimented with adding regularization techniques to reduce overfitting, such as including dropout layers and applying L2 regularization. However, to strike a balance between preventing overfitting and maintaining predictive capabilities as good as possible, I found that early stopping combined with using a small batch size ultimately provided the best results. I also experimented with different learning rates, starting with a relatively low rate of 0.001 using the Adam optimizer to ensure smooth convergence. I also tested various numbers of units. After several trials, a learning rate of 0.0001 and 40 units proved to be the most effective combination. Root Mean Squared Error (RMSE) and R-squared data for the training

and test samples, associated with parameters `delta_onset_ms`, `prec_rest_ms`, and `velocity`, are also given in tabular form in the Appendix (see page 147) to allow easy numerical comparison between the different solutions.

5.4.1 Onsets

The Decision Tree-based regressor generated good predictions, as shown in Figure 5.7. The Mean Squared Error on the test set was 238,46, and the R-squared was 0,56. On the train set the Mean Squared Error was 167,28, and the R-squared 0.70.

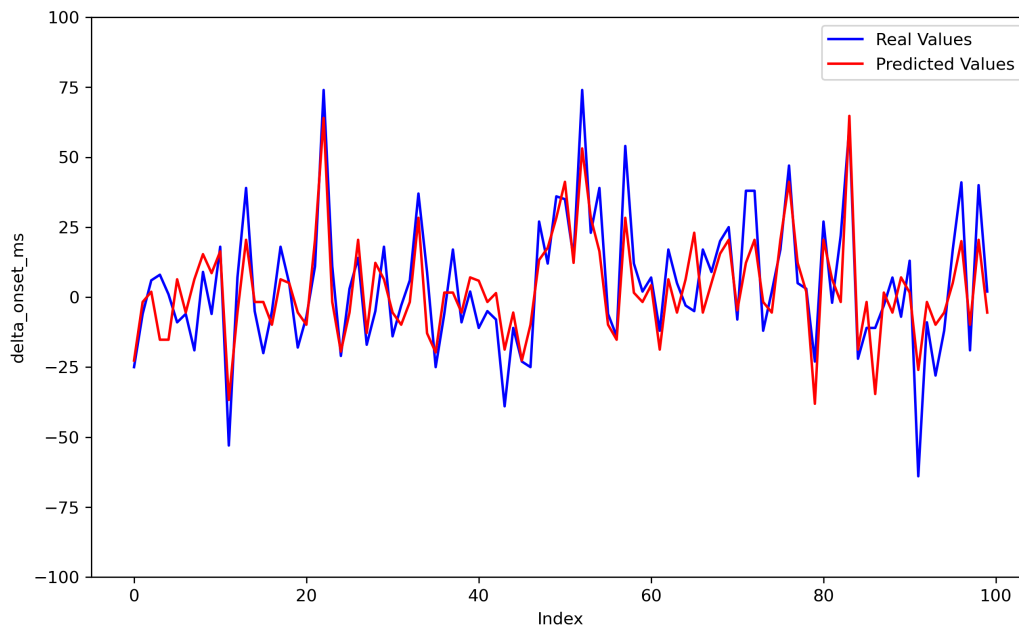


Figure 5.7: Comparison of real and predicted values of `delta_onset_ms`, based on the Decision Tree regressor.

The Random Forest-based regressor generated even better predictions, as shown in Figure 5.8. The Mean Squared Error on the test set was 206,00, and the R-squared was 0,62. On the train set the Mean Squared Error was 150,75, and the R-squared 0,73.

The RNN with only one time step generated worse predictions than the ones given by the Decision Tree and the Random Forest approach, as shown in Figure 5.9. The Mean Squared Error on the test set was 282,51, and the R-squared 0,48. On the train set the Mean Squared Error was 256,96, and the R-squared 0,53.

The RNN with four time steps generated similar predictions, as shown in Figure 5.10. The Mean Squared Error on the test set was 235,51, and the R-squared was 0,57. On the train set the Mean Squared Error was 206,02, and the R-squared 0,62.

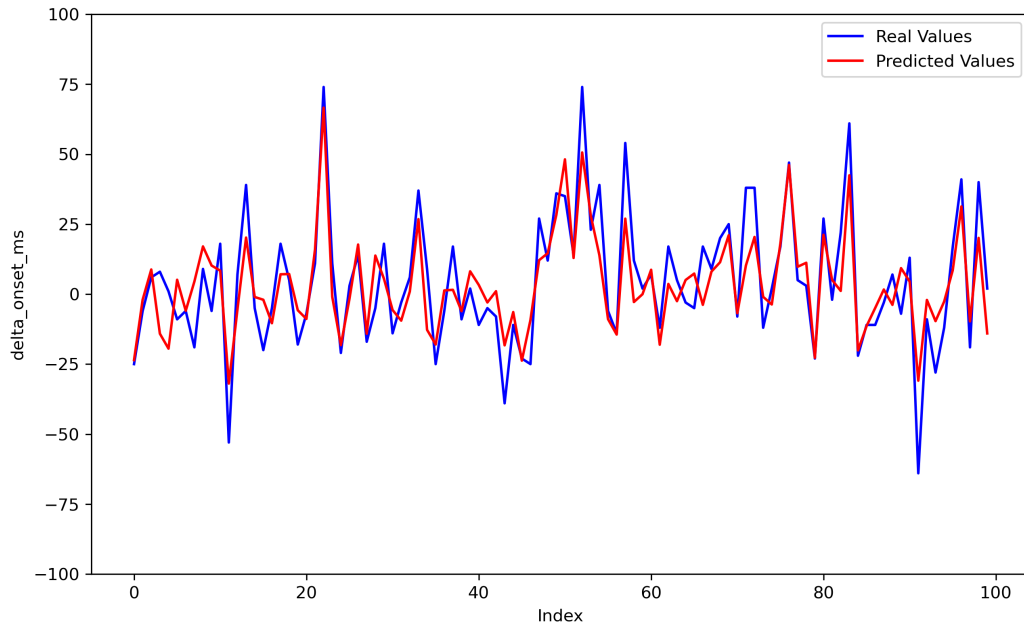


Figure 5.8: Comparison of real and predicted values of `delta_onset_ms`, based on the Random Forest regressor.

One time step and four time step LSTM performed similarly to the RNNs. In the first case, the MSE and R-squared values were 277,66 and 0,49 on the test set, while on the train set they were 248,60 and 0,54 (Figure 5.11). The four time steps LSTM gave values of 235,19 and 0,57 for the test set MSE and R-squared, and 212,69 and 0,60 for the train set (Figure 5.12).

In conclusion, we can observe how, in this case, the most precise regression algorithm turned out to be Random Forest. That was then used in the expressive end-to-end model described in Chapter 6. In the neural networks solutions the use of one or four time steps produced only moderate improvements, a sign that in this specific context the timing deviations are basically local. It cannot be ruled out that a larger dataset could lead to an improvement in the performance of the neural networks, but the distance to the predictive capabilities of the Random Forest solution is not negligible, and the latter appears more than adequate for the specific application field.

5.4.2 Offsets (preceding rest)

Only the best-performing prediction solution identified from the six available will be reported below for predicting notes' offsets and velocity. The data for the other options and their graphical representation are included in the Appendix.

In the case of the prediction of note offsets (as a consequence of the prediction of the pause

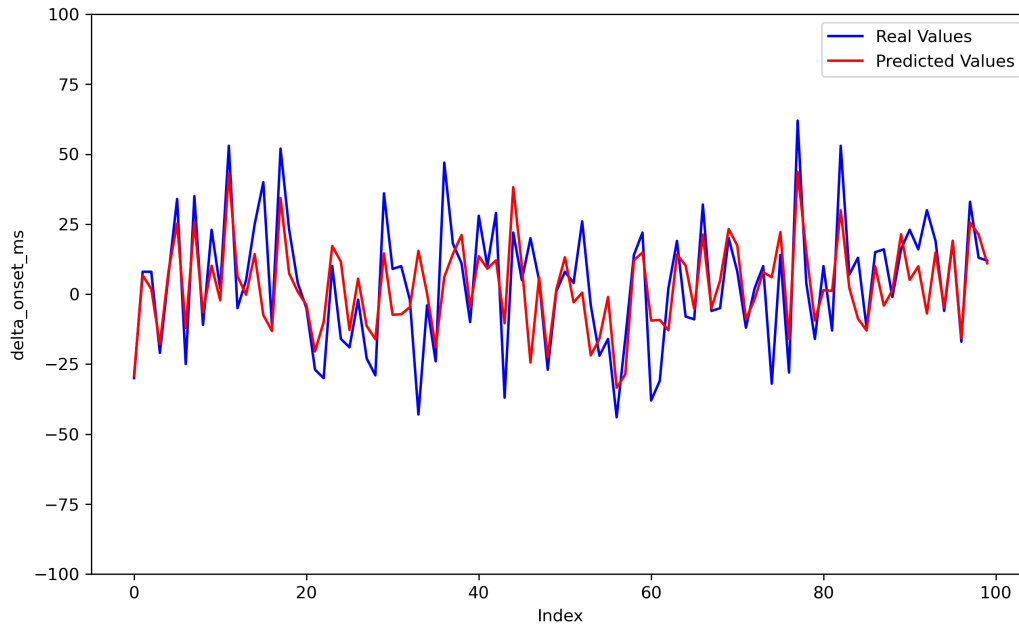


Figure 5.9: Comparison of real and predicted values of `delta_onset_ms`, based on the RNN with one time step.

preceding each note), the RNN with four time steps proved to be the best-performing solution, which was then included in the expressive model described in Chapter 6 (see Figure 5.13). On the train set the Mean Squared Error was 2311,57, and the R-squared 0,52. On the test set the Mean Squared Error was 3179,60, and the R-squared 0,39. Offsets are probably less accurate than onsets because they can be modulated more freely without easily introducing the perception of “notes out of time” to the listener, and the musician, therefore, may pay less attention to them. They are also more complicated to recognize automatically. In this case, as can be seen in the Appendix, the use of one or four time steps in the neural networks produced very significant differences, both in the RNN and the LSTM.

5.4.3 Velocity

In the case of the prediction of notes velocity, the RNN with four time steps and the LSTM with four time steps proved to be the best-performing solutions (with the RNN non-significantly outperforming the LSTM with four time steps). The RNN, also because of its lower computational requirements, was then included in the expressive model described in Chapter 6. Also in this case, as can be seen here and in the Appendix, the use of one or four time steps in the neural networks produced very significant differences. The four time steps RNN Mean Squared Error on the test set was 109,82, and the R-squared 0,54. On the train set the Mean

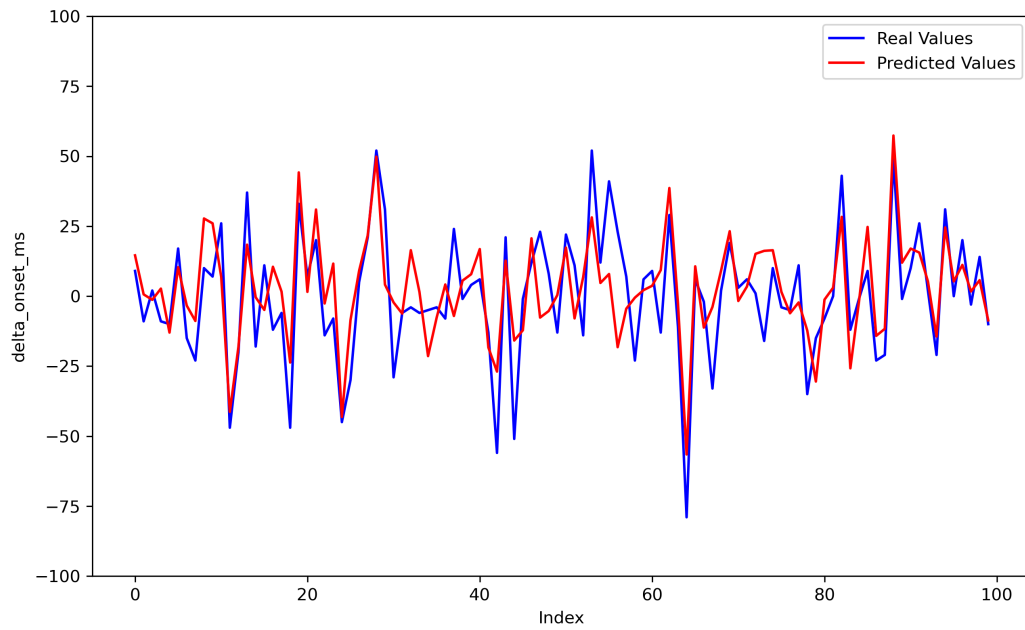


Figure 5.10: Comparison of real and predicted values of `delta_onset_ms`, based on the RNN with four time steps.

Squared Error was 112,85, and the R-squared 0,56 (see Figure 5.14).

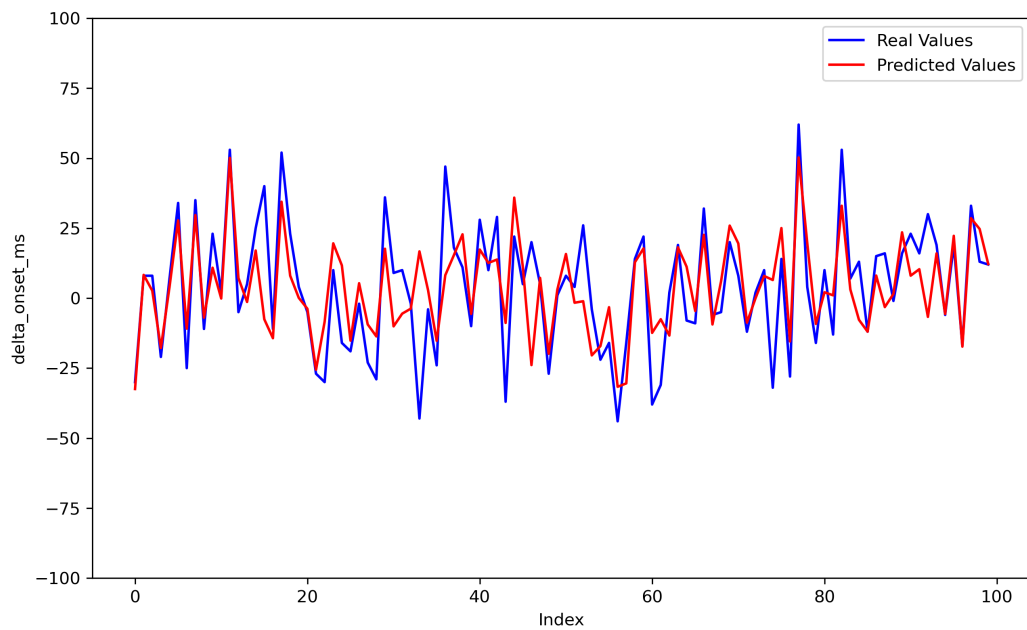


Figure 5.11: Comparison of real and predicted values of delta_onset_ms, based on the LSTM with one time step.

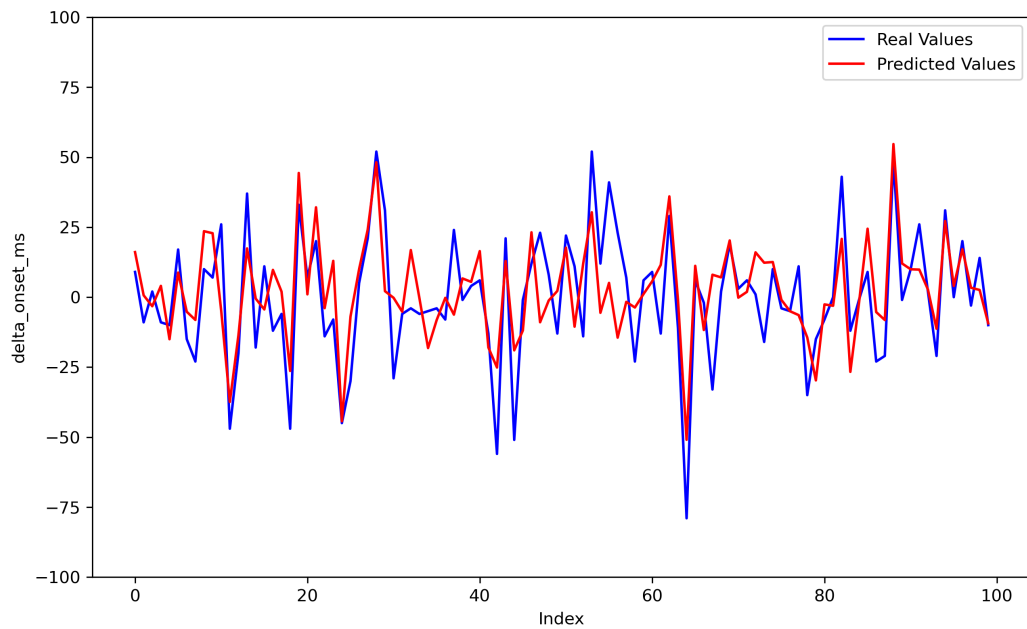


Figure 5.12: Comparison of real and predicted values of delta_onset_ms, based on the LSTM with four time steps.

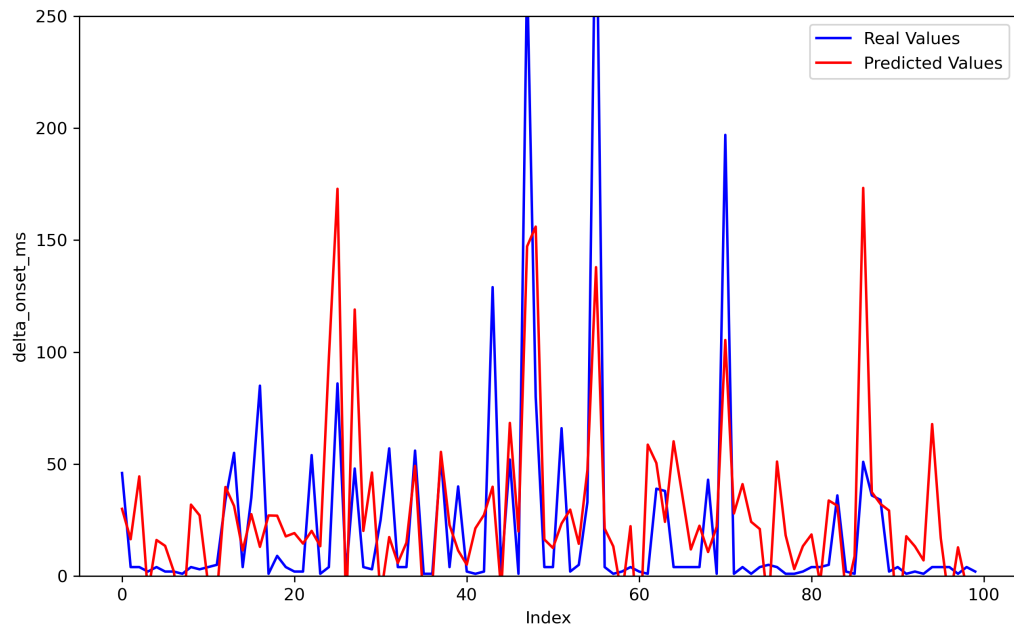


Figure 5.13: Comparison of real and predicted values of `prec_rest_ms`, based on the RNN regressor with four time steps.

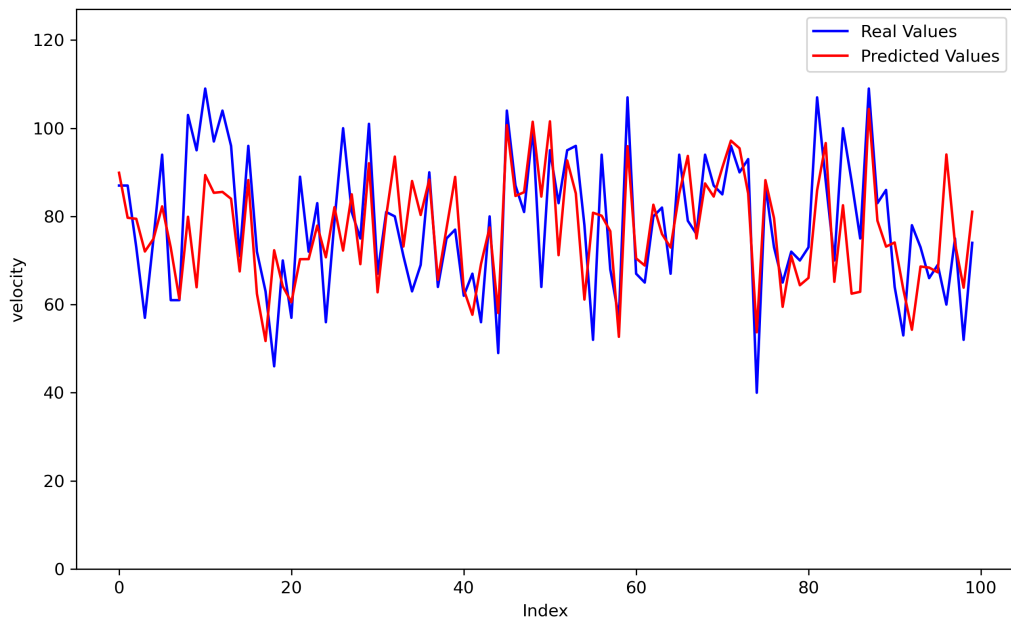


Figure 5.14: Comparison of real and predicted velocity values, based on the RNN with four time steps.

Chapter 6

Control of intentional expressiveness

Chapter introduction

Having reached this point, it has been seen how, starting from a melodic line, it is possible to automatically generate the guitar fingering, insert performance techniques, and generate regressors for the calculation of the unintentional deviations in the timing of the notes' onset and offset, and loudness (the latter in the form of MIDI velocity). The next and last step, the one addressed in this chapter, is to superimpose the voluntary modulation of the same expressive parameters and include all the steps described above in a single end-to-end software solution. Specifically, reference was made to the recording studio context in popular productions. The user of the expressive model finds its parallel in the artistic producer, who gives expressive directions to the performer (i.e., in the flow of the expressive model, to the already partially elaborated musical part, up to the insertion of unintended expressiveness). The types of indications included are based on what emerged in the interviews reported in Section 1.4. The user is then entrusted with the choices regarding general loudness and the positioning of the performance in relation to the beat.

6.1 Conductor systems

As observed in Section 1.5.6, solutions for computational music expressiveness that include user interaction are known as conductor systems. In these systems, users can intervene in the expressive rendering of a musical piece either asynchronously or in real-time. Many of these systems are presented with an emphasis on the fact that they generally allow even those without musical training to interact with music, guiding its expressive deviations.

In this regard, a particularly representative case, in addition to those mentioned previously,

is that of the Expression Synthesis Project (ESP), described in Chew et al. (2005) [152]. There, expressive performances are generated through an intuitive driving interface. Users control a virtual car navigating along a virtual road representing music development. The interface uses physical components such as pedals and a steering wheel to control the car's dynamics and the musical interpretation, creating a tactile connection between the user and the performance. The visual display provides a first-person view from the driver's seat, showcasing the road and dashboard, which enhances the immersive experience. As already seen, other approaches have instead based the control of the expressive pattern on positioning in two-dimensional planes (often based on Russel's circumplex model [67]) or on the use of gestures (as in the use of the Theremin [100] or the conductor's baton [78], [92], [95], [96]).

In the case of the model presented in this thesis, I tried to maintain an approach to interfacing with the user as friendly as possible, studying how to include the desired control possibilities without requiring the user to have music-specific advanced knowledge. The two parameters I decided to make accessible to the user, based on what emerged in Section 1.4, were the notes' average loudness and the positioning with respect to the beat. To be understood, those require two levels of musical competence that are very distant from each other. On the one hand, loudness is easily understandable; on the other, positioning in relation to the beat is a concept that is often obscure even to amateur musicians. Given that I opted for a traditional software GUI, I, therefore, decided to offer the user only one intervention mode with regards to loudness (a five positions slider) and two with regards to time positioning with respect to the beat (a slider for continuous control or, alternatively, the choice of one among the three options *pressing*, *precise* and *relaxed*). Before analyzing the GUI of the end-to-end model, how it works, and the characteristics of its output file, it is considered appropriate to present the logic followed for the calculation of timing and loudness deviations based on the preferences set by the user¹.

6.2 Loudness control

In the expressive model, timing and loudness deviations are calculated downstream of unintentional expressivity deviations. This is done by intervening on the Pandas dataframe created in the previous steps, and populated with all relevant information.

The five positions of the loudness slider input the value of that into the model. All previously calculated velocity values are normalized based on that value:

- 0: the highest velocity value present in the section is scaled to 90, and all other values are proportionally adjusted;

¹The code related to this chapter is available at https://github.com/pierluigi-bontempi/PhD_thesis/tree/main/Chapter_6_Control_of_intentional_expressiveness.

- 1: the highest velocity value present in the section is scaled to 100, and all other values are proportionally adjusted;
- 2: the highest velocity value present in the section is scaled to 110, and all other values are proportionally adjusted;
- 3: the highest velocity value present in the section is scaled to 120, and all other values are proportionally adjusted;
- 4: the highest velocity value present in the section is scaled to 127, and all other values are proportionally adjusted.

6.3 Onsets timing control

Regarding onsets placement with respect to the beat, the slider allows users to define the average value of timing deviations for the note onsets, with 1 ms increments ranging from -15 ms to +15 ms. The three predefined settings *pressing*, *precise*, and *relaxed* adjust the value to -10 ms, 0 ms, and +10 ms, respectively.

The two control modes are mutually exclusive. Selecting one of the categorical options resets the slider to its starting position. The slider is intended to offer the user the possibility of fine-tuning the average positioning of the onsets with respect to the beat. The categorical options, deliberately labeled with terms that are not part of the technical musical jargon, have been included to allow users without adequate music training to use the model.

After calculating the mean value of the deviations of the onsets from the input melody (this is assumed to be quantized), the system calculates the difference between this and the desired mean value and uniformly modifies all onsets and offsets accordingly.

Clearly, if the generated expressive MIDI File is played back in solo, no difference in timing will be perceptible due to this modification. However, the modification's effects will become evident when listening to the MIDI file alongside a metronome track or incorporating the MIDI file into a musical production that includes other instruments.

6.4 The end-to-end implementation of the expressive model

6.4.1 Overall structure

The end-to-end implementation of the model was realised in Python, using the additional libraries Mido, Pandas, Numpy, Cplex, Joblib, Tensorflow. I divided the various tasks of the software into a series of dedicated Python files, all of which are then called within main.py. Part of these files has been taken from what was presented in the previous chapters, with the

modification, with regard to the generation of the fingering and the automatic insertion of articulations and expressive techniques, of the removal of the possibility of customization by the user, as already mentioned. This was in order to make the user experience as simple and straightforward as possible.

Below is a list of the Python files that make up the software, in the order in which they are called up, together with a brief description of them.

main.py

This is the core of the software. All other files are imported here, and their respective functions are executed from this point in the predefined order.

gui.py

This is responsible for the graphical interface, which was developed using Python's native library Tkinter. It requires loading the quantized MIDI File, which is to be rendered expressively, and input of data related to the scale, root, and user preferences for loudness and positioning of the onsets relative to the beat (the topic will be dealt with in more detail in Section 6.4.2).

MidiFile_parser.py

This performs the parsing of the input MIDI File, extracting relevant data for generating its expressive version (e.g., notes' onsets and offsets delta time, MIDI note number, etc.). It then generates a Pandas dataframe containing this information.

CPLEX_data_generator.py

Based on the dataframe generated in the previous step, this creates a .dat file containing the instructions for the subsequent automatic fingering generation, correctly encoded for the CPLEX model.

CPLEX_fingering_generator.py

This uses the CPLEX Python API to provide the solver with the two necessary files: CPLEX_data_file.dat (generated in the previous step) and CPLEX_model.mod (the mathematical model in OPL, which also includes a JavaScript section for exporting the results in CSV format). In the '.dat' file, a guitar in standard tuning (E2-E4) with 24 frets is assumed. The settings related to the solver (which balance the various discomfort factors) are as follows:

```

// Hand and comfort

SP_max = [1,1,2]; // maximum hand spread for fingers 2, 3, and 4
f_lb = 5; // lower fret of the handy interval
f_ub = 12; // upper fret of the handy interval
timePerFret = 20; // time in ms to move the hand, for each fret covered

// Discomfort converting factors

d_PC = 40; // position change
d_SC = 20; // string change
d_HS = 20; // hand spread
d_DH = 60; // distance from the handy position
d_NO = 40; // number of notes played on open strings\

// Balance between the different factors

alpha_PC = 0.2; // position change
alpha_SC = 0.1; // string change
alpha_HS = 0.2; // hand spread
alpha_DH = 0.3; // distance from the handy position
alpha_NO = 0.2; // number of notes played on open strings

```

add_techniques.py

This merges the data generated in the previous step with the dataframe created by `MidiFile_parser.py`. It then automatically inserts articulations and expressive techniques (based on the *MySongBook* stats), creating and populating the corresponding dataframe columns.

predict_deviations.py

This calculates the unintended deviations in timing and velocity based on the dataframe generated in the previous steps, using the offline versions of the previously trained predictive models (Random Forest and RNN with four time steps). For the first note, values that are not available but are necessary for subsequent predictions are generated randomly. In the case of the parameters `prec_rest_ms` and `velocity`, since the predictive models require data for the first four notes to start making predictions, these values are calculated for the second, third and fourth

note based on the PC and SC values (for `prec_rest_ms`, using the data outlined in Figures 5.4 and 5.5) or randomly within a range of 80-100 (for velocity).

`df_update_with_predictions_and_user_prefs.py`

This modifies the dataframe created in the previous steps by calculating and applying intentional deviations in timing and loudness, based on the user preferences provided earlier and as outlined in Sections 6.2 and 6.3. Data validity checks are also performed, with interventions made if necessary (for example, setting the delta time of the onset of the first note to 0 if it results to be less than 0, preventing the predicted value of `prec_rest_ms` from completely removing the previous note if it exceeds its duration, and avoiding note overlap: the delta time between the note-off of one note and the note-on of the next is constrained to be ≥ 0).

`updated_MidiFile_generation.py`

Based on the updated dataframe and the data previously extracted from the input MIDI File, this generates a new MIDI File that includes the expressive timing and velocity-related modifications, and the information related to fingering, articulations, and expressive techniques in the form of MIDI Control Change messages (the topic will be dealt with in more detail in Section 6.4.3).

6.4.2 The GUI

As anticipated, the graphical interface was designed to be as simple and intuitive as possible (see Figure 6.1).

The user is asked to select the quantized MIDI File, which will be made expressive, by clicking on the *Select MIDI File* button and then selecting the file in the filesystem using the mouse. By checking the extension, it is automatically verified that a MIDI File, and not a file of another kind, has actually been selected.

Root and *Scale* must be selected from dropdown lists because, for the automatic insertion of bends, the system needs to know which notes can be used as starting points to reach the target notes (i.e., those contained in the input melody) via bending. All notes of the tempered system can be selected as the Root. The most common scales have been implemented: major, natural minor, major pentatonic, minor pentatonic, and blues scale.

As previously described, the options for selecting loudness and positioning relative to the beat follow.

Finally, by clicking the *Get expressive!* button, if all data have been entered correctly, the sequence of operations that leads to the generation of the expressive MIDI File is initiated (see

Figure 6.2).

6.4.3 The output MIDI File

The system generates an output MIDI File based on the input one (assumed to be quantized), with timing and velocity deviations applied as previously described. The MIDI File also contains information regarding string, fret, and expressive articulations/techniques used, represented as Control Change (CC) messages. These CC messages are positioned immediately before the note-on message of each note. The CC values are reset immediately after the note-off message of the note to which they apply (see Figure 6.3).

Specifically, the following Control Change (CC) messages have been used (selected from those listed as “Undefined single-byte controllers” in the MIDI 1.0 specifications)²:

- CC102: string number (scaled by a factor of 10 for improved visualization in graphical MIDI editors);
- CC103: fret number (scaled by a factor of 5 for improved visualization in graphical MIDI editors);
- CC104: hammer-on (assigned a value of 127 if present, and 0 if absent);
- CC105: pull-off (assigned a value of 127 if present, and 0 if absent);
- CC106: bending (expressed in tenths of a semitone; for example, a bend of two semitones corresponds to a CC value of 20);
- CC107: vibrato (assigned a value of 127 if present, and 0 if absent);
- CC108: slide legato (assigned a value of 127 if present, and 0 if absent).

6.5 Sound-based control

Although, in the development of the user interface for the model presented in this thesis, I relied on a traditional GUI, with user data input through a mouse or keyboard, other options were also considered. Specifically, as a co-supervisor, I oversaw a bachelor’s thesis in Information Engineering³, which explored the possibility of controlling the parameters of computational expressive models through nonverbal audio input. The thesis demonstrates how control

²The rationale for not using the MIDI Pitch Bend message or CC 1, which is often used to control vibrato, will be clarified in Section 7.2.

³By Claudio Costantini, available at <https://hdl.handle.net/20.500.12608/57082>, last accessed: 2024/11/23.

parameters, particularly related to loudness, dynamic range, and the positioning of notes relative to the beat, can be extracted from audio materials.

This preliminary investigation of the topic may serve as a precursor for potential future developments of the model, particularly regarding its user interface. In particular, the use of vocal audio signals as input could potentially bypass, in whole or in part, the need for the user to have a music theory background.

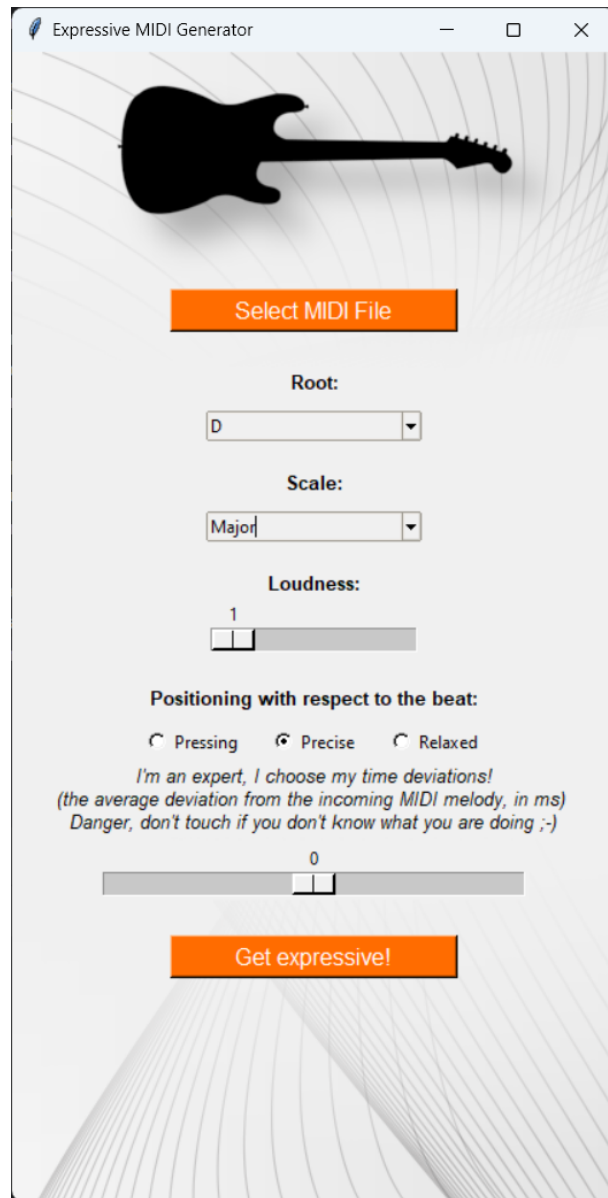


Figure 6.1: The GUI of the end-to-end implementation of the expressive model.

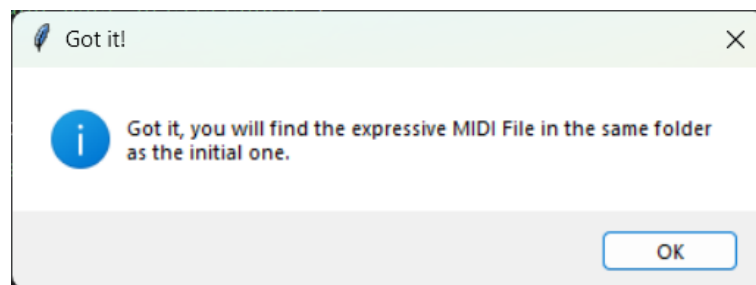


Figure 6.2: The confirmation window for the correct entry of input data and the initiation of the expressive MIDI File creation process.

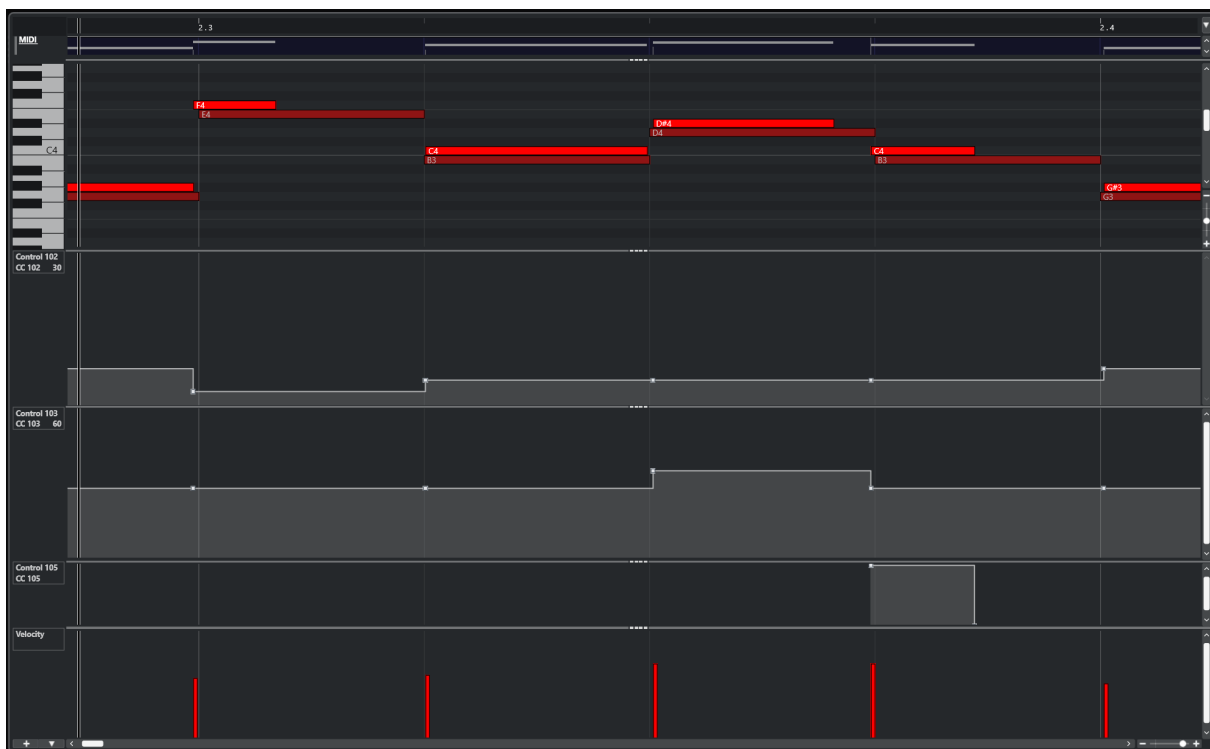


Figure 6.3: Comparison between the input MIDI melody and the one generated by the expressive model (transposed up by one semitone for greater clarity). Below the piano roll, the graphical representations of CC 102 (string), CC 103 (fret), and CC 105 (pull-off) are displayed, along with the velocity values at the bottom, all pertaining to the expressive MIDI file generated by the system.

Chapter 7

The next step: audio rendering with innovative virtual instruments

Chapter introduction

The construction of the expressive model presented in this thesis was based on a form of symbolic representation of musical content, specifically the MIDI format. This is fundamentally justified by respect for the natural distinction between the musician's gesture (determined by intentional and unintentional components, modeled here) and the actual sound performance, which is determined, in the real world, also by the physical characteristics of the instrument, the acoustic context, and the possible application of amplification and effects. This chapter, therefore, aims first of all to present the essential characteristics of the virtual instruments commonly used for the sonification of MIDI material, and specifically presents the peculiarities that a virtual instrument intended for the sonification of MIDI data generated by the expressive model presented in the previous chapters should include. At the same time, a reflection is proposed on the potentialities inherent in shifting part of the computational generation of expressivity from the symbolic domain to that of audio signal generation, a field previously explored in a decidedly less significant manner with respect to symbolic expressivity codification. Finally, the motivations for adopting the MIDI 1 protocol, rather than the more recent version 2, for the development of the expressivity model are discussed.

7.1 Digital Audio Workstations, virtual instruments and expressivity

In audio production, a virtual instrument (VI) is a piece of software designed to generate sound. VIs often emulate traditional musical instruments or synthesizers, but may also create original audio textures. They usually operate within digital audio workstations (DAWs) but can frequently also be used as stand-alone applications. They are typically controlled via MIDI [153].

The history of VIs can be traced back to 1996, when Steinberg's Cubase, one of the pioneering digital audio workstations, introduced the Virtual Studio Technology (VST) interface, which allowed the integration of audio effects and, three years later, software instruments (VSTi - Virtual Studio Technology instrument) directly into the DAW environment. This innovation paved the way for other DAWs, like Avid Pro Tools and Apple Logic Pro, to incorporate features like MIDI sequencing and instrument tracks, providing comprehensive support for virtual instruments. The VST standard is still one of the most widely used platforms for creating virtual instruments and audio effects [154], along with other standards such as AAX and Audio Units.

VIs can be based on sampling sound sources, as well as on sound synthesis techniques such as subtractive, additive, granular, frequency modulation (FM), or physical modeling [155]. VIs frequently allow for extensive manipulation of parameters like pitch, dynamics, and timbre, making them a versatile and cost-effective alternative to traditional instruments in both studio and live performance settings.

Commercially available virtual instruments are mostly focused on the inclusion of significant sound generation and manipulation capabilities, but not on the automatic generation of expressiveness. In most cases, the generation of expressive sound renditions is left to skillful manipulation by the producer of the MIDI signal with which the virtual instrument is controlled [3]. Nonetheless, understanding the capabilities of generating deviations and expressive variations in the generated sounds, and the ways in which these goals are or could be achieved, are of fundamental importance in defining how the expressive potential of the MIDI Files generated by the model described in the previous chapters could be fully exploited.

7.1.1 Tools commonly available in VIs

Most modern virtual instruments allow users to control basic parameters such as equalization (EQ) and high-pass filter (HPF) using MIDI messages and/or DAW automations. These controls often include adjustable HPF resonance, attack and release manipulation, and, in some cases, compression. Advanced virtual instruments may offer additional sound control options,

providing a wide range of sonic capabilities.

Sampled instruments typically feature multiple samples for each note, selected based on the MIDI velocity of the played one. To prevent the "shotgun effect" - the unnatural repetition of identical sounds - sample-based virtual instruments often employ the *round-robin* technique. This solution cycles through different samples randomly or in a set order when the same note is played multiple times.

Usually, DAWs also allow randomizing MIDI data within a user-defined range, which can help mitigate the mechanical feel of quantized, fixed-velocity parts [156]. The limitation of simple randomization techniques and their comparison with more structured approaches in order to generate the perception of performative expressiveness in the listener was addressed in Worrall et al. (2024) [157].

These expressive tools can be activated or used either offline or in real-time, using offline editing or MIDI controllers (usually with CCs) [158].

7.1.2 Multiple articulations-based VIs

In contemporary music production, the term *articulation* refers to the different playing techniques or timbres that may be associated with a virtual instrument. For example, a virtual violin may be able to perform the same notes using various articulations, such as *legato*, *staccato*, or *pizzicato*. This flexibility allows for a more realistic and expressive musical performance rendition.

Articulations are particularly common in sample-based VIs designed to emulate classical orchestral instruments, including strings, brasses, and woodwinds. However, this approach is not limited to that area and can be found in virtually any virtual instrument. Major manufacturers like EastWest, IK Multimedia, Vienna Symphonic Library, Steinberg, Native Instruments, and Spitfire Audio offer commercial products including articulations. In a similar way to what was seen in Section 7.1.1, controlling articulations typically requires manual intervention by the producer, who must specify the desired articulation for each passage.

In most cases, articulations are triggered using note-on messages at pitches outside the instrument's playable range (see, for example, Figure 7.1). Alternatively, they can be activated via Control Change (CC) or Program Change messages.



Figure 7.1: The orchestral library Miroslav Philharmonic Orchestra, produced by IK Multimedia. The notes outside the violin’s range to be used to trigger the available articulations (sustain, detache, etc.) are visible in the secondary keyboard located above the main one, bearing the words KEY SWITCHES.

7.2 Expressiveness at a symbolic level or at the level of generation and manipulation of the audio signal?

A notable exception to the approaches described in the previous Sections is that of the Synful Orchestra, which operates on a different principle, as detailed by its creator, E. Lindemann [159] (see Figure 7.2). Unlike traditional sample-based instruments, the Synful Orchestra does not rely on PCM audio samples (except for the noise components of the sounds). Instead, it uses additive synthesis, generating sound based on harmonic amplitude envelopes extracted from real instrument recordings. Here, the system derives slowly varying pitch and amplitude controls directly from MIDI input and uses them in additive synthesis to produce sound. To handle the rapid fluctuations in pitch and timbre that usually occur during note transitions, the Synful Orchestra employs a method called Reconstructive Phrase Modeling (RPM). This technique searches a database of instrumental phrases, identifies fragments similar to those requested via MIDI, modifies them accordingly, and combines them to create fluid and natural transitions

between notes. According to Lindemann, this approach addresses the main shortcomings of sampled instruments: achieving dynamic long notes that vary over time and ensuring smooth, natural transitions between notes, rather than a simple juxtaposition of separate sound events. The Synful Orchestra stands out as one of the few examples where expressive control is managed, at least partially, at the level of sound synthesis or generation rather than at the level of virtual instrument control. The software autonomously selects articulations based on incoming MIDI notes, eliminating the need for explicit articulation triggers. This autonomous operation offers a level of expressiveness not commonly found in traditional sample-based virtual instruments.



Figure 7.2: The orchestral library Synful Orchestra.

Even within the academic context, the concept of shifting the generation of expressiveness from symbolic representations of music to the sound generation phase, or even to manipulating existing sounds, has not been extensively explored, with a few notable exceptions. One such

example is a line of research pursued within the Centro di Sonologia Computazionale of the Department of Information Engineering of the University of Padua, with works published between 1998 and 2004, concerning the possibility of manipulating sonically non-expressive performances directly at the audio level through sound re-synthesis, with the aim of introducing the desired expressive intentionality [75], [160]–[162]. Part of this research then found application in the previously mentioned CaRo system [77]. In Arcos et al. (1998) [163] it is described a system for generating saxophone expressive musical performances, based on the manipulation of audio recordings of inexpressive interpretations of the musical phrases, played by real musicians.

In the specific case of the guitar, in Jonason et al. (2023) [164], a comparison between different neural synthesis systems for polyphonic acoustic guitar performances, based on the input of MIDI data on pitch, intonation, and string used, is proposed. Specifically, four different neural synthesizers were trained on hexaphonic and microphone recordings of acoustic guitar performances.

As described in Chapter 6, the expressive model presented here generates a symbolic description in MIDI format of the expressive guitar performance, complete with:

- time displacement of notes’ onset and offset;
- notes’ velocity;
- string and (possibly) fret used to play each note;
- possible use of specific articulations and expressive techniques (hammer-on, pull-off, bending, vibrato, legato slide);
- bending amplitude, if any.

Given the complexity and richness of the input data, to be adequately sonified, a virtual instrument based on an extremely high number of audio samples would be required, or alternatively, an audio sample manipulation system similar to those mentioned above could be used. Another solution would be the use of a physical model VI (see, among others, [165]–[169]).

In any case, it is relevant to emphasize that part of the expressiveness, in order to maximize its effectiveness, should be delegated to the actual generation of the audio signal. This is because, as on the other hand happens in the score, the symbolic representation of the musical part generated by the expressive model can only be partial with respect to the richness and complexity of the audio signal. Listed below, specifically, are possible interventions at the level of audio generation that could enrich and complete the expressive potential of the model:

- definition of the general timbre characteristics of the virtual instrument (in electric guitar, the sound is conditioned by the pickups, the structure of the instrument, the woods used, the strings, amplification, and effects);
- characterization of note attack transients (these can be varied according to velocity, as on the real instrument, but it could also be simulated, for example, the contribution given by the use of plectrums of different materials, shapes, and thicknesses);
- characterization of the release time of the notes and consequent generation of a more or less legato phrasing. In the expressive, monophonic model, the offset of each note is always prior to or coincident with the onset of the next. When offset and subsequent onset are simultaneous, the virtual instrument could implement strategies to maximize the fluidity of the passage. With data on the strings used also available, the VI could behave differently when the two notes are on the same string (overlap not possible) or on different strings (overlap possible). On the other hand, when there is a pause between one note and the next, the decay characteristics of the preceding note can markedly influence the perception of continuity or discontinuity between notes;
- definition of the sound characteristics related to the expressive techniques (frequency and amplitude of the vibrato, duration of the legato slide, duration of the bending phase of the note). Different guitarists or different stylistic contexts may interpret the actual execution of the same techniques differently. For this reason, it was decided not to encode the bending with a Pitch Bend MIDI message, which would then have constrained the virtual instrument to faithfully follow the course of the pitch change, and likewise not to associate the vibrato with CC1, which generally controls its amplitude in VIs. Using CCs without standard assignment and only marking the type of technique and note to which it is applied allows the virtual instrument to autonomously interpret the information according to its internal parameters.

By defining the sound characteristics described above, the VI could model part of the style of real guitarists or generate different virtual guitarists. It could also better accommodate the genre's stylistic conventions while maintaining the same input MIDI data.

7.3 Why MIDI 1?

MIDI (Musical Instrument Digital Interface) is a standard born in the early 1980s to unify communication between electronic musical instruments or other devices such as sequencers, computers, lighting controllers, mixers, etc [101]. The first version of the protocol, presented in 1983, comprised both hardware connections and software (data format) specifications,

introducing what in the following decades would have been, in fact, the reference protocol for interfacing electronic musical devices or, in any case, machines and software more or less directly linked to this context. Although MIDI was born as a protocol for exchanging data in real-time, specifications for offline saving of sequences of MIDI instructions were also adopted in 1988 with the birth of Standard MIDI Files. Since then, the protocol has undergone only minor updates (the latest version of the MIDI 1 specification¹ was released in 1996). Despite some inherent limitations of MIDI 1 (in particular, the limited bandwidth, and the application of CCs to all notes present on the channel, and not note-wise), the presentation of version 2 of the protocol, based on the previous one but profoundly renewed, only occurred in 2020, and initially only at the level of real-time data exchange [170]. The specifications for creating MIDI Files based on the MIDI 2.0 protocol were presented in June 2023.

Among the numerous innovations introduced by MIDI 2.0 we must mention:

- a better resolution of the messages (where MIDI 1, in most cases, uses 7 bits - and so 128 possible values -, MIDI 2 implements 32-bit messages);
- per-note controllers (specific controls, such as pitch bend, can be applied to individual notes rather than the entire channel);
- bidirectional communication between connected devices (which can query each other's capabilities, configure settings dynamically, and synchronize more effectively);
- significantly higher bandwidth when the connection (for example, USB 2) makes it possible.

Therefore, the question of why MIDI 2 was not used for the development of this thesis could arise spontaneously. The reasons are multiple. First of all, the still poor adoption of the new protocol was kept into consideration. At the end of 2023, only four DAWs supported, and partially, the real-time use of MIDI 2.0². For obvious reasons, the use of Standard MIDI Files based on MIDI 2 is today even more limited. On the other hand, the innovations introduced with MIDI 2 would not have brought clear benefits in the specific field of application. Also applying CCs at channel level instead of note-wise has no disadvantages here, as the model is monophonic. By working on MIDI files and being able to set their temporal resolution as desired (by setting the ticks-per-beat parameter), it was possible to achieve temporal precision in the order of a millisecond, which is more than adequate for the application context. Finally, even if MIDI 2 becomes established in the coming years, this guarantees full backward compatibility with the previous version of the protocol. Priority was therefore given to the ease of use

¹<https://midi.org/specs>, last accessed: 2024/11/19

²see <https://midi.org/new-midi-2-0-products-released-november-2023>, last accessed: 2024/11/19.

and dissemination of the model over the use of a newer protocol which, although valid and commendable, would not have offered real advantages in the specific field of application. The difficulty of MIDI 2 catching on and some possible alternative solutions were analyzed in Luke and Ludovico (2024) [171].

Chapter 8

Conclusion

8.1 Thesis path, objectives achieved, and possible future developments

This thesis addressed the research field of Computational Expressive Music Performance (CEMP), which deals with developing computational models capable of generating, in full autonomy or by interacting with the human operator, virtual performances of musical parts with expressiveness and comparable to those of a real performer. Many sectors of information engineering have historically been involved in the field. A notable area is affective computing, which focuses on enabling computers to recognize, express, and simulate emotions. Generally speaking, CEMP is part of Artificial Intelligence's vast and multifaceted area. Another important engineering field connected to computational musical expressiveness is human-computer interaction. Additionally, though less prominent, the field has engaged in biomechanical studies of musical performance and its replication through robotics.

The investigation first required a thorough review of relevant previous research and the current state of the art on the subject. Given the specific field of interest, computational expressiveness in electric guitar lead parts in popular music, it was also necessary to clarify the instrument's physical characteristics and its performance techniques. Having noted the substantial absence of academic reflection regarding performative expressiveness in popular music, it became necessary to investigate its nature through interviews with leading producers and musicians.

The proposed expressive model is based on the symbolic representation of musical and performance data in MIDI format. It receives the quantized MIDI melody as input (the digital analog of the score in the case of the human performer). A series of steps linked together generates a new MIDI File in which expressive timing and loudness deviations have been applied to the original melody. The generated MIDI file also includes information on the strings and frets

of the instrument virtually used for each note and on the articulations and expressive techniques introduced in the virtual performance. All this is achieved by modeling the choices a human performer has to make to play the part, and intentional and unintentional human expressiveness.

From a technical point of view, different approaches found space in the creation of the model: rule-based, linear mathematical programming, Machine Learning and Deep Learning were each used where most appropriate, interacting virtuously according to the overall objective. Furthermore, a new audio, MIDI, and CSV dataset was created relating to deliberately non-expressive guitar performances, which was then used for training the predictive ML and DL models.

Among the possible future lines of research that could arise from this work and enrich it, we should at least mention the development of automatic systems for the control of intentional expressiveness, also with particular regard to the interplay between the different instruments and the vocals (at present, the control of intentional expressiveness is delegated to the manual intervention of the operator), and the possibility of managing several simultaneous notes, an occurrence which, although not particularly frequent, is present in contemporary lead electric guitar playing. It will also be important to generate, using the expressive model, virtual performances to be subjected to statistically significant subjective validations. It would also be interesting to attempt, when possible, a comparison with other systems for generating fingering or expressive deviations, to detect any convergences or divergences.

Alongside the central theme of the research, a parallel line was followed, related to the automatic identification of expressive passages encoded in metronomic paper rolls for player piano.

Both the guitar computational expressiveness and the player piano lines of research have led to collaborations with Italian and foreign scholars, to publications in prestigious international scientific journals, and to the presentation of part of the work at the Sound and Music Computing Conference (SMC) held in Porto in 2024. All code and the dataset produced have been released publicly. The research carried out for the thesis has also led to new studies currently underway and future projects of academic relevance.

8.2 Potential fields of application

8.2.1 Music Production

The primary application field of the research presented here, and the one that inspired the model's structure, is contemporary music production. While virtual instruments available on the market today offer significant timbral variety (see Sections 7.1.1 and 7.1.2), expressive performance generation is typically handled through real-time or offline user control of

expressive nuances, rather than by automatic or semi-automatic systems ([3], [158]).

Research into innovative controllers capable of translating performer gestures into expressive instructions is particularly active in the scientific community (notably in the New Interfaces for Musical Expression Conference - NIME [172]), and has found some commercial traction, such as with ROLI's expressive keyboards featuring the MIDI Polyphonic Expression (MPE) protocol, bridging academic research and mainstream music production [173] (see Figure 8.1).

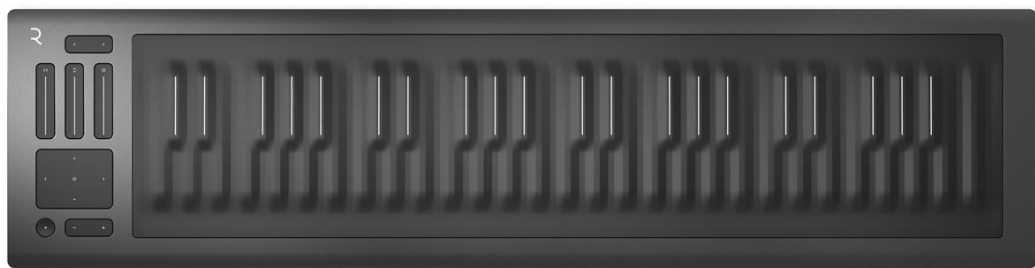


Figure 8.1: The ROLI Seaboard Rise 49 keyboard.

While this approach may be valuable for experienced musicians and experimenters, it offers little utility to producers who lack the technical ability or resources to use such controllers. Additionally, every musical instrument has its own specific techniques, modes of use, and limitations, often mastered only by skilled players. Expressive models, such as the one described here for lead electric guitar, could serve as crucial tools for composers and producers, helping

them to translate their musical ideas into highly realistic performances.

8.2.2 Education

The benefits of digital media and new technologies in enhancing and making more involving and fruitful the musical learning experience have been highlighted by research (see for example, in the specific area of guitar playing, [174]–[176]).

In lead electric guitar learning, automatically generating fingerings that adjust based on user-defined parameters — such as the penalty associated with distance from the handy position — could be highly beneficial. It would allow students to gradually familiarize themselves with the fretboard and progressively increase the biomechanical difficulty of playing exercises and pieces. Additionally, incorporating variably articulations and expressive techniques, depending on user preferences, could support the creation of targeted exercises designed to improve students’ mastery of these techniques.

8.2.3 Automatic music generation

To achieve fully realistic rendering, automatic musical material generation requires expressive models to be applied to the generated symbolic musical representations. While some systems have been proposed that generate both music and its expressive rendition (see, for example, [61], [177]), this is often not the case. The proposed model could serve as an intermediate step between generating guitar melodies and their subsequent sound realization, helping bridge the gap between symbolic notation and expressive performance.

8.2.4 Automatic generation of guitar tablatures from MIDI materials

The guitar notation software available on the market allows MIDI File imports and automatic translation into tablature, indicating the string and fret to use for each note. As discussed in Edward et al. (2024) [178], this adaptation often falls short compared to real musicians’ fingerings. The authors propose a system based on Masked Language Modeling, which generally produces superior tablatures but is limited by its reliance on pre-trained data. Unlike this, the system presented in this thesis could support the generation of alternative fingerings and validate those created by other systems.

8.3 Multidisciplinary and interdisciplinary research and computer engineering

This thesis is an example of multidisciplinary and interdisciplinary research in the tradition of the Centro di Sonologia Computazionale, the research center of the Department of Information Engineering (DEI) of the University of Padua where I did my doctoral studies (see Canazza and De Poli (2019) [110]).

If, on the one hand, disciplinary research concentrates within the boundaries of a single field, allowing researchers to build specialized knowledge and expertise, on the other hand multidisciplinary approaches involve multiple disciplines working in parallel on a problem, with each retaining its own methods and perspectives without fully integrating with others. Interdisciplinary research, in turn, takes it a step further by merging insights, methodologies, and tools from different fields, creating a more unified and holistic approach to solving complex problems [179].

Although computer engineering is at the heart of the work described in the previous chapters, the results obtained could not have existed without significant interaction and integration with other areas of knowledge, particularly musicology and psychology. Since the mid-20th century, there has been a noticeable move away from strictly monodisciplinary approaches in research [180]. This is primarily driven by the understanding that complex challenges necessitate the collaboration of specialists from diverse areas [181], or at least of scholars capable of moving easily between different fields of knowledge, or integrating tools and approaches from different disciplines. This combination of expertise allows for a more comprehensive understanding of problems and fosters the creation of innovative solutions that would not be possible within the confines of a single discipline. The relevance of the interaction between different fields is also highlighted in Vantard et al. (2023) [182], where it is observed that interdisciplinary research has become increasingly prominent in contemporary science, as it is essential for tackling complex problems. There, it is regarded as a crucial component in developing holistic approaches to address many of today's unresolved scientific questions. Nonetheless, the coexistence and integration of different disciplines in the same line of research has historically presented complex challenges in its evaluation. In this regard, in Wagner et al. (2011) [183] attention is drawn to two fundamental points: the relevance of embracing the concept of knowledge integration, and that of acknowledging that such integration can take place both within an individual's mind and collaboratively within a team. Interdisciplinary research requires distinct evaluation criteria that reflect the integration of knowledge across multiple fields, as it cannot be properly assessed by applying the standards of a single discipline [184].

Although the heart of this thesis is undoubtedly linked to computer science, the advances

in scientific knowledge are purely multidisciplinary and interdisciplinary, and cannot be traced back to any of the individual disciplines involved separately from the others. As such, I hope it will be read and will be able to contribute to subsequent research.

Appendix

Chapter 4

Bending

The algorithm for automatic bending input presented here is based on the six-string guitar in standard E2 > E4 tuning, but can easily be adapted to any number of strings and tuning.

Variables description

- **melody**: sequence of notes, each with pitch, duration, fingering
- **scale**: reference scale (the set of usable notes)
- **target_bend_perc[str_idx]**: predefined target percentages for bending distribution on each string, retrieved from the *MySongBook* corpus
- **max_bend[str_idx]**: array containing the maximum bending in semitones playable with each finger of the fretting hand
- **bend_min_dur**: minimum duration of the note to consider bending applicable
- **tot_note_count[str_idx]**: array containing the total number of notes present on each string
- **bend_count[str_idx]**: array containing the total number of notes belonging to high_vib_prob_notes on each string

Pseudo code

Input :

```
melody
scale
target_bend_perc[str_idx]
max_bend[fng_idx]
bend_min_dur
```

```

user_response = AskUser("Do you want to customize the target distribution \
per string?")

if user_response == "Yes" then
    Update(target_bend_perc[str_idx])
end if

tot_note_count[str_idx] = CountNotesPerString(melody)
bend_count[str_idx] = 0
target_bend[str_idx] = tot_note_count[str_idx] * target_bend_perc[str_idx]

# Identify ascending, descending and unison intervals between contiguous \
# notes in the melody

for note in melody do
    if (note + 1).pitch > note.pitch then
        ascending interval
    else if (note + 1).pitch < note.pitch then
        descending interval
    else
        unison interval      # in this case also a "horizon interval" \
                             # (the first non-unison following interval) \
                             # is stored
    end if

# Definition of a function that finds the nearest lower note belonging to \
# the selected scale to that to be reached by the bend

Function: FindLowerNote
Input: scale, note
Output: lower_note

# Definition of functions to check that there are not bends already \
# applied to previous, current/following and previous/current/following notes

Function: PrevBendCheck
Input: previous note
Output: boolean (True if there is no bend applied to the previous note)

```

Function: CurrFlwBendCheck

Input: previous note

Output: boolean

Function: PrevCurrFlwBendCheck

Input: previous note

Output: boolean

Function for the correction of the fingering of the note preceding \
a bend, where needed

Function: FingCorr

Input: previous and current note

Output: new fingering

Begin

```
    if note.string == (note - 1).string and note.fret == (note - 1).fret then  
        (note - 1).finger == note.finger  
    end if
```

Definition of the function responsible to move the note obtained using a bend \
to a lower string

Function: BendLowerString

Input: previous, current and following note

Output: insertion of bend, with a new fingering

Begin

```
    if note.string == second string then # move to 3rd string and bend  
        if bend_ampl <= 2 then # -> use ring_fng to bend  
            if max_bend[ring_finger] >= bend_ampl then # max bend for ring_fng  
                Nested Function : Bend3rdStr3rdFngr  
                    if target_bend[third string] > bend_count[third string] then  
                        note.string = third string  
                        note.fret += (4 - bend_ampl)  
                        note.finger = ring_fng  
                        note.bend = bend_ampl  
                        bend_count[third string] += 1  
                        Call FingCorr
```

```

        end if
    if first note of the melody then
        Call Bend3rdStr3rdFngr
    end if
    else if (note - 1).finger != third finger:
        Call Bend3rdStr3rdFngr
    end if
else if bend_ampl == 3 # move note to third string and bend with middle_fng
    if max_bend[middle_fng] >= bend_ampl then # max bend for middle_fng
        Nested Function : Bend3rdStr2ndFngr
            if target_bend[third string] > bend_count[third string] then
                note.string = third string
                note.fret += (4 - bend_ampl)
                note.finger = middle_fng
                note.bend = bend_ampl
                bend_count[third string] += 1
                Call FingCorr
            end if
        if first note of the melody then
            Call Bend3rdStr2ndFngr
        end if
        else if (note - 1). finger != middle_fng:
            Call Bend3rdStr2ndFngr
        end if
    end if

```

The same logic is applied to all the other strings, keeping into consideration the \
different interval between second and third string (4 semitones) and between each \
other couple of contiguous strings (5 semitones)

Return updated fingering, with bend

End

The algorithms for inserting bending into the typical contexts \
previously identified follow

#####

Function: UnisAscBend # Bend on the first of two unison notes, \

followed by an ascending interval

Input: current and following note

Output: insertion of bend, with a new fingering

Begin

```
    if note.interval == unison and note.horizon_interval == ascending then
        if note.finger == index_fng and note.string != 6 \
        and (note+1).finger == index_fng then
            Call BendLowerString
        end if
    end if
```

End

#####

Function: UnisDescBend # Bend on the second of two unison notes, \
followed by a descending interval

Input: previous, current and following note

Output: insertion of bend, with a new fingering

Begin

```
    if (Call PrevCurrFlwBendCheck) and (note - 1).interval == unison \
    and (note-1).horizon_interval == descending then
        if note.finger == index_fng and note.string != 6 \
        and (note - 1).finger == index_fng then
            Call BendLowerString
            if note.string = (note + 1).string and note.fret = (note + 1).fret then
                (note + 1).finger == note.finger
            end if
        end if
    end if
```

End

#####

Function: ThreeNotesGroup

Input: current and the two following notes

Output: insertion of bend, with a new fingering

Begin

```
bend_ampl = (note + 1).pitch - note.pitch
if (Call CurrFlwBendCheck) and (note + 2).bending == 0 \
and note.string == (note + 2).string \
and note.pitch == (note + 2).pitch and bend_ampl > 0 and \
bend_ampl <= max_bend[note.finger] then
    if target_bend[note.string] > bend_count[note.string] then
        (note + 1).string = note.string
        (note + 1).fret = note.fret
        (note + 1).finger = note.finger
        (note + 1).bending = bend_ampl
        bend_count[note.string] += 1
    end if
end if
```

End

#####

Function: AscBend

Input: current and following note

Output: insertion of bend, with a new fingering

Begin

```
bend_ampl = (note + 1).pitch - note.pitch
if bend_ampl > 0 and (Call PrevCurrFlwBendCheck) \
and note.string == (note + 1).string \
and (note + 2).finger != note.finger and bend_ampl <= max_bend[note.finger] \
and target_bend[note.string] > bend_count[note.string] then
    (note + 1).fret = note.fret
    (note + 1).finger = note.finger
    (note + 1).bending = bend_ampl
    bend_count[note.string - 1] += 1
end if
```

End

#####

Function: DescBend

Input: current and following note

Output: insertion of bend, with a new fingering

Begin

```
bend_ampl = note.pitch - (note + 1).pitch
if bend_ampl > 0 and (Call PrevCurrFlwBendCheck) \
and note.string == (note + 1).string \
and (note - 1).fret != (note + 1).fret \
and bend_ampl <= max_bend[(note + 1).finger] \
and target_bend[note.string] > bend_count[note.string] then
    note.fret = (note + 1).fret
    note.finger = (note + 1).finger
    note.bending = bend_ampl
    bend_count[note.string] += 1
end if
```

End

#####

```
# The functions are not called when they would give errors (for example on the \
# first note of the melody if the previous one is used in the function. \
# The conditions under which functions are called are not formalised \
# in pseudo-code for simplicity, but are based on verifying \
# the absence of incompatibility with the note preceding and following \
# the application of the bending, in terms of fingering and previously \
# applied performance techniques
# (see, for example, the revBendCheck and the other similar functions).
```

```
for note in melody do
```

```
    if note.duration > bend_min_dur then
        bend_ampl = Call FindLowerNote(note.pitch)
        Call UnisAscBend
    end if
```

```
for note in melody do
```

```
    if note.duration > bend_min_dur then
        bend_ampl = Call FindLowerNote(note.pitch)
        Call UnisDescBend
    end if
```

```

for note in melody do
  if note.duration > bend_min_dur then
    Call ThreeNotesGroup
  end if

for note in melody do
  if note.duration > bend_min_dur then
    Call AscBend
  end if

for note in melody do
  if note.duration > bend_min_dur then
    Call DescBend
  end if

return modified_melody

```

Hammer-on/Pull-off

In the model, hammer-on and pull-off are handled by the same section of code.

Variables description

- **melody**: sequence of notes, each with pitch, duration, fingering
- **target_hmm_on_perc**: predefined target percentages for hammer-on distribution on each string, retrieved from the *MySongBook* corpus
- **target_pll_off_perc**: predefined target percentages for pull-off distribution on each string, retrieved from the *MySongBook* corpus
- **hmm_on_opt[str_idx]**: array containing the total number of contexts in which hammer-on is possible, for each string
- **pll_off_opt[str_idx]**: array containing the total number of contexts in which pull-off is possible, for each string
- **hmm_on_opt_perc[str_idx]**: array containing the percentage of notes to which it is possible to apply a hammer-on for each string, with respect to the total number
- **pll_off_opt_perc[str_idx]**: array containing the percentage of notes to which it is possible to apply a pull-off for each string, with respect to the total number

- **hmm_on_target[str_idx]**: array containing the target number of hammer-on applications on each string
- **pll_off_target[str_idx]**: array containing the target number of pull-off applications on each string
- **hmm_on_count[str_idx]**: array containing the total number of hammer-on applied on each string
- **pll_off_count[str_idx]**: array containing the total number of pull-off applied on each string

Pseudo code

Input:

```
melody
target_hmm_on_perc[str_idx]
target_pll_off_perc[str_idx]
```

```
user_response = AskUser("Do you want to customize the target distribution \
per string?")
```

```
if user_response == "Yes" then
    Update(target_hmm_on_perc[str_idx])
    Update(target_pll_off_perc[str_idx])
end if
```

```
#####
```

Function: Hammer_on_Pull_off_feasibility_check

Input: current and following note

Output: identification of notes on which hammer-ons or pull-offs are possible

Begin

```
if note.string == (note + 1).string and note.pitch != (note + 1).pitch and \
[note, note + 1].bending == False then
    if note.pitch > (note + 1).pitch then
        (note + 1).pll_off_possible = True
        pll_off_opt[str_idx] += 1
    end if
else
```

```

        (note + 1).hmm_on_possible = True
        hmm_on_opt[str_idx] += 1
    end if
end if
End

#####

for note in melody do
    Call Hammer_on_Pull_off_feasibility_check
end for

tot_note_count[str_idx] = CountNotesPerString(melody)
hmm_on_target[str_idx] = tot_note_count[str_idx] * target_hmm_on_perc[str_idx] / 100
pll_off_target[str_idx] = tot_note_count[str_idx] * target_pll_off_perc[str_idx] / 100
hmm_on_opt_perc[str_idx] = hmm_on_opt[str_idx] / tot_note_count[str_idx] * 100
pll_off_opt_perc[str_idx] = pll_off_opt[str_idx] / tot_note_count[str_idx] * 100

#####

Function: Add_Hammer_on_Pull_off
Input: current following note
Output: insertion of hammer-on or pull-off

Begin
    if note.pll_off_possible == True then
        if pll_off_opt[str_idx] <= pll_off_target[str_idx] then
            ApplyPullOff(note)
        else
            probability = pll_off_target_perc[str_idx] / pll_off_opt_perc[str_idx] * 100
            if random number(0,100) < probability:
                ApplyPullOff(note)
                pll_off_count[str_idx] += 1
            end if
        end if
    end if
# It follows the application of hammer-on to compatible notes, \
# in a similar way as seen in the previous step

end if

```

End

#####

```
for note in melody do
    Call Add_Hammer_on_Pull_off
end for

return modified_melody
```

Vibrato

Variables description

- **melody**: sequence of notes, each with pitch, duration, fingering
- **target_vib_perc[str_idx]**: predefined target percentages for vibrato distribution on each string, retrieved from the *MySongBook* corpus
- **t1**: threshold duration 1, above this duration application of the vibrato is possible; can be set by the user
- **t2**: threshold duration 2, above this duration application of the vibrato is considered highly probable; can be set by the user
- **high_vib_prob_notes**: categorization to which fretted notes with $t2 > \text{duration} > t1$ belong
- **vib_comp_notes**: categorization to which fretted notes with $\text{duration} > t2$ belong
- **tot_note_count[str_idx]**: array containing the total number of notes present on each string
- **hvpn_count[str_idx]**: array containing the total number of notes belonging to high_vib_prob_notes on each string
- **vcn_count[str_idx]**: array containing the total number of notes belonging to vib_comp_notes on each string
- **req_vib_count[str_idx]**: array containing the target number of vibratos to be applied on each string
- **vib_count[str_idx]**: array containing the total number of vibratos applied on each string
- **hvpn_perc[str_idx]**: array containing the percentage of notes belonging to high_vib_prob_notes to which vibrato should be applied, to reach the target distribution, for each string

Pseudo code

Input:

```
melody
target_vib_perc[str_idx]
t1
t2
```

```
for each note in melody do
  if note.fretted then
    if note.duration > t2 then
      Add note to high_vib_prob_notes
    else if note.duration > t1 then
      Add note to vib_comp_notes
    end if
  end if
end for
```

```
user_response = AskUser("Do you want to apply vibrato to all compatible \
notes, or customize the target distribution per string?")
```

```
if user_response == "Apply to all compatible notes" then
  for each note in vib_comp_notes do
    ApplyVibrato(note)
  end for
  return modified_melody
else
  tot_note_count[str_idx] = CountNotesPerString(melody)
  for each note in melody do
    if note is in high_vib_prob_notes do
      hvpn_count[str_idx] += 1
    else if note is in vib_comp_notes do
      vcn_count[str_idx] += 1
    end if
  end for

  if user_response == "Customize the target distribution" then
    Update(target_vib_perc[str_idx])
  end if
```

```

    req_vib_count[str_idx] = target_vib_perc[str_idx] * tot_note_count[str_idx]
    hvpn_perc[str_idx] = hvpn_count[str_idx] * 100 / tot_note_count[str_idx]

# Application of vibrato to notes included in high_vib_prob_notes

    for each note in melody do
        if note is in high_vib_prob_notes \
        and hvpn_count[str_idx] < req_vib_count[str_idx] then
            ApplyVibrato(note)
            vib_count[str_idx] += 1
        else if note is in high_vib_prob_notes \
        and hvpn_count[str_idx] > req_vib_count[str_idx] then
            probability = target_vib_perc[str_idx] / hvpn_perc[str_idx] * 100
            if random number(0,100) < probability:
                ApplyVibrato(note)
                vib_count[str_idx] += 1
            end if
        end if
    end for
end if

# It follows the application of vibrato to notes included in vib_comp_notes \
# in a similar way as seen in the previous step, reshaping the target \
# percentages for each string, based on the vibratos already applied

return modified_melody

```

Chapter 5

Exercises used for dataset construction

Detailed fingering was only included where more biomechanically valid solutions would be possible.

Exercise no. 1

Standard tuning

Exercise no. 2

Standard tuning

el.guit.

12-11-10-9 11-10-9-12 10-9-12-11 9-12-11-10 12-11-10-9 11-10-9-12 10-9-12-11 9-12-11-10

3 4

12-11-10-9 11-10-9-12 10-9-12-11 9-12-11-10 12-11-10-9 11-10-9-12 10-9-12-11 9-12-11-10

Exercise no. 3

Standard tuning

el.guit.

12-11-10-8 11-10-8-12 10-8-12-11 8-12-11-10 12-11-10-8 11-10-8-12 10-8-12-11 8-12-11-10

3 4

12-11-10-8 11-10-8-12 10-8-12-11 8-12-11-10 12-11-10-8 11-10-8-12 10-8-12-11 8-12-11-10

Exercise no. 4

Standard tuning

el. guit.

8-5 8-5-8-5 7-5-7-5 7-5-7-5 7-5 7-5 8-5 7-5 8-5 8-5 10-8 10-8-10-8 9-7-9-7 10-7

10-7 10-7 9-7 10-7 10-8 10-8 10-8 12-10 13-10 13-10 12-9 12-9 12-10 12-10 12-10 12-10 12-10

12-9 13-10 12-9 13-10 15-12 15-13 15-13 14-12 14-12 14-12 15-12 15-12 14-12 15-12 14-12 15-12 15-13

17-15 17-15-17-15 17-14-17-14 17-14-17-14 17-15 17-15 17-14 17-14 17-15 17-15 17-15 17-15

Exercise no. 5

Standard tuning

od. guit.

1 2 3 4

5 6 7 8 9 10 11 12

13 14 15 16 17 18 19 20 21 22 23 24

25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

Exercise no. 6

Standard tuning

el.guit.

T	
A	
B	

T
A
B

TAB

T	
A	
B	

Exercise no. 7

Standard tuning

el.guit.

1 5 3 2 5 3 2 5 3 2 5 3 2 5 3 2

5 4 2 5 4 3 5 5 3 6 5 3 6 5 3 7

[illegible]

The image displays a musical score for the song "The Wind" by The Beatles. It includes a guitar part and a bass part. The guitar part is written in 4/4 time, with a key signature of one sharp (F#). The melody is composed of eighth and sixteenth notes, with some triplets. The bass part is written in a simplified notation, using numbers 1-4 to represent frets. The score is divided into two systems, each containing a guitar staff and a bass staff. The guitar staff has a treble clef, and the bass staff has a bass clef. The guitar part includes a capo on the 11th fret, indicated by a bracket and the number 11. The bass part includes a capo on the 12th fret, indicated by a bracket and the number 12. The guitar part includes a key signature change to one sharp (F#) at the beginning of the second system. The bass part includes a key signature change to one sharp (F#) at the beginning of the second system. The guitar part includes a key signature change to one sharp (F#) at the beginning of the second system. The bass part includes a key signature change to one sharp (F#) at the beginning of the second system.

The image displays a musical score for the song "The Wind" by The Beatles. It includes a guitar part and a bass part. The guitar part is written in 2/4 time, starting with a treble clef and a key signature of one sharp (F#). The bass part is written in 4/4 time, starting with a bass clef and a key signature of one sharp (F#). The guitar part features a melodic line with various intervals and a final chord. The bass part provides a rhythmic accompaniment with a steady eighth-note pattern. The score is presented in a clean, black-and-white format with standard musical notation and tablature.

Guitar Tab:

15

16

1 4 2 1 4 2 1 4 2 1 4 3 1 4 3 1

7 10 8 7 10 8 7 10 8 7 10 9 7 10 9 7

Bass Tab:

(5)

7 10 8 7 10 8 7 10 8 7 10 9 7 10 9 7

The second system of the musical score for 'The Wind' (Part 2) is shown. It continues from the first system, starting at measure 17. The notation includes a treble clef, a key signature of one flat (B-flat), and a 4/4 time signature. The melody is written on a single staff, and the guitar tablature is written below it. The tablature uses numbers 1-4 to indicate frets and 'V' for vibrato. The system ends at measure 18.

19 3 4 1 3 4 1 2 4 1 2 4 1 2 4 1

7 9 10 7 9 10 7 8 10 7 8 10 7 8 10 7

20 2

(7)

25 26

T
A
B

(8) 10 13 12 10 13 12 10 14 12 14 12 10 14 12 10

27 28

T
A
B

12-10-14-12 12 14 13-12-15-13 12 15 13-12-15 15-12-13 12 10 12-13 14 12-14-10-12 14

29 30

T
A
B

10 12 14 10 12 14 10 12 14 10 12 13 10 12 13 10 (10)

31 32

T
A
B

12-15-13 12 15 14 12-15-14 12 15 14-12-15-14 12 15 14-12-16-14 13 16 15-13-17-15 13 17 15-13-17

33 34

T
A
B

17-13-15 13 17 15-17-13-15 13 12 16 14 12-14 15 12 14-15-12-14 12 15 14-15-12-14 12 15 13-15-12

Exercise no. 8

Standard tuning

el guit.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000

Exercise no. 9

Standard tuning

[illegible]

Exercise no. 10

Standard tuning

[illegible]

25 26 27 28

29 30 31 32

33 34 35 36

37 38 39 40

TAB B

Chapter 5

Performance of predictive models in relation to `prec_rest_ms` and velocity variables

`prec_rest_ms`

Figures from 8.2 to 8.6.

velocity

Figures from 8.7 to 8.11.

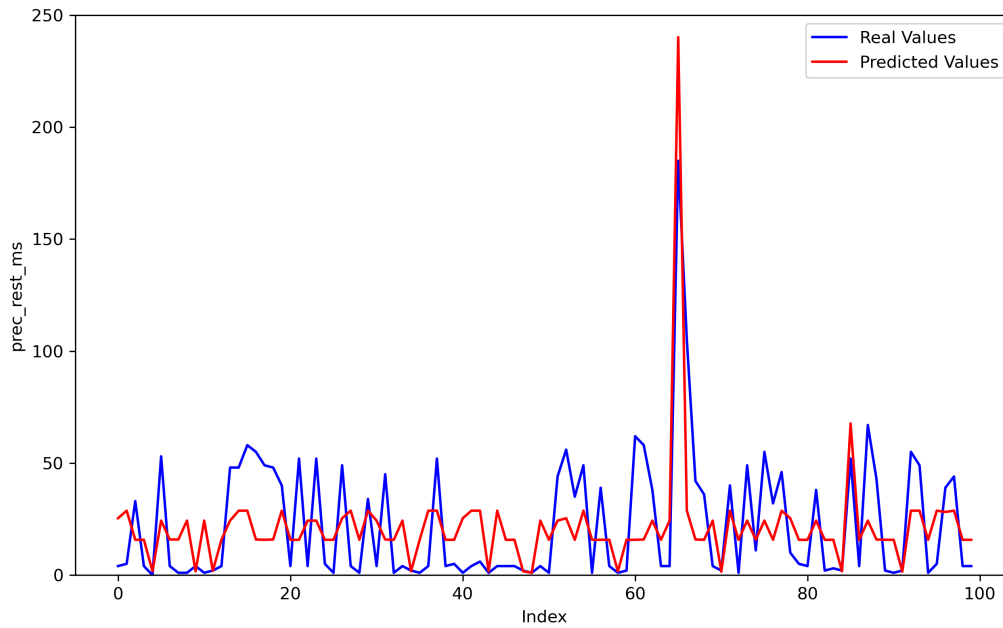


Figure 8.2: Comparison of real and predicted `prec_rest_ms` values, based on the decision tree regressor. Test Mean Squared Error: 4834.01, Test R-squared: -0.43, Train Mean Squared Error: 660.70, Train R-squared: 0.86.

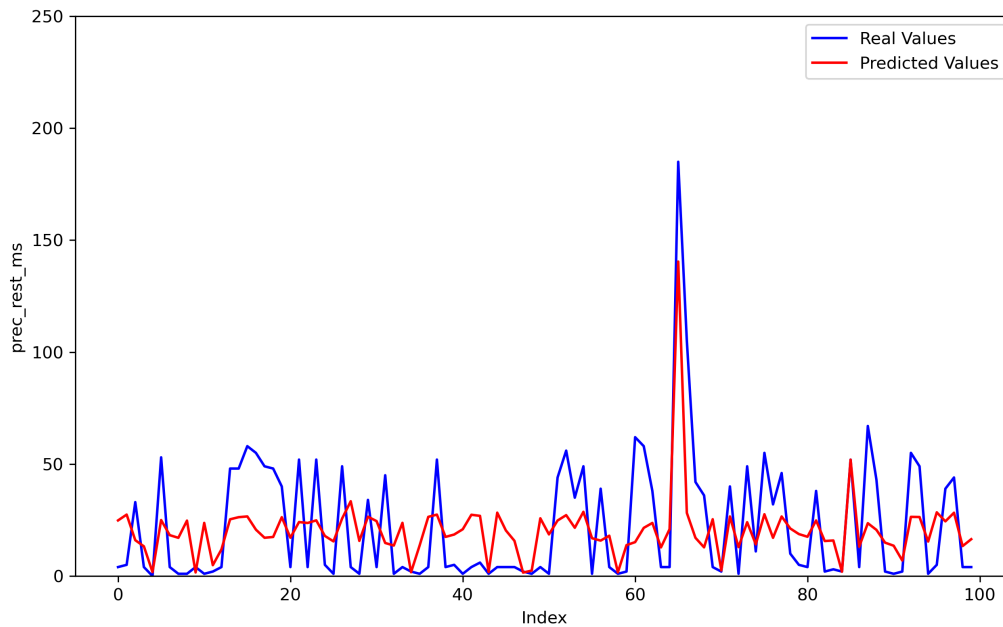


Figure 8.3: Comparison of real and predicted `prec_rest_ms` values, based on the random forest regressor. Test Mean Squared Error: 3053.26, Test R-squared: 0.09, Train Mean Squared Error: 789.15, Train R-squared: 0.84.

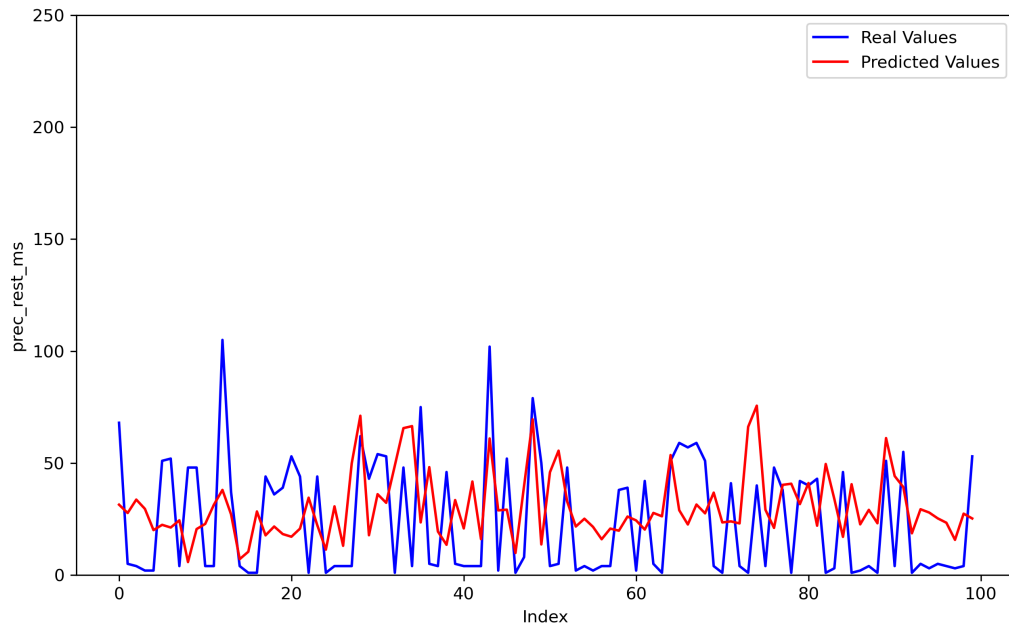


Figure 8.4: Comparison of real and predicted `prec_rest_ms` values, based on the RNN with one time step. Test Mean Squared Error: 4496.36, Test R-squared: 0.05, Train Mean Squared Error: 4495.58, Train R-squared: 0.05.

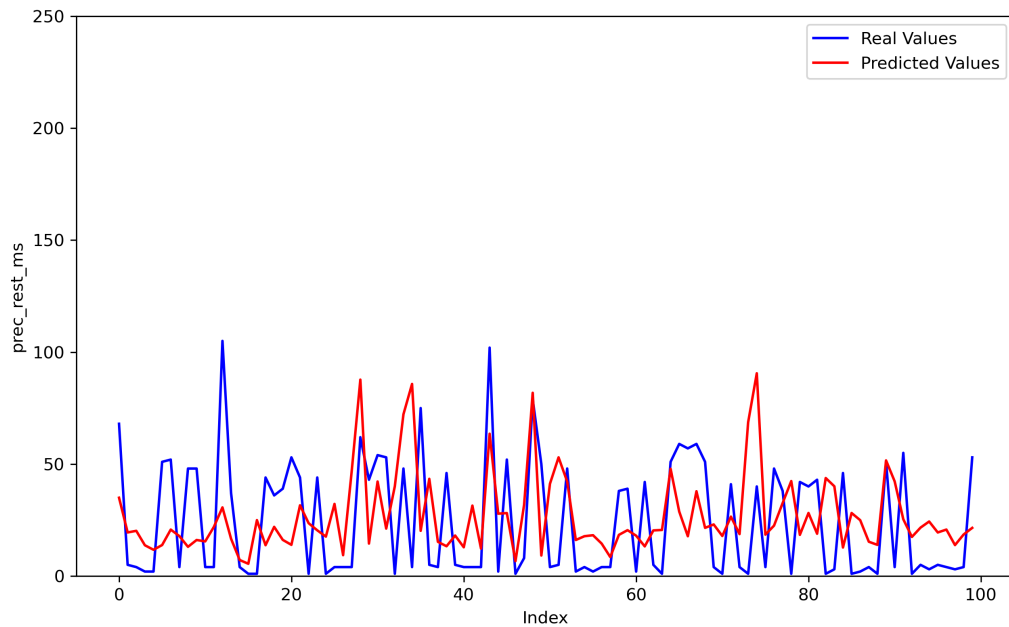


Figure 8.5: Comparison of real and predicted `prec_rest_ms` values, based on the LSTM with one time step. Test Mean Squared Error: 4428.30, Test R-squared: 0.06, Train Mean Squared Error: 4358.72, Train R-squared: 0.08.

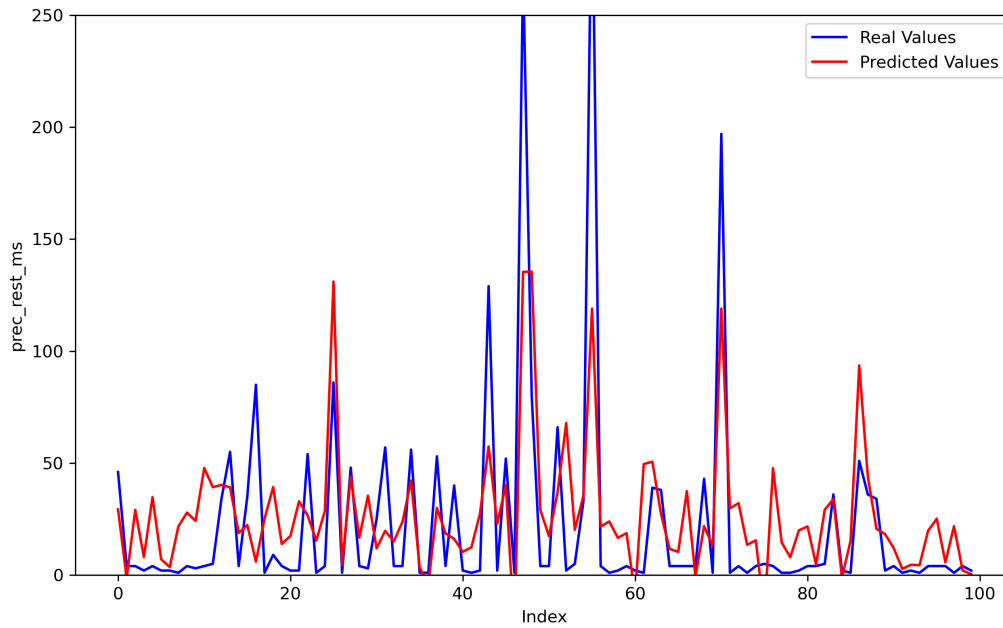


Figure 8.6: Comparison of real and predicted `prec_rest_ms` values, based on the LSTM with four time steps. Test Mean Squared Error: 3819.06, Test R-squared: 0.27, Train Mean Squared Error: 2760.63, Train R-squared: 0.42.

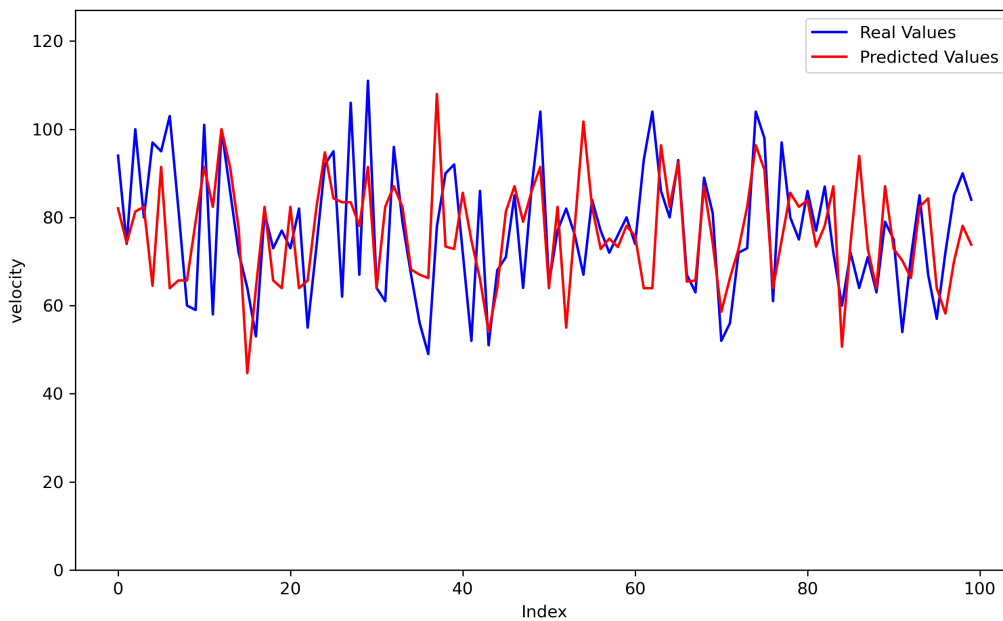


Figure 8.7: Comparison of real and predicted velocity values, based on the decision tree regressor. Test Mean Squared Error: 191.91, Test R-squared: 0.27 Train Mean Squared Error: 122.97, Train R-squared: 0.53.

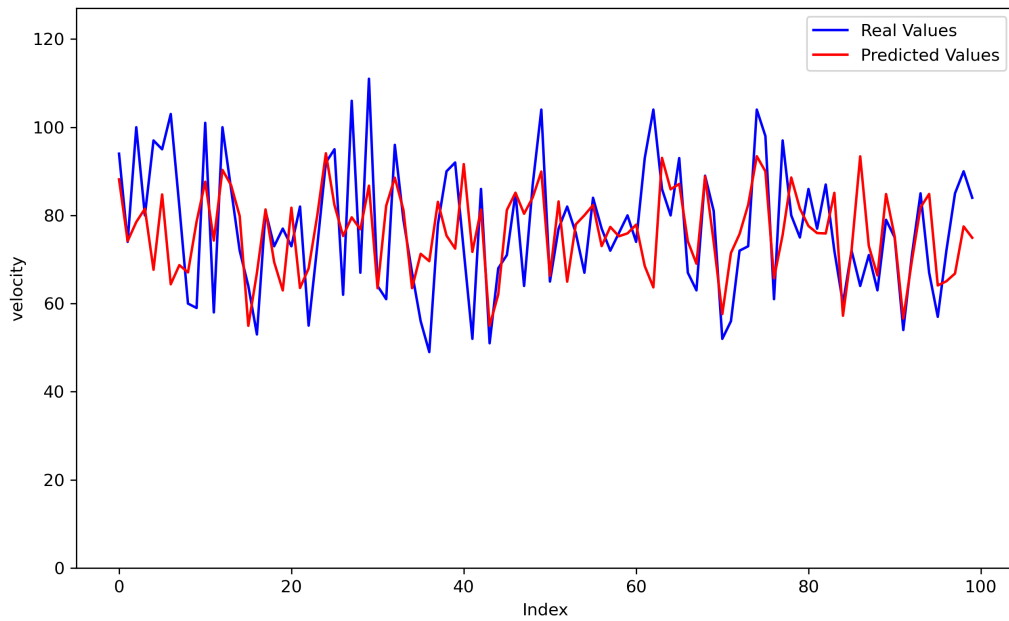


Figure 8.8: Comparison of real and predicted velocity values, based on the random forest regressor. Test Mean Squared Error: 168.96, Test R-squared: 0.36, Train Mean Squared Error: 107.13, Train R-squared: 0.58.

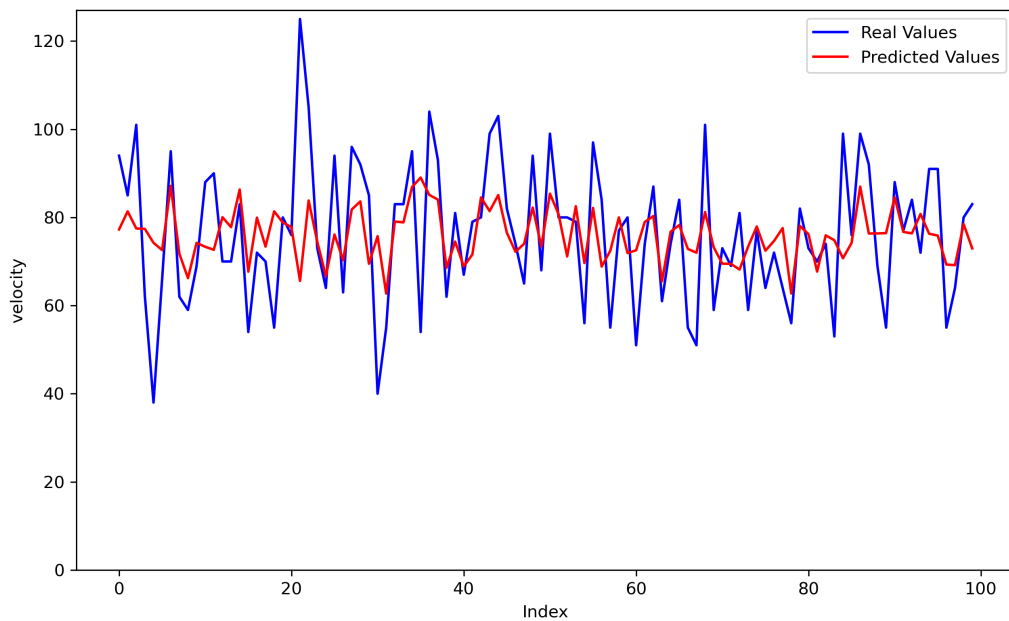


Figure 8.9: Comparison of real and predicted velocity values, based on the RNN with one time step. Test Mean Squared Error: 230.88, Test R-squared: 0.09, Train Mean Squared Error: 226.13, Train R-squared: 0.12.

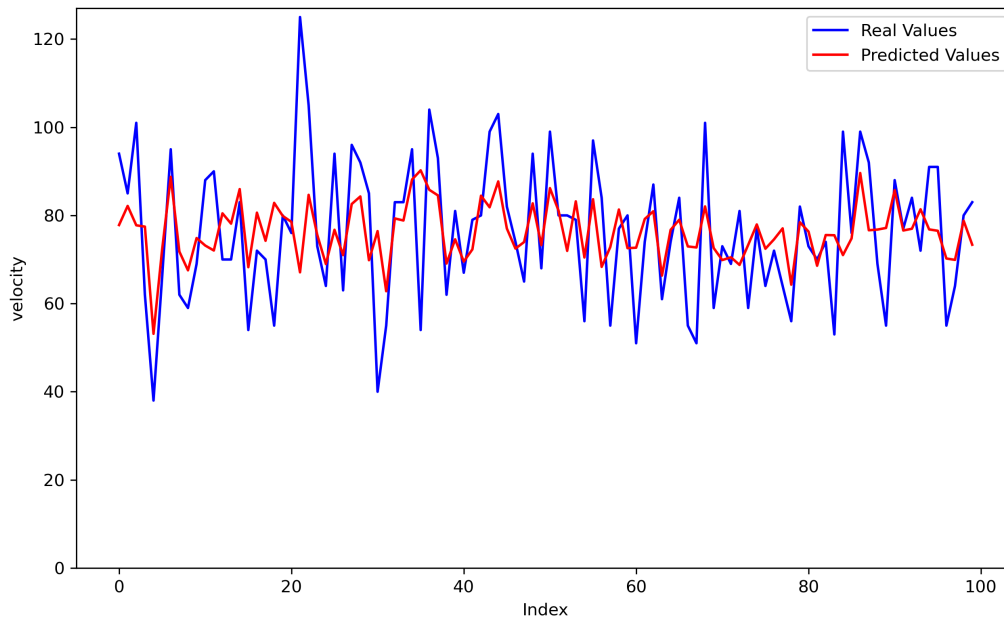


Figure 8.10: Comparison of real and predicted velocity values, based on the LSTM with one time step. Test Mean Squared Error: 223.99, Test R-squared: 0.12, Train Mean Squared Error: 216.63, Train R-squared: 0.15.

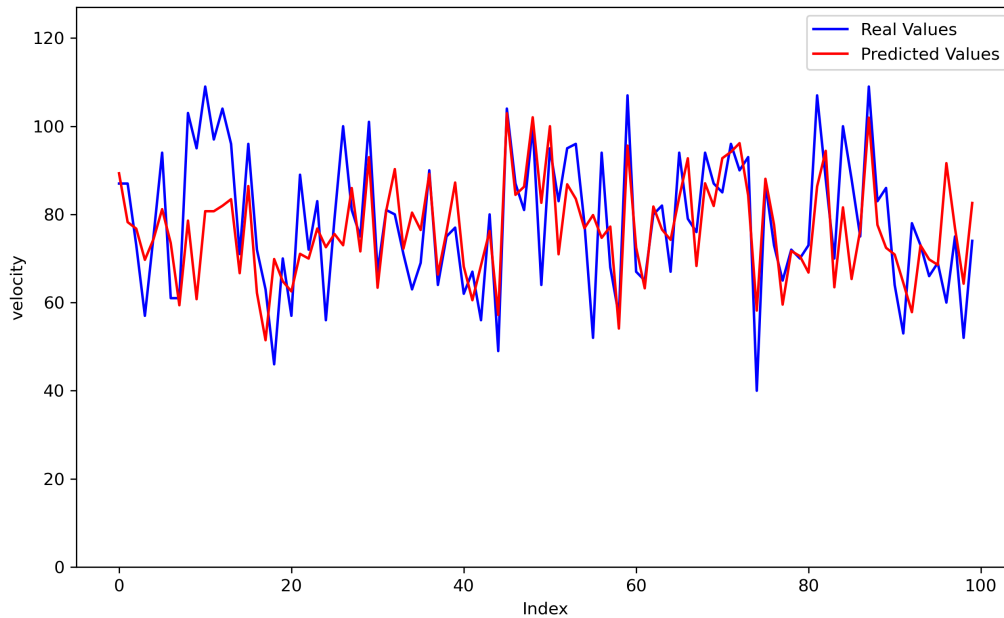


Figure 8.11: Comparison of real and predicted velocity values, based on the LSTM with four time steps. Test Mean Squared Error: 111.61, Test R-squared: 0.54, Train Mean Squared Error: 121.68, Train R-squared: 0.53.

Summary tables of predictive models results

delta_onset_ms

	DT		RF		RNN-1		RNN-4		LSTM-1		LSTM-4	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
RMSE (ms)	12,93	15,44	12,28	14,35	16,03	16,81	14,35	15,35	15,77	16,66	14,58	15,34
R-squared	0,70	0,56	0,73	0,62	0,53	0,48	0,62	0,57	0,54	0,49	0,60	0,57

Table 8.1: Performance evaluation of predictive models based on Root Mean Squared Error (RMSE) and R-Squared values, delta_onset_ms feature.

prec_rest_ms

	DT		RF		RNN-1		RNN-4		LSTM-1		LSTM-4	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
RMSE (ms)	25,70	69,53	28,10	55,26	67,05	67,05	48,08	56,39	66,02	66,55	52,54	61,80
R-squared	0,86	-0,43	0,84	0,09	0,05	0,05	0,52	0,39	0,08	0,06	0,42	0,27

Table 8.2: Performance evaluation of predictive models based on Root Mean Squared Error (RMSE) and R-Squared values, prec_rest_ms feature.

velocity

	DT		RF		RNN-1		RNN-4		LSTM-1		LSTM-4	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
RMSE	11,10	13,85	10,35	13,00	15,04	15,19	10,62	10,48	14,72	14,97	11,03	10,56
R-squared	0,53	0,27	0,58	0,36	0,12	0,09	0,56	0,54	0,15	0,12	0,53	0,54

Table 8.3: Performance evaluation of predictive models based on Root Mean Squared Error (RMSE) and R-Squared values, velocity feature.

Scientific publications

Published journal papers

P. Bontempi, S. Canazza, F. Carnovalini, *et al.*, “Research in computational expressive music performance and popular music production: A potential field of application?” *Multimodal Technologies and Interaction*, vol. 7, no. 2, 2023, ISSN: 2414-4088. DOI: 10.3390/mti7020015

A. Zuccante, A. Fiordelmondo, P. Bontempi, *et al.*, “A signal-processing-based simulation system for high-end stereo headsets,” *Sensors*, vol. 24, no. 7, p. 2190, 2024. DOI: 10.3390/s24072190

P. Bontempi, P. Zappalà, and S. Canazza, “Expressiveness in metronomic rolls for player piano: A new field of research?” *Journal of Cultural Heritage*, vol. 67, pp. 414–420, 2024, ISSN: 1296-2074. DOI: 10.1016/j.culher.2024.04.004

Conference proceedings

P. Bontempi, D. Manerba, A. D’Hooge, *et al.*, “From MIDI to rich tablatures: An automatic generative system incorporating lead guitarists’ fingering and stylistic choices,” in *Proceedings of the Sound and Music Computing Conference (SMC)*, 2024. DOI: 10.48550/arXiv.2407.09052

Journal papers under review

P. Bontempi and S. Canazza, “Expressiveness in popular music performance and production: A reasoned review of interviews with leading engineers, producers and performers,” *Journal of the International Association for the Study of Popular Music (IASPM)* - Under review, publication date to be determined following completion of reviews

The paper has passed the first revision stage. Both reviewers assessed the work as worthy of publication, with modifications.

Journal papers under preparation

A dataset based on the one presented in Section 5.2, but collecting the contributions of a large number of musicians, is currently being realised. It will be able to count on a mass of data considerably more significant and potentially able to allow comparative studies between different performers. The Centre for Computational Sonology of the University of Padua, the Department of Cultural Heritage of the University of Padua, and scholars from abroad are involved in the realization of the project.

A new version of the system for automatically generating guitar fingerings described in Chapter 3 is under development. This will be based on a multi-objective optimization model, and the inclusion of performance techniques will be managed simultaneously with the fingering generation. The system will have two distinct implementations: one based on the CPLEX solver (called in Python without using the IBM ILOG CPLEX Optimization Studio software) and one redefined in Python as a Shortest Path Problem. This will allow us to evaluate possible performance improvements and facilitate the model's inclusion in end-to-end systems. In addition to me, Sergio Canazza, Daniele Manerba, and Giada Zuccolo are working on the project.

Activities in support of research

During my doctorate I acted as a scientific reviewer for IEEE Transactions on Artificial Intelligence, JCIE-Journal of the Chinese Institute of Engineers, Qeios, DAFx-International Conference on Digital Audio Effects.

Activities for the Third Mission

Logistics and scientific organisation of the CSC stand at the Science4All 2023 event (Saturday 30 September 2023, Palazzo del Bo, Padua)

Science4All is a major popular event organised annually by the University of Padua, in which researchers offer experiments, games, quizzes, workshops, guided tours and scientific demonstrations open to the public.

At the 2023 edition, the Centro di Sonologia Computazionale (CSC) of the Department of Information Engineering (DEI) exhibited to the public the original computer systems that made possible the creation of some of the most important computer music works of the last 50 years, including *Prometeo. La tragedia dell'ascolto* (by Luigi Nono, first performed in 1985) and *Perseo e Andromeda* (by Salvatore Sciarrino, first performed in 1990). The computers were used, at the time, for the synthesis, processing, and spatialisation of sounds. Among the exhibits were old calculators such as Digital's PDP11 and the Apple II, perfectly restored. The public was able to see the original software developed at the CSC and listen to the works, in a restored version. A computer model for the conservation and reactivation of contemporary art heritage (interactive installations) was also presented.

The estimated visits were about 2,000 people.

Presentation at the CSC stand of research on rolls for player piano at the Science4All 2024 event (Saturday 28 and Sunday 29 September 2023, Palazzo del Bo, Padua)

I presented the research work on the intentional insertion of expressive passages into metronomic paper rolls for player piano described in Chapter 2. Historically valuable rolls, their conversion



Figure 8.12: Science4All 2023 edition, the CSC stand.



Figure 8.13: Science4All 2023 edition, the CSC stand.

to MIDI Files, and techniques for automatically searching for expressive passages were presented. Finally, the audience could listen to the performance of MIDI Files generated from the rolls using a MIDI-controlled Yamaha Disklavier vertical piano.

The estimated visits were about 2,000 people.

Reinterpretation with modern technological instruments of unpublished pieces by Gian Francesco Malipiero preserved on a piano roll, and record release

In collaboration with the Conservatory of Padua, I worked on the expressive sonorization of the philological transcription into music score of three pieces by the Italian composer Gian Francesco Malipiero, which have only come down to us in the form of paper roll for player piano. The work involved an evaluation and comparison between the possible interventions in favor of the expressive rendering of the material present on the roll typical of its execution on the player piano (or its modern virtualization) and of modern MIDI manipulation techniques. The result was recorded using a Yamaha Disklavier vertical piano, and it will become part of a record that is scheduled for release in early 2025.

Thesis correlations

F. Frontini, *Enhancement of perforated rolls for player piano: Design and development of software for converting scans into MIDI format*, Bachelor's degree thesis, Available at <https://hdl.handle.net/20.500.12608/53326>, 2023

C. Costantini, *Design of the sound-based nonverbal human-computer interaction for automatic expressive music performance systems*, Bachelor's degree thesis, Available at <https://hdl.handle.net/20.500.12608/57082>, 2023

Popular publications

Books

Original publications

P. Bontempi, *Produzione audio e musicale con Cubase. Percorso teorico e pratico dalle basi ai concetti avanzati*. Milano: LSWR, 2024

Supervisions of the Italian translation

M. Senior, *Recording secrets for the small studio. I segreti della registrazione per gli home studio e i piccoli studi*. Milano: Volontè & Co., 2022

Italian translations

B. Owsinski, *The Mixing Engineer. Manuale completo*. Milano: Volontè & Co., 2024

Bibliography

- [1] G. Widmer and W. Goebel, “Computational models of expressive music performance: The state of the art,” *Journal of New Music Research*, vol. 33, no. 3, pp. 203–216, 2004. DOI: 10.1080/0929821042000317804.
- [2] C. E. Cancino-Chacón, M. Grachten, W. Goebel, and G. Widmer, “Computational models of expressive music performance: A comprehensive and critical review,” *Frontiers in Digital Humanities*, vol. 5, 2018, ISSN: 2297-2668. DOI: 10.3389/fdigh.2018.00025.
- [3] P. Bontempi, S. Canazza, F. Carnovalini, and A. Rodà, “Research in computational expressive music performance and popular music production: A potential field of application?” *Multimodal Technologies and Interaction*, vol. 7, no. 2, 2023, ISSN: 2414-4088. DOI: 10.3390/mti7020015.
- [4] R. W. Picard *et al.*, *Affective computing*, 1997.
- [5] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, “A proposal for the Dartmouth summer research project on artificial intelligence, august 31, 1955,” *AI Magazine*, vol. 27, no. 4, p. 12, 2006. DOI: 10.1609/aimag.v27i4.1904.
- [6] D. Monett and C. W. P. Lewis, “Getting clarity by defining artificial intelligence—a survey,” in *Philosophy and Theory of Artificial Intelligence 2017*, V. C. Müller, Ed., Cham: Springer International Publishing, 2018, pp. 212–214, ISBN: 978-3-319-96448-5. DOI: 10.1007/978-3-319-96448-5_21.
- [7] P. Wang, “On defining artificial intelligence,” *Journal of Artificial General Intelligence*, vol. 10, no. 2, pp. 1–37, 2019. DOI: 10.2478/jagi-2019-0002.
- [8] L. L. Costalonga, M. S. Pimenta, and E. R. Miranda, “Understanding biomechanical constraints for modelling expressive performance: A guitar case study,” *Journal of New Music Research*, vol. 48, no. 4, pp. 331–351, 2019. DOI: 10.1080/09298215.2019.1643892.

- [9] J. Murphy, J. McVay, P. Mathews, D. A. Carnegie, and A. Kapur, “Expressive robotic guitars: Developments in musical robotics for chordophones,” *Computer Music journal*, vol. 39, no. 1, pp. 59–73, 2015. DOI: 10.1162/COMJ_a_00285.
- [10] P. N. Juslin, A. Friberg, and R. Bresin, “Toward a computational model of expression in music performance: The GERM model,” *Musicae Scientiae*, vol. 5, no. 1_suppl, pp. 63–122, 2001. DOI: 10.1177/10298649020050S104.
- [11] C. E. Seashore, *Psychology Of Music*. Milano: McGraw Hill Book Company Inc., 1938.
- [12] L. B. Meyer, *Emotion and meaning in music*. University of Chicago Press, 1956.
- [13] B. H. Repp, “The Aesthetic Quality of a Quantitatively Average Music Performance: Two Preliminary Experiments,” *Music Perception*, vol. 14, no. 4, pp. 419–444, Jul. 1997, ISSN: 0730-7829. DOI: 10.2307/40285732.
- [14] R. I. Godøy and M. Leman, *Musical gestures: Sound, movement, and meaning*. Routledge, 2010.
- [15] P. Schmeling, *Berklee Music Theory Book 2*. Berklee Press, 2011.
- [16] D. Furini, “From recording performances to performing recordings. recording technology and shifting ideologies of authorship in popular music,” *Trans. Revista Transcultural de Música*, no. 14, pp. 1–8, 2010.
- [17] Z. Moir and H. Medbøe, “Reframing popular music composition as performance-centred practice,” *Journal of Music, Technology & Education*, vol. 8, no. 2, pp. 147–161, 2015. DOI: 10.1386/jmte.8.2.147_1.
- [18] R. Moy, *Authorship roles in Popular Music: Issues and debates*. Routledge, 2015. DOI: 10.4324/9781315770550.
- [19] V. Caporaletti *et al.*, *Introduzione alla teoria delle musiche audiotattili. Un paradigma per il mondo contemporaneo*. Ariccia, Rome: Aracne, 2019.
- [20] A. Millard, “The electric guitar: A history of an American icon,” in JHU Press, 2004, ch. Inventing the Electric Guitar, pp. 41–62.
- [21] M. Maas, “Kithara,” in *Grove Music Online*, Oxford University Press, 2001, ISBN: 9781561592630. DOI: 10.1093/gmo/9781561592630.article.15077.
- [22] C. Poché, “ūd,” in *Grove Music Online*, Oxford University Press, 2001, ISBN: 9781561592630. DOI: 10.1093/gmo/9781561592630.article.28694.
- [23] D. Poulton, “Vihuela,” in *Grove Music Online*, Oxford University Press, 2001, ISBN: 9781561592630. DOI: 10.1093/gmo/9781561592630.article.29360.

- [24] T. F. Heck, H. Turnbull, P. Sparks, *et al.*, “Guitar,” in *Grove Music Online*, Oxford University Press, 2001, ISBN: 9781561592630. DOI: 10.1093/gmo/9781561592630.article.43006.
- [25] T. Gerken, D. Hunter, M. Jasson, R. Johnston, J. Morrish, and M. Simmons, *Electric guitars: the illustrated encyclopedia*. Chartwell Books, 2018.
- [26] D. Weinstein, “Rock’s guitar gods — avatars of the sixties,” *Archiv für Musikwissenschaft*, pp. 139–154, 2013.
- [27] P. Bontempi, D. Manerba, A. D’Hooge, and S. Canazza, “From MIDI to rich tablatures: An automatic generative system incorporating lead guitarists’ fingering and stylistic choices,” in *Proceedings of the Sound and Music Computing Conference (SMC)*, 2024. DOI: 10.48550/arXiv.2407.09052.
- [28] M. Murari, A. Roda, O. Da Pos, E. Schubert, S. Canazza Targon, G. De Poli, *et al.*, “Mozart is still blue: A comparison of sensory and verbal scales to describe qualities in music,” in *Proceedings of the Sound and Music Computing Conference (SMC)*, Maynooth University, 2015, pp. 351–357.
- [29] M. Murari, A. Rodà, S. Canazza, G. De Poli, and O. Da Pos, “Is Vivaldi smooth and take? Non-verbal sensory scales for describing music qualities,” *Journal of New Music Research*, vol. 44, no. 4, pp. 359–372, 2015. DOI: 10.1080/09298215.2015.1101475.
- [30] I. D. Bent, *Analysis*, Accessed: 2024-07-05, 2001. DOI: 10.1093/gmo/9781561592630.article.41862.
- [31] W. Drabkin, “Motif [motive],” in *Grove Music Online*, Accessed: 2022-06-05, 2001. DOI: 10.1093/gmo/9781561592630.article.19221.
- [32] “Phrase,” in *Grove Music Online*, Accessed: 2022-06-05, 2001. DOI: 10.1093/gmo/9781561592630.article.21599.
- [33] L. G. Ratner, “Period,” in *Grove Music Online*, Accessed: 2022-06-05, 2001. DOI: 10.1093/gmo/9781561592630.article.21337.
- [34] C. Palmer, “Music performance,” *Annual Review of Psychology*, vol. 48, no. 1, pp. 115–138, 1997. DOI: 10.1146/annurev.psych.48.1.115.
- [35] E. F. Clarke, “Generative principles in music performance,” in *Generative Processes in Music: The Psychology of Performance, Improvisation, and Composition*, J. Sloboda, Ed., Clarendon Press/Oxford University Press, 2001, pp. 1–26. DOI: 10.1093/acprof:oso/9780198508465.003.0001.
- [36] F. Lerdahl and R. Jackendoff, *A generative theory of tonal music*. The MIT Press, 1983.

- [37] N. Todd, “A model of expressive timing in tonal music,” *Music Perception: an Interdisciplinary journal*, vol. 3, no. 1, pp. 33–57, 1985, ISSN: 07307829, 15338312. DOI: 10.2307/40285321.
- [38] N. Todd, “A computational model of rubato,” *Contemporary Music Review*, vol. 3, no. 1, pp. 69–88, 1989. DOI: 10.1080/07494468900640061.
- [39] N. P. McAngus Todd, “The dynamics of dynamics: A model of musical expression,” *The Journal of the Acoustical Society of America*, vol. 91, no. 6, pp. 3540–3550, 1992. DOI: 10.1121/1.402843.
- [40] W. L. Windsor and E. F. Clarke, “Expressive timing and dynamics in real and artificial musical performances: Using an algorithm as an analytical tool,” *Music Perception*, vol. 15, no. 2, pp. 127–152, 1997. DOI: 10.2307/40285746.
- [41] E. F. Clarke and W. L. Windsor, “Real and simulated expression: A listening study,” *Music Perception*, vol. 17, no. 3, pp. 277–313, 2000. DOI: 10.2307/40285819.
- [42] G. Grindlay and D. Helmbold, “Modeling, analyzing, and synthesizing expressive piano performance with graphical models,” *Machine learning*, vol. 65, no. 2, pp. 361–387, 2006. DOI: 10.1007/s10994-006-8751-3.
- [43] G. Widmer and A. Tobudic, “Playing Mozart by analogy: Learning multi-level timing and dynamics strategies,” *Journal of New Music Research*, vol. 32, no. 3, pp. 259–268, 2003. DOI: 10.1076/jnmr.32.3.259.16860.
- [44] G. Widmer, “Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries,” *Artificial Intelligence*, vol. 146, no. 2, pp. 129–148, 2003. DOI: 10.1016/S0004-3702(03)00016-X.
- [45] E. Narmour, *The analysis and cognition of basic melodic structures: the implication-realization model*. University of Chicago Press, 1990.
- [46] C. Cancino-Chacón and M. Grachten, “A computational study of the role of tonal tension in expressive piano performance,” *arXiv preprint*, 2018. DOI: 10.48550/arXiv.1807.01080.
- [47] D. Herremans and E. Chew, “Tension ribbons: Quantifying and visualising tonal tension,” in *Proceedings of the International Conference on Technologies for Music Notation and Representation (TENOR)*, R. Hoadley, C. Nash, and D. Fober, Eds., Cambridge, UK: Anglia Ruskin University, 2016, pp. 8–18, ISBN: 978-0-9931461-1-4.

- [48] T. Mizutani and S. Sasaki, "Style transfer of musical performance expression using note classification based on the implication-realization model," in *Proceedings of the International Conference on Computer Science and Application Engineering*, 2023, pp. 1–7. DOI: 10.1145/3627915.3628093.
- [49] A. Friberg, R. Bresin, and J. Sundberg, "Overview of the kth rule system for musical performance," *Advances in cognitive psychology*, vol. 2, no. 2, pp. 145–161, 2006. DOI: 10.2478/v10053-008-0052-x.
- [50] A. Gabrielsson, "Interplay between analysis and synthesis in studies of music performance and music experience," *Music Perception*, vol. 3, no. 1, pp. 59–86, 1985. DOI: 10.2307/40285322.
- [51] P. Bontempi, S. Canazza, F. Carnovalini, and A. Rodà, "Research in computational expressive music performance and popular music production: A potential field of application?" *Multimodal Technologies and Interaction*, vol. 7, no. 2, 2023, ISSN: 2414-4088. DOI: 10.3390/mti7020015.
- [52] A. Gabrielsson, "Performance of rhythm patterns," *Scandinavian Journal of Psychology*, vol. 15, no. 1, pp. 63–72, 1974. DOI: 10.1111/j.1467-9450.1974.tb00557.x.
- [53] A. Gabrielsson, "Performance and training of musical rhythm," *Psychology of Music - Special Issue*, pp. 42–46, 1982.
- [54] A. Gabrielsson, I. Bengtsson, and B. Gabrielsson, "Performance of musical rhythm in 3/4 and 6/8 meter," *Scandinavian Journal of Psychology*, vol. 24, no. 1, pp. 193–213, 1983. DOI: 10.1111/j.1467-9450.1983.tb00491.x.
- [55] F. Carnovalini, "Computational musical creativity inspired by musicological and statistical analysis," PhD thesis, University of Padua, 2022. [Online]. Available: <https://hdl.handle.net/11577/3454660>.
- [56] K. Teramura, H. Okuma, Y. Taniguchi, S. Makimoto, and S.-i. Maeda, "Gaussian process regression for rendering music performance," *Proceedings of International Conference on Music Perception and Cognition (ICMPC)*, pp. 167–172, 2008.
- [57] S. Flossmann, M. Grachten, and G. Widmer, "Expressive performance rendering with probabilistic models," in *Guide to Computing for Expressive Music Performance*, Springer, 2013, pp. 75–98. DOI: 10.1007/978-1-4471-4123-5_3.

- [58] K. Okumura, S. Sako, and T. Kitamura, “Stochastic modeling of a musical performance with expressive representations from the musical score,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2011, pp. 531–536.
- [59] K. Okumura, S. Sako, and T. Kitamura, “Laminae: A stochastic modeling-based autonomous performance rendering system that elucidates performer characteristics,” in *Proceedings of the International Computer Music Conference (ICMC)*, 2014.
- [60] S. Moulieras and F. Pachet, “Maximum entropy models for generation of expressive music,” *arXiv preprint*, 2016. DOI: 10.48550/arXiv.1610.03606.
- [61] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, “This time with feeling: Learning expressive musical performance,” *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020. DOI: 10.1007/s00521-018-3758-9.
- [62] P. N. Juslin and P. Laukka, “Expression, perception, and induction of musical emotions: A review and a questionnaire study of everyday listening,” *Journal of New Music Research*, vol. 33, no. 3, pp. 217–238, 2004. DOI: 10.1080/0929821042000317813.
- [63] P. N. Juslin and J. Sloboda, *Handbook of music and emotion: theory, research, applications*. Oxford University Press, 2011. DOI: 10.1093/acprof:oso/9780199230143.001.0001.
- [64] T. Eerola and J. K. Vuoskoski, “A review of music and emotion studies: Approaches, emotion models, and stimuli,” *Music Perception*, vol. 30, no. 3, pp. 307–340, Feb. 2013, ISSN: 0730-7829. DOI: 10.1525/mp.2012.30.3.307.
- [65] A. Gabrielsson, “Emotion perceived and emotion felt: Same or different?” *Musicae Scientiae*, vol. 5, no. 1_suppl, pp. 123–147, 2001. DOI: 10.1177/10298649020050S105.
- [66] P. Evans and E. Schubert, “Relationships between expressed and felt emotions in music,” *Musicae Scientiae*, vol. 12, no. 1, pp. 75–99, 2008. DOI: 10.1177/102986490801200105.
- [67] J. A. Russell, “A circumplex model of affect,” *Journal of Personality and Social Psychology*, vol. 39, no. 6, p. 1161, 1980. DOI: 10.1037/h0077714.
- [68] A. Gabrielsson, “Intention and emotional expression in music performance,” in *Proceedings of the Stockholm Music Acoustics Conference (SMAC)*, 1993, pp. 108–111.
- [69] A. Gabrielsson, “Expressive intention and performance,” in *Music and the Mind Machine*, Springer, 1995, pp. 35–47. DOI: 10.1007/978-3-642-79327-1_4.

- [70] S. R. Livingstone, R. Muhlberger, A. R. Brown, and W. F. Thompson, “Changing musical emotion: A computational rule system for modifying score and performance,” *Computer Music journal*, vol. 34, no. 1, pp. 41–64, 2010. DOI: 10.1162/comj.2010.34.1.41.
- [71] R. Bresin and A. Friberg, “Emotion rendering in music: Range and characteristic values of seven musical variables,” *Cortex*, vol. 47, no. 9, pp. 1068–1081, 2011. DOI: 10.1016/j.cortex.2011.05.009.
- [72] R. Bresin, A. Friberg, J. Sundberg, *et al.*, “Director Musices: The kth performance rules system,” *Proceedings of SIGMUS-46*, vol. 2002, no. 63 (2002-MUS-046), pp. 43–48, 2002.
- [73] S. Canazza, G. D. Poli, S. Rinaldin, and A. Vidolin, “Sonological analysis of clarinet expressivity,” in *Proceedings of the Joint International Conference on Cognitive and Systematic Musicology*, Springer, 1996, pp. 431–440. DOI: 10.1007/BFb0034131.
- [74] S. Canazza, G. Poli, A. Rodà, and A. Vidolin, “An abstract control space for communication of sensory expressive intentions in music performance,” *Journal of New Music Research*, vol. 32, no. 3, pp. 281–294, 2003. DOI: 10.1076/jnmr.32.3.281.16862.
- [75] S. Canazza, G. De Poli, C. Drioli, A. Roda, and A. Vidolin, “Modeling and control of expressiveness in music performance,” *Proceedings of the IEEE*, vol. 92, no. 4, pp. 686–701, 2004. DOI: 10.1109/JPROC.2004.825889.
- [76] G. De Poli, “Expressiveness in music performance,” in *Algorithms for Sound and Music Computing*, 2018.
- [77] S. Canazza, G. De Poli, and A. Rodà, “Caro 2.0: An interactive system for expressive music rendering,” *Advances in Human-Computer Interaction*, vol. 2015, 2015. DOI: 10.1155/2015/850474.
- [78] A. Friberg, “Home conducting-control the overall musical expression with gestures,” in *Proceedings of the International Computer Music Conference (ICMC)*, Michigan Publishing, 2005.
- [79] T. Porcello, “Speaking of sound: Language and the professionalization of sound-recording engineers,” *Social Studies of Science*, vol. 34, no. 5, pp. 733–758, 2004. DOI: 10.1177/0306312704047328.
- [80] L. Costalonga, *Biomechanical modelling of guitar performance*. Edufes, 2023.

- [81] C. D. Metcalf, T. A. Irvine, J. L. Sims, Y. L. Wang, A. W. Su, and D. O. Norris, "Complex hand dexterity: A review of biomechanical methods for measuring musical performance," *Frontiers in psychology*, vol. 5, 2014. DOI: 10.3389/fpsyg.2014.00414.
- [82] B. G. Wristen, "Avoiding piano-related injury: A proposed theoretical procedure for biomechanical analysis of piano technique," *Medical Problems of Performing Artists*, vol. 15, no. 2, pp. 55–64, 2000. DOI: 10.21091/mppa.2000.2012.
- [83] R. Parncutt, J. A. Sloboda, E. F. Clarke, M. Raekallio, and P. Desain, "An ergonomic model of keyboard fingering for melodic fragments," *Music Perception*, vol. 14, no. 4, pp. 341–382, 1997. DOI: 10.2307/40285730.
- [84] J. P. Jacobs, "Refinements to the ergonomic model for keyboard fingering of parncutt, sloboda, clarke, raekallio, and desain," *Music Perception*, vol. 18, no. 4, pp. 505–511, 2001. DOI: 10.1525/mp.2001.18.4.505.
- [85] J. Sundberg, A. Friberg, and R. Bresin, "Attempts to reproduce a pianist's expressive timing with Director Musices performance rules," *Journal of New Music Research*,
- [86] R. Bresin, "Artificial neural networks based models for automatic performance of musical scores," *Journal of New Music Research*, vol. 27, no. 3, pp. 239–270, 1998. DOI: 10.1080/09298219808570748.
- [87] E. Stamatatos and G. Widmer, "Automatic identification of music performers with learning ensembles," *Artificial Intelligence*, vol. 165, no. 1, pp. 37–56, 2005, ISSN: 0004-3702. DOI: 10.1016/j.artint.2005.01.007.
- [88] R. Ramirez, E. Maestre, A. Pertusa, E. Gomez, and X. Serra, "Performance-based interpreter identification in saxophone audio recordings," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 3, pp. 356–364, 2007. DOI: 10.1109/TCSVT.2007.890862.
- [89] A. Tobudic and G. Widmer, "Learning to play like the great pianists," in *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, 2005, pp. 871–876.
- [90] S. Giraldo and R. Ramirez, "A machine learning approach to ornamentation modeling and synthesis in jazz guitar," *Journal of Mathematics and Music*, vol. 10, no. 2, pp. 107–126, 2016. DOI: 10.1080/17459737.2016.1207814.
- [91] S. I. Giraldo and R. Ramirez, "A machine learning approach to discover rules for expressive performance actions in jazz guitar music," *Frontiers in psychology*, vol. 7, 2016. DOI: 10.3389/fpsyg.2016.01965.

- [92] M. V. Mathews, "The radio baton and conductor program, or: Pitch, the most important and least expressive part of music," *Computer Music journal*, vol. 15, no. 4, pp. 37–46, 1991. DOI: 10.2307/3681070.
- [93] J. Lawson and M. V. Mathews, "Computer program to control a digital real-time sound synthesizer," *Computer Music journal*, pp. 16–21, 1977.
- [94] M. V. Mathews, "The conductor program and mechanical baton," in *Proceedings of the International Symposium on Music and Information Science*, 1989, pp. 58–70.
- [95] E. Lee, T. M. Nakra, and J. Borchers, "You're the conductor: A realistic interactive conducting system for children," in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*, 2004, pp. 68–73. DOI: 10.5281/zenodo.1176629.
- [96] T. Baba, M. Hashida, and H. Katayose, "'virtualPhilharmony': A conducting system with heuristics of conducting an orchestra," in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*, vol. 2010, 2010, pp. 263–270. DOI: 10.5281/zenodo.1177715.
- [97] J. Langner and W. Goebel, "Representing expressive performance in tempo-loudness space," in *Proceedings of the ESCOM Conference on Musical Creativity*, 2002, pp. 5–8.
- [98] S. Dixon, W. Goebel, and G. Widmer, "Real time tracking and visualisation of musical expression," in *Proceedings of the International Conference on Music and Artificial Intelligence (ICMAI)*, Springer, 2002, pp. 58–68. DOI: 10.1007/3-540-45722-4_7.
- [99] S. Dixon, W. Goebel, and G. Widmer, "The performance worm: Real time visualisation of expression based on langner's tempo-loudness animation," in *Proceedings of the International Computer Music Conference (ICMC)*, Michigan Publishing, 2002. [Online]. Available: <https://hdl.handle.net/2027/spo.bbp2372.2002.073>.
- [100] S. Dixon, W. Goebel, and G. Widmer, "The 'air worm': An interface for real-time manipulation of expressive music performance," in *Proceedings of the International Computer Music Conference (ICMC)*, Michigan Publishing, 2005. [Online]. Available: <https://hdl.handle.net/2027/spo.bbp2372.2005.048>.
- [101] B. Moog, "MIDI," *Journal of Audio Engineering Society*, vol. 34, no. 5, 1986.
- [102] M. Good, "MusicXML: An internet-friendly format for sheet music," in *Proceedings of XML conference and exposition*, 2001, pp. 03–04.
- [103] N. Fradet, J.-P. Briot, F. Chhel, A. E. F. Seghrouchni, and N. Gutowski, "MidiTok: A python package for MIDI file tokenization," in *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference*, 2021.

- [104] Q. D. Bowers, *Encyclopedia of automatic musical instruments*. Vestal Press, 1972.
- [105] P. Bontempi, P. Zappalà, and S. Canazza, “Expressiveness in metronomic rolls for player piano: A new field of research?” *Journal of Cultural Heritage*, vol. 67, pp. 414–420, 2024, ISSN: 1296-2074. DOI: 10.1016/j.culher.2024.04.004.
- [106] A. W. Ord-Hume, “Player piano,” in *Grove Music Online*, Oxford University Press, 2001, ISBN: 9781561592630. DOI: 10.1093/gmo/9781561592630.article.21928.
- [107] B. Dolan, *Inventing entertainment: The player piano and the origins of an american musical industry*. Rowman & Littlefield Publishers, 2009.
- [108] E. T. Johnson, “Player piano,” in *Grove Music Online*, Oxford University Press, Oct. 2013, ISBN: 9781561592630. DOI: 10.1093/gmo/9781561592630.article.A2253760.
- [109] F. W. Holland, A. W. Ord-Hume, and L. Sitsky, “Reproducing piano,” in *Grove Music Online*, Oxford University Press, 2001, ISBN: 9781561592630. DOI: 10.1093/gmo/9781561592630.article.52058.
- [110] S. Canazza and G. De Poli, “Four decades of music research, creation, and education at Padua’s Centro di Sonologia Computazionale,” *Computer Music journal*, vol. 43, no. 4, pp. 58–80, 2019. DOI: 10.1162/comj_a_00537.
- [111] S. Canazza, G. De Poli, and A. Vidolin, “Gesture, music and computer: The Centro di Sonologia Computazionale at Padova University, a 50-year history,” *Sensors*, vol. 22, no. 9, 2022, ISSN: 1424-8220. DOI: 10.3390/s22093465.
- [112] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 96, 1996, pp. 226–231.
- [113] M. F. McKinney, D. Moelants, M. E. P. Davies, and A. Klapuri, “Evaluation of audio beat tracking and music tempo extraction algorithms,” *Journal of New Music Research*, vol. 36, no. 1, pp. 1–16, 2007. DOI: 10.1080/09298210701653252.
- [114] O. M. Bjørndalen, *Mido: Midi objects for python*, 2023. [Online]. Available: <https://mido.readthedocs.io/en/latest/>.
- [115] J. Cournut, L. Bigo, M. Giraud, N. Martin, and D. Régnier, “What are the most used guitar positions?” In *Proceedings of the International Conference on Digital Libraries for Musicology (DLfM)*, 2021, pp. 84–92. DOI: 10.1145/3469013.3469024.

- [116] H. Heijink and R. G. Meulenbroek, “On the complexity of classical guitar playing: Functional adaptations to task constraints,” *Journal of Motor Behavior*, vol. 34, no. 4, pp. 339–351, 2002. DOI: 10.1080/00222890209601952.
- [117] S. I. Sayegh, “Fingering for string instruments with the optimum path paradigm,” *Computer Music journal*, vol. 13, no. 3, pp. 76–84, 1989. DOI: 10.2307/3680014.
- [118] Sayegh and Tenorio, “Inverse Viterbi algorithm as learning procedure and application to optimization in the string instrument fingering problem,” in *Proceedings of the IEEE International Conference on Neural Networks*, IEEE, 1988, pp. 491–497.
- [119] D. Radicioni, L. Anselma, V. Lombardo, *et al.*, “A segmentation-based prototype to compute string instruments fingering,” in *Proceedings of the Conference on Interdisciplinary Musicology*, vol. 17, 2004, p. 97.
- [120] M. Miura, I. Hirota, N. Hama, and M. Yanagida, “Constructing a system for finger-position determination and tablature generation for playing melodies on guitars,” *Systems and Computers in Japan*, vol. 35, no. 6, pp. 10–19, 2004. DOI: 10.1002/scj.10609.
- [121] D. Tuohy and W. Potter, “A genetic algorithm for the automatic generation of playable guitar tablature,” in *Proceedings of the International Computer Music Conference (ICMC)*, Jan. 2005, pp. 499–502.
- [122] G. Hori, H. Kameoka, and S. Sagayama, “Input-output hmm applied to automatic arrangement for guitars,” *Journal of Information Processing*, vol. 21, no. 2, pp. 264–271, 2013. DOI: 10.2197/ipsjjip.21.264.
- [123] A. Viterbi, “Error bounds for convolutional codes and an asymptotically optimum decoding algorithm,” *IEEE transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.
- [124] Y. Bengio and P. Frasconi, “Input-output HMMs for sequence processing,” *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1231–1249, 1996.
- [125] G. Hori and S. Sagayama, “Minimax Viterbi algorithm for hmm-based guitar fingering decision,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 448–453.
- [126] T. H. Raboanary, F. H. Randriamahanintsoa, H. A. Raboanary, T. M. Raboanary, and J. A. Raboanary, “Finding optimal bass guitar fingerings,” in *Proceedings of the IEEE Africon Conference*, IEEE, 2017, pp. 65–71. DOI: 10.1109/AFRCON.2017.8095457.

- [127] A. D’Hooge, L. Bigo, K. Déguernel, and N. Martin, “Guitar chord diagram suggestion for western popular music,” in *Proceedings of the Sound and Music Computing Conference (SMC)*, 2024. DOI: 10.48550/arXiv.2407.14260.
- [128] F. Carnovalini and A. Rodà, “Computational Creativity and Music Generation Systems: An Introduction to the State of the Art,” *Frontiers in Artificial Intelligence*, vol. 3, p. 14, 2020, ISSN: 2624-8212. DOI: 10.3389/frai.2020.00014.
- [129] J. Biles, “GenJam: A genetic algorithm for generating jazz solos,” in *Proceedings of the International Computer Music Conference (ICMC)*, Ann Arbor, MI, vol. 94, 1994, pp. 131–137.
- [130] M. McVicar, S. Fukayama, and M. Goto, “AutoLeadGuitar: Automatic generation of guitar solo phrases in the tablature space,” in *Proceedings of the International Conference on Signal Processing (ICSP)*, IEEE, 2014, pp. 599–604.
- [131] N. d. S. Cunha, A. Subramanian, and D. Herremans, “Generating guitar solos by integer programming,” *Journal of the Operational Research Society*, vol. 69, no. 6, pp. 971–985, 2018.
- [132] P. Sarmento, A. Kumar, D. Xie, C. Carr, Z. Zukowski, and M. Barthet, “ShredGP: Guitarist style-conditioned tablature generation with transformers,” in *Proceedings of the International Symposium on Computer Music Multidisciplinary Research*, Zenodo, 2023. DOI: 10.5281/zenodo.10129720.
- [133] D. R. Grimes, “String theory - the physics of string-bending and other electric guitar techniques,” *PLOS ONE*, vol. 9, no. 7, pp. 1–9, Jul. 2014. DOI: 10.1371/journal.pone.0102088.
- [134] C. Goddard, M. Barthet, and G. A. Wiggins, “Designing computationally creative musical performance systems,” in *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*, 2017, pp. 1–8. DOI: 10.1145/3123514.3123541.
- [135] P. Sarmento, A. Kumar, C. Carr, Z. Zukowski, M. Barthet, and Y.-H. Yang, “DadaGP: A dataset of tokenized guitarpro songs for sequence models,” *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.
- [136] D. Régnier, N. Martin, and L. Bigo, “Identification of rhythm guitar sections in symbolic tablatures,” en, in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2021.
- [137] P. Fischer, *Masters of Rock Guitar*. Ama Verlag, 2000.
- [138] T. Stetina, *Metal lead guitar*. Hal Leonard, 1992.

- [139] J. Wheatcroft, *Improvising Blues Guitar*. Schott & Co. Ltd, 2008.
- [140] S. Canazza, G. D. Poli, A. Rodà, and A. Vidolin, “Analysis by synthesis of the expressive intentions in musical performance,” in *Proceedings of the International Computer Music Conference (ICMC)*, Michigan Publishing, 1997. [Online]. Available: <https://hdl.handle.net/2027/spo.bbp2372.1997.036>.
- [141] L. Su, L.-F. Yu, and Y.-H. Yang, “Sparse cepstral, phase codes for guitar playing technique classification,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 9–14.
- [142] M. Stein, J. Abeßer, C. Dittmar, and G. Schuller, “Automatic detection of audio effects in guitar and bass recordings,” in *Proceedings of the Audio Engineering Society Convention*, Audio Engineering Society, 2010.
- [143] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, “GuitarSet: A dataset for guitar transcription,” in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 453–460.
- [144] Y.-H. Chen, W.-Y. Hsiao, T.-K. Hsieh, J.-S. R. Jang, and Y.-H. Yang, “Towards automatic transcription of polyphonic electric guitar music: A new dataset and a multi-loss transformer model,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 786–790. DOI: 10.48550/arXiv.2202.09907.
- [145] X. Riley, Z. Guo, D. Edwards, and S. Dixon, “GAPS: A large and diverse classical guitar dataset and benchmark transcription model,” *arXiv preprint*, 2024. DOI: 10.48550/arXiv.2408.08653.
- [146] C. Marco, S. Dan, and R. J. D., “Guitar effects recognition and parameter estimation with convolutional neural networks,” *Journal of the Audio Engineering Society*, vol. 69, pp. 594–604, 7/8 2021. DOI: 10.17743/jaes.2021.0019.
- [147] A. Mitsou, A. Petrogianni, E. A. Vakalaki, C. Nikou, T. Psallidas, and T. Giannakopoulos, “A multimodal dataset for electric guitar playing technique recognition,” *Data in Brief*, vol. 52, p. 109 842, 2024, ISSN: 2352-3409. DOI: 10.1016/j.dib.2023.109842.
- [148] S. Kim, T. Hayashi, and T. Toda, “Note-level automatic guitar transcription using attention mechanism,” in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, IEEE, 2022, pp. 229–233. DOI: 10.23919/EUSIPCO55093.2022.9909659.

- [149] A. M. Barbancho, A. Klapuri, L. J. Tardón, and I. Barbancho, “Automatic transcription of guitar chords and fingering from audio,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 3, pp. 915–921, 2011. DOI: 10.1109/TASL.2011.2174227.
- [150] X. Fiss and A. Kwasinski, “Automatic real-time electric guitar audio transcription,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2011, pp. 373–376.
- [151] K. Yazawa, K. Itoyama, and H. G. Okuno, “Automatic transcription of guitar tablature from audio signals in accordance with player’s proficiency,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2014, pp. 3122–3126.
- [152] E. Chew, A. François, J. Liu, and A. Yang, “ESP: A driving interface for expression synthesis,” in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*, Vancouver, Canada: National University of Singapore, 2005, pp. 224–227.
- [153] Y. Yun and S.-H. Cha, “Designing virtual instruments for computer music,” *International Journal of Multimedia and Ubiquitous Engineering*, vol. 8, no. 5, pp. 173–178, 2013. DOI: 10.14257/ijmue.2013.8.5.16.
- [154] G. Tanev and A. Božinovski, “Virtual Studio Technology inside music production,” in *Proceedings of the ICT Innovations Conference*, V. Trajkovik and M. Anastas, Eds., Heidelberg: Springer, 2014, pp. 231–241, ISBN: 978-3-319-01466-1. DOI: 10.1007/978-3-319-01466-1_22.
- [155] W. C. Pirkle, *Designing Software Synthesizer Plugins in C++: With Audio DSP*. Focal Press, 2021.
- [156] H. Hennig, R. Fleischmann, A. Fredebohm, *et al.*, “The nature and perception of fluctuations in human musical rhythms,” *PloS one*, vol. 6, no. 10, e26457, 2011. DOI: 10.1371/journal.pone.0026457.
- [157] K. Worrall, Z. Yin, and T. Collins, “Comparative evaluation in the wild: Systems for the expressive rendering of music,” *IEEE Transactions on Artificial Intelligence*, 2024. DOI: 10.1109/TAI.2024.3408717.
- [158] “Organic mixdowns,” *Computer Music*, vol. 224, 2015.
- [159] E. Lindemann, “Music synthesis with reconstructive phrase modeling,” *IEEE Signal Processing Magazine*, vol. 24, no. 2, pp. 80–91, 2007. DOI: 10.1109/MSP.2007.323267.

- [160] C. Drioli and R. D. Federico, “Toward an integrated sound analysis and processing framework for expressiveness rendering,” in *Proceedings of the International Computer Music Conference (ICMC)*, Michigan Publishing, 1998. [Online]. Available: <https://hdl.handle.net/2027/spo.bbp2372.1998.508>.
- [161] S. Canazza, G. De Poli, C. Drioli, A. Rodà, and F. Zamperini, “Real-time morphing among different expressive intentions in audio playback,” in *Proceedings of the International Computer Music Conference (ICMC)*, 2000.
- [162] S. Canazza, G. De Poli, C. Drioli, A. Rodà, and A. Vidolin, “Expressive morphing for interactive performance of musical scores,” in *Proceedings First International Conference on WEB Delivering of Music (WEDELMUSIC)*, IEEE, 2001, pp. 116–122.
- [163] J. L. Arcos, R. L. De Mántaras, and X. Serra, “Saxex: A case-based reasoning system for generating expressive musical performances,” *Journal of New Music Research*, vol. 27, no. 3, pp. 194–210, 1998.
- [164] N. Jonason, X. Wang, E. Cooper, L. Juvela, B. L. Sturm, and J. Yamagishi, “DDSP-based neural waveform synthesis of polyphonic guitar performance from string-wise MIDI input,” *arXiv preprint*, 2023. DOI: 10.48550/arXiv.2309.07658.
- [165] D. H. Keefe, “Physical modeling of wind instruments,” *Computer Music journal*, vol. 16, no. 4, pp. 57–73, 1992.
- [166] V. Välimäki and T. Takala, “Virtual musical instruments—natural sound using physical models,” *Organised Sound*, vol. 1, no. 2, pp. 75–86, 1996.
- [167] X. Rodet and C. Vergez, “Nonlinear dynamics in physical models: From basic models to true musical-instrument models,” *Computer Music journal*, vol. 23, no. 3, pp. 35–49, 1999.
- [168] M. Sterling, X. Dong, and M. Bocko, “Representation of solo clarinet music by physical modeling synthesis,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008, pp. 129–132. DOI: 10.1109/ICASSP.2008.4517563.
- [169] S. Bilbao, B. Hamilton, A. Torin, *et al.*, “Large scale physical modeling sound synthesis,” in *Proceedings of the Stockholm Musical Acoustics Conference/Sound and Music Computing Conference*, 2013.
- [170] R. Ciesla, “Introducing MIDI 2.0,” in *Sound and Music for Games: The Basics of Digital Audio for Video Games*. Berkeley, CA: Apress, 2022, pp. 187–196, ISBN: 978-1-4842-8661-6. DOI: 10.1007/978-1-4842-8661-6_7.

- [171] L. Sean and L. Ludovico, “MIDI 1.5, so to speak,” in *Proceedings of the Sound and Music Computing Conference (SMC)*, 2024.
- [172] S. Fasciani and J. Goode, “20 NIMes: Twenty years of new interfaces for musical expression,” in *Proceedings of the Conference on New Interfaces for Musical Expression (NIME)*, Shanghai, China, 2021. DOI: 10.21428/92fbeb44.b368bcd5.
- [173] R. Lamb and A. Robertson, “Seaboard: A new piano keyboard-related interface combining discrete and continuous control,” in *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*, 2011, pp. 503–506. DOI: 10.5281/zenodo.1178081.
- [174] F. Liarokapis, “Augmented reality scenarios for guitar learning,” in *EG UK Theory and Practice of Computer Graphics*, 2005, pp. 163–170. DOI: 10.2312 / LocalChapterEvents/TPCG/TPCGUK05/163-170.
- [175] S. J. Havre, L. Väkevä, C. R. Christophersen, and E. Haugland, “Playing to learn or learning to play? Playing Rocksmith to learn electric guitar and bass in nordic music teacher education,” *British Journal of Music Education*, vol. 36, no. 1, pp. 21–32, 2019. DOI: 10.1017/S026505171800027X.
- [176] R. Hidayatullah, P. Tejapermana, A. H. Hernanda, C. A. N. Mazlan, and H. Hasyimkan, “Guitar pro as learning tools in major class,” *Musikolastika*, vol. 6, no. 1, pp. 56–67, 2024.
- [177] F. Carnovalini and A. Rodà, “A multilayered approach to automatic music generation and expressive performance,” in *Proceedings of the International Workshop on Multilayer Music Representation and Processing (MMRP)*, 2019, pp. 41–48. DOI: 10.1109/MMRP.2019.00016.
- [178] D. Edwards, X. Riley, P. Sarmento, and S. Dixon, “MIDI-to-tab: Guitar tablature inference via masked language modeling,” *arXiv preprint*, 2024. DOI: 10.48550 / arXiv.2408.05024.
- [179] P. Van den Besselaar, G. Heimeriks, *et al.*, “Disciplinary, multidisciplinary, interdisciplinary: Concepts and indicators,” in *Proceedings of the International Conference on Scientometrics and Informetrics (ISSI)*, 2001, pp. 705–716.
- [180] A. Dalton, K. Wolff, and B. Bekker, “Multidisciplinary research as a complex system,” *International Journal of Qualitative Methods*, vol. 20, p. 16 094 069 211 038 400, 2021. DOI: 10.1177/16094069211038400.
- [181] T. Sanchis, “Strategies for multidisciplinary research,” *Nature Physics*, vol. 19, no. 12, pp. 1736–1737, 2023, ISSN: 1745-2481. DOI: 10.1038/s41567-023-02277-z.

- [182] M. Vantard, C. Galland, and M. Knoop, “Interdisciplinary research: Motivations and challenges for researcher careers,” *Quantitative Science Studies*, vol. 4, no. 3, pp. 711–727, 2023. DOI: 10.1162/qss_a_00265.
- [183] C. S. Wagner, J. D. Roessner, K. Bobb, *et al.*, “Approaches to understanding and measuring interdisciplinary scientific research (IDR): A review of the literature,” *Journal of informetrics*, vol. 5, no. 1, pp. 14–26, 2011. DOI: 10.1016/j.joi.2010.06.004.
- [184] B. K. Laursen, N. Motzer, and K. J. Anderson, “Pathways for assessing interdisciplinarity: A systematic review,” *Research Evaluation*, vol. 31, no. 3, pp. 326–343, Jul. 2022, ISSN: 0958-2029. DOI: 10.1093/reseval/rvac013.
- [185] A. Zuccante, A. Fiordelmondo, P. Bontempi, and S. Canazza, “A signal-processing-based simulation system for high-end stereo headsets,” *Sensors*, vol. 24, no. 7, p. 2190, 2024. DOI: 10.3390/s24072190.
- [186] P. Bontempi and S. Canazza, “Expressiveness in popular music performance and production: A reasoned review of interviews with leading engineers, producers and performers,” *Journal of the International Association for the Study of Popular Music (IASPM)* - Under review, publication date to be determined following completion of reviews.
- [187] F. Frontini, *Enhancement of perforated rolls for player piano: Design and development of software for converting scans into MIDI format*, Bachelor’s degree thesis, Available at <https://hdl.handle.net/20.500.12608/53326>, 2023.
- [188] C. Costantini, *Design of the sound-based nonverbal human-computer interaction for automatic expressive music performance systems*, Bachelor’s degree thesis, Available at <https://hdl.handle.net/20.500.12608/57082>, 2023.
- [189] P. Bontempi, *Produzione audio e musicale con Cubase. Percorso teorico e pratico dalle basi ai concetti avanzati*. Milano: LSWR, 2024.
- [190] M. Senior, *Recording secrets for the small studio. I segreti della registrazione per gli home studio e i piccoli studi*. Milano: Volontè & Co., 2022.
- [191] B. Owsinski, *The Mixing Engineer. Manuale completo*. Milano: Volontè & Co., 2024.