Automatic Guitar Tablature Transcription Online

Gregory D. Burlet

Music Technology Area
Department of Music Research
Schulich School of Music
McGill University · Montréal, Québec

April 2013

A thesis submitted to McGill University in partial fulfilment of the requirements for the degree of Master of Arts.

We are drowning in information and starving for knowledge.

—RUTHERFORD D. ROGER

Abstract

Manually transcribing guitar tablature from an audio recording is a difficult and time-consuming process, even for experienced guitarists. While several algorithms have been developed to automatically extract the notes occurring in an audio recording, and several algorithms have been developed to produce guitar tablature arrangements of notes occurring in a music score, no frameworks have been developed to facilitate the combination of these algorithms. This work presents a web-based guitar tablature transcription framework capable of generating guitar tablature arrangements directly from an audio recording.

The implemented transcription framework, entitled *Robotaba*, facilitates the creation of web applications in which polyphonic transcription and guitar tablature arrangement algorithms can be embedded. Such a web application is implemented, resulting in a unified system that is capable of transcribing guitar tablature from a digital audio recording and displaying the resulting tablature in the web browser. The performance of the implemented polyphonic transcription and guitar tablature arrangement algorithms are evaluated using several metrics on a new dataset of manual transcriptions gathered from tablature websites.

Résumé

Transcrire à la main une tablature pour guitare à partir d'un enregistrement audio est un processus difficile et long, même pour les guitaristes chevronnés. Bien que plusieurs algorithmes aient été créés pour extraire automatiquement les notes d'un enregistrement audio, et d'autres pour préparer des arrangements de notes de tablature pour guitare tels qu'on les retrouve dans la création musicale, aucun environnement n'a été mise en place pour faciliter l'association de ces algorithmes. Le travail qui suit présente un environnement accessible sur l'Internet, permettant la transcription et la préparation d'arrangements de tablatures de guitare, directement à partir d'un enregistrement audio.

Cet environnement de transcription, nommée Robotaba, facilite la création d'applications Web, dans lesquelles la transcription polyphonique et les algorithmes d'arrangements de tablature pour guitare peuvent être intégrés. Une telle application Web permet d'obtenir un système unifié, capable de transcrire une tablature pour guitare à partir d'un enregistrement audio numérique, et d'afficher la tablature obtenue dans un navigateur Web. La performance de la transcription polyphonique mise en place et des algorithmes d'arrangements de tablature pour guitare est évaluée à l'aide de plusieurs paramètres et d'un nouvel ensemble de données, constitué de transcriptions manuelles recueillies dans des sites Web consacrés aux tablatures.

Acknowledgements

I would like to acknowledge and thank Ruohua Zhou and Joshua Reiss for the open-source implementation of the polyphonic transcription algorithm used in this thesis. I also wish to express my gratitude to those individuals who have uploaded manual tablature transcriptions to the Ultimate Guitar website. In the interest of space these people will go unnamed; however, their contribution to the datasets compiled in this thesis is substantial.

I have been fortunate to work in the stimulating environment of the Distributed Digital Music Archives and Libraries laboratory throughout the course of this work. My sincerest thanks are extended to my colleagues and friends, who are always willing to take time out of their busy schedules to provide insight into a problem or provide comedic relief. In particular, I wish to thank Hannah Robertson for proofreading segments of this work. I would also care to thank Hélène Floch de Gallaix-Boude for translating the abstract of this thesis.

Special thanks are owed to my advisor Ichiro Fujinaga, who introduced me to the field of Music Information Retrieval and has provided invaluable advice and feedback over the course of this work.

My heartfelt thanks are extended to my partner for her patience, understanding, and love throughout the creation of this work. I would also like to thank my parents for their encouragement and advice in all of my endeavours, as well as their financial support.

This research was supported by a Joseph-Armand Bombardier Canada Graduate Master's Scholarship provided by the Social Sciences and Humanities Research Council of Canada.

Contents

List of Figures

List of Tables

Code Listings

List of Acronyms

ASCII	American standard code for information interchange
DP	dynamic programming
ER	entity relationship
FFT	fast Fourier transform
GA	genetic algorithm
GMM	Gaussian mixture model
HMM	hidden Markov model
HTML	hypertext markup language
MEI	music encoding initiative
MIDI	musical instrument digital interface
MIR	music information retrieval
MIREX	music information retrieval evaluation exchange
NMF	non-negative matrix factorization
RTFI	resonator time-frequency image
SQL	structured query language
STFT	short-time Fourier transform
SVG	scalable vector graphic
SVM	support vector machine
UML	unified modeling language
URL	uniform resource locator
XML	extensible markup language

Chapter 1

Introduction

TABLATURE, or less formally, "tabs", have become the primary form of communication between guitarists on the Internet. Tablature is an unstandardized music notation system that defines gestural information and operations to be applied to a specific instrument, such as "depress the fourth fret on the fifth string from the bottom and pluck". Symbols in tablature indicate the instrument fingering required to produce a specific pitch. In this way, pitch information is represented implicitly rather than explicitly. Rhythmic information may be conveyed through the relative spacing of musical symbols, through specialized glyphs above the staff, or not at all. This differs from common Western music notation where standardized musical symbols convey both pitch and rhythm information explicitly, but the details of how to perform the musical score on an instrument is left to the musician. Tablature notation is a symbolic representation of instrument commands that is frequently used on the Internet.

The driving force behind the popularity of tablature is that the notation is accessible to a large body of people. This is true both physically, in the sense that tablature is readily available online, but also intellectually, in the sense that the notation system is easy to understand, decipher, and translate into an instrumental performance. The transcription of music in tablature format has enabled members of the musical community who are not fluent readers or writers of common Western music notation to perform and share their interpretations of a musical work. The ease of access of tablature notation has contributed to its popularity in the musical community.

On the Internet, guitar pieces are frequently displayed in tablature notation. Guitar tablature is a music notation system with a six-line staff that represents the strings on a guitar. A numeric entry on a line represents the fret to depress on a particular string. In order to share tablature on a website, guitarists must manually input fret numbers into the computer for each note in the musical work. If the tablature to perform a song is unknown, the guitarist must first perform the task of manual transcription. This task is accomplished by estimating the pitches of notes in a guitar recording, either by ear or with the aid of audio analysis tools such as *Transcribe!*[1]. The guitarist must then choose from a set of candidate string and fret combinations for each note to produce a tablature arrangement. Manual transcription is a difficult and laborious task even for experienced guitarists.

Researchers in the field of music information retrieval (MIR)—a multi-disciplinary field combining Electrical Engineering, Computing Science, and Music—aim to automate the task of manual tablature transcription. To this end, several algorithms have been proposed to accomplish the task of polyphonic transcription, which seeks to estimate the note events occurring in an audio recording. Although the automatic transcription of monophonic (one note at a time) musical passages is considered a solved problem (Benetos et al. 2012), polyphonic (multiple notes at the same time) transcription is still an open problem. As well, several algorithms have been proposed to accomplish the task of guitar tablature arrangement, which seeks to assign a string and fret combination to each note in a musical score such that the resulting tablature arrangement is easy for a human to perform. However, there is a gap in the current body of research for which no bridge has been built: there does not exist any tools to facilitate the combination of polyphonic transcription and guitar tablature arrangement algorithms to automatically generate tablature from a guitar recording. The work presented in this thesis addresses this discontinuity.

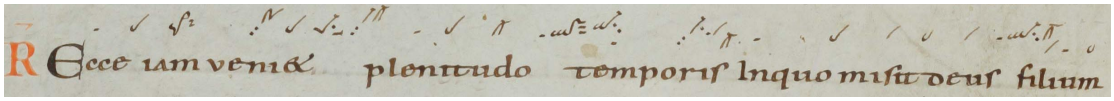
This chapter will begin by introducing and describing the history of tablature notation. Section 1.2 will provide an overview of the work entailed in this thesis, followed by an outline of the structure of this thesis in Section 1.3.

1.1 Evolution of Tablature

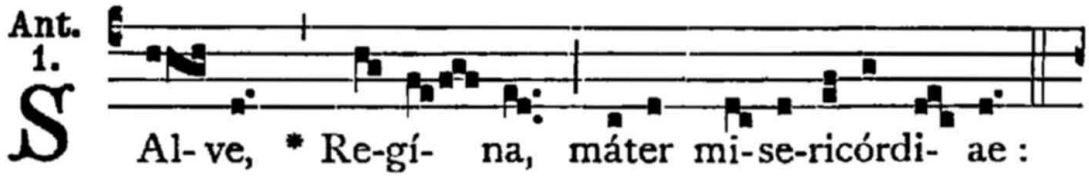
Many music notation systems have been proposed over the centuries. Early notation systems have evolved into their modern counterparts through the process of natural selection—

[1] `http://www.seventhstring.com/xscribe`

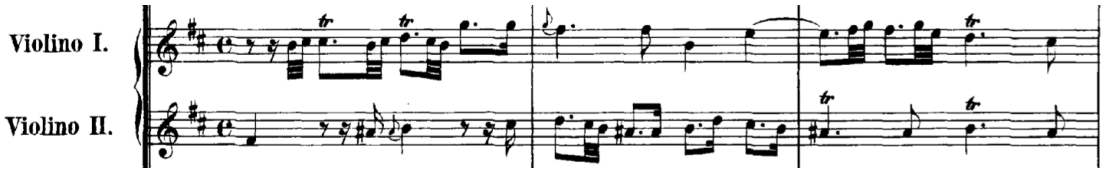
the systematic variation and combination of desirable properties of existing notation systems. For example, modern staff notation in Western culture evolved from staffless neume notation (Figure 1.1(a)) in the ninth century. In this system of notation, a neume represents the pitch contour of a vocal melody spanning a single syllable of chant. Four-line staves and clefs were later introduced in square-note notation (Figure 1.1(b)), which became five lines in the sixteenth century, and eventually evolved into the modern Western music notation used today (Figure 1.1(c)[2]). Tablature notation also has a large evolutionary tree, with many variations in syntax and intended meaning.



(a) System of music in staffless neume notation from the Antiphonarium Officii (Carolingian 990)



(b) System of music in square-note notation from the *Liber Usualis* (Catholic Church 1963)



(c) Staves of music in modern staff notation from BWV 1067 (Bach 1885).

Fig. 1.1: The evolution of common Western music notation.

1.1.1 History of Tablature

The word tablature originates from the Latin word *tabula*, referring to a writing tablet. In the modern English language, the word *tabulate* is a verb describing the act of noting a record in a table. This relates to the act of transcribing tablature for plucked string instruments, where a system of tablature resembles numeric entries in a table.

[2] http://imslp.org/wiki/Orchestral_Suite_No.2_in_B_minor,_BWV_1067_(Bach,_Johann_Sebastian)

Since tablature notation defines operations to be applied to a specific instrument to produce sound, it follows that there are different notations for different instruments. Tablature is classified according to the instrument to which it belongs, and further, the nation of the tablature's origin (Apel 1953). For example, tablature may be referred to as French lute tablature, German organ tablature, and so on. Tablature is most common for stringed instruments such as the bass guitar, electric or acoustic guitar, lute, vihuela, cittern, and harp. In the case of stringed instruments with frets, the tablature notation consists of systems of lines that correspond to the courses (groupings) of strings on the instrument, as well as numerical entries on the lines which denote the fret to depress on a specific string. An example of vihuela tablature can be seen in Figure 1.3(a). Tablature also exists for wind instruments such as the ocarina and harmonica, where diagrams or symbols show which air holes are to be open and closed. Figure 1.2 displays an example of an ocarina tablature of the melody "Serenade of Water" from the video game *The Legend of Zelda: Ocarina of Time*.[3] Theoretically, any instrument where the operations required to play the instrument can be discretely defined and represented symbolically can have a tablature notation.

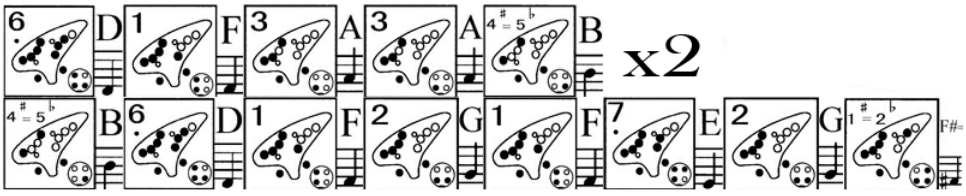


Fig. 1.2: Ocarina tablature of the melody "Serenade of Water" from the video game *The Legend of Zelda: Ocarina of Time*.

Just as common music notation evolved from early music notation systems such as neume and mensural notation, modern tablature notation for stringed instruments also evolved from early tablature notation systems dating back to the Middle Ages. In the Renaissance, there were two main variations of tablature notation: Spanish and Italian tablature, and French tablature (Turnbull 1991). Spanish and Italian tablature used numbers to specify which frets to depress. The top line of the staff represented the lowest pitched course of strings on the instrument. An example of Spanish tablature can be seen in Figure 1.3(a). French tablature notation used letters of the alphabet to indicate which

[3] `http://ocarina-tabs.com/2011/11/30/brendoges-collection-zelda-ocarina-of-time`

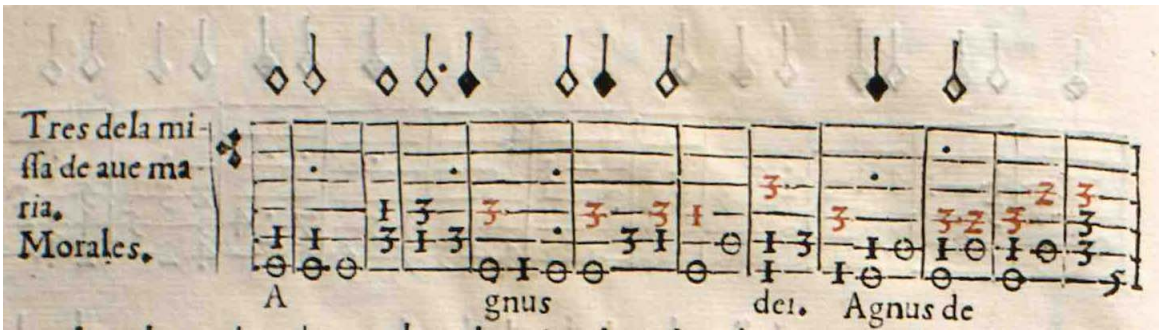
fret to depress. The letter 'a' represents an open string, 'b' indicates a depression of the first fret, and so forth. In contrast to Spanish and Italian tablature, the top line of the staff represented the highest pitched course of strings on the instrument. In the case of instruments with courses of unfretted strings, a pluck of an unfretted course is denoted by a sequence of forward slashes.[4] To demonstrate, '/' denotes a pluck of the first unfretted course, '//' is the second unfretted course, and so on. A system of French tablature can be seen in Figure 1.3(b). In both of these examples of tablature, rhythmic information is displayed above the system in mensural notation.

Modern tablature notation for guitar, as seen in Figure 1.3(c), inherited features of both Spanish and Italian tablature, and French tablature. From Spanish and Italian tablature, modern tablature notation uses numbers to denote which frets to depress, but has a similar orientation to that of French tablature, where the top line of the system represents the highest pitched string. Symbols may appear before, after, or between numbers to indicate a range of note ornamentations such as vibrato, pitch bends, and slides. Since these symbols are unstandardized, tablature websites and books usually provide a legend for the intended meaning of printed symbols. A list of commonly used guitar tablature symbols can be seen in Table 1.1. In contrast to its notational predecessors, modern tablature notation does not typically display rhythmic information above the staff; though, some forms of modern tablature use the relative spacing between symbols to hint at the strumming or plucking pattern.

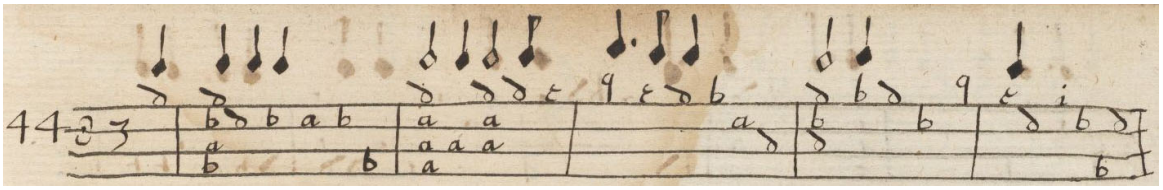
Table 1.1: Commonly used guitar tablature symbols.

Instrument Operation	Symbol	Alternate Symbol
slide up	/	s
slide down	\	s
hammer-on	h	⌒
pull-off	p	⌒
vibrato	∼	∿
mute/dead note	x	×
bend/release	b	↷

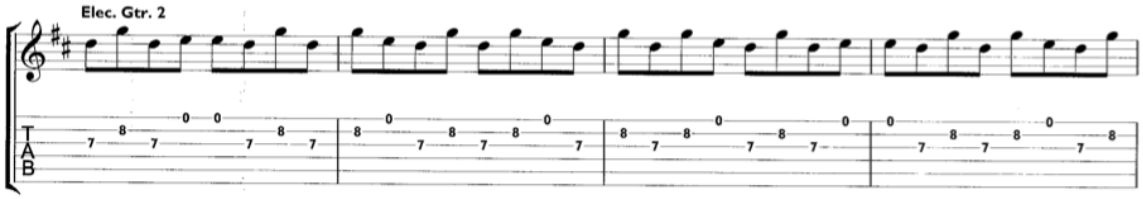
[4]The lute is an example of an instrument that may have both fretted and unfretted courses of strings.



(a) System of Spanish vihuela tablature from the Orphenica Lyra (Fuenllana 1978).



(b) System of French cittern tablature (Otley 1600).



(c) System of modern guitar tablature with common music notation above (Weeks 2008).

Fig. 1.3: The evolution of tablature notation.

Modern tablature for stringed instruments with frets may also be displayed in the form of chord diagrams, which can be thought of as a matrix of boolean values. The rows of the matrix represent the frets of the instrument and the columns of the matrix represent the strings of the instrument. Entry i, j in the matrix represents a depression of the j^{th} string above the i^{th} fret. For the guitar, chord diagrams typically display the first few frets on the fretboard, as seen in Figure 1.4.

1.1.2 Tablature Versus Common Western Music Notation

Tablature notation, although widely used, is not meant to replace standardized music notation systems, but rather pose as an alternative representation that is lightweight and

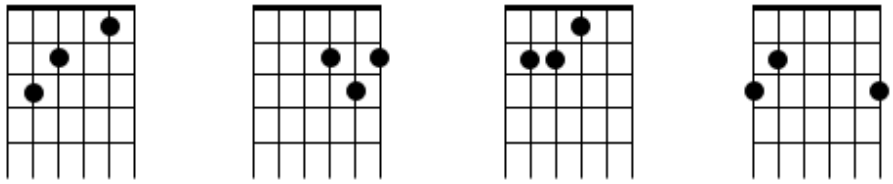


Fig. 1.4: Chord diagrams of a C-Major, D-Major, E-Major, and G-Major guitar chord (from left to right).

easily comprehensible. There are many advantages to a notation system of this nature. Tablature notation provides the sequence of operations required to perform a piece of music on a specific instrument, whereas common Western music notation requires the musician to decide how to interact with their instrument to perform a piece of music. Consequently, tablature notation has a lower barrier to entry than common Western music notation. Tablature notation is minimal by design, and therefore requires minimal training to comprehend and minimal time to write. Conversely, common Western music notation requires years of training to achieve a similar level of expertise. In comparison to common Western music notation, tablature has a visual representation which is closer to that of the intended performance instrument. For example, a system of guitar tablature is made to resemble the fretboard of a guitar. Finally, guitar tablature can be represented in plain text format for simple display in the web browser. Tablatures in this format are commonly referred to as "ASCII tabs" because they consist of a sequence of ASCII characters. An example of an ASCII guitar tab can be seen in Figure 1.5.[5]

There are also drawbacks of representing a music score in tablature notation. Most significantly, modern tablature notation provides reduced, or no rhythm information to the musician. Although the explicit description of instrument operations provides more information than common Western music notation, it may suppress the style of the musician who may have different views on how the piece should be performed. Furthermore, the language and analytical tools of music theory revolve around common Western music notation, not tablature notation. Despite its weaknesses, tablature notation has many strengths that have contributed to its popularity in the musical community.

[5] `http://tabs.ultimate-guitar.com/j/jimmy_eat_world/23_tab.htm`

```
solo

e|-----------------------------------------10-----10-----10--|
B|--15-15-15--14-14-14--12-12-12--10/8---7h8----7h8----7h8-----|
G|-------------------------------------------------------------|
D|-------------------------------------------------------------|
A|-------------------------------------------------------------|
E|-------------------------------------------------------------|
```

Fig. 1.5: A segment of guitar tablature in ASCII format from the song "23" by Jimmy Eat World.

1.1.3 Tablature Websites

As the Internet became accessible to a large body of musicians, many websites emerged where tablature could be uploaded, displayed, and peer-reviewed. *Ultimate Guitar* is a popular website which hosts guitar, bass guitar, and drum tabs.[6] The tablature may be viewed in the web browser as ASCII tabs or within *Tab Pro*—a proprietary Adobe Flash web application which renders tablature in the web browser and provides multi-track audio playback of tablature using synthesized instruments. Some tablature uploaded to Ultimate Guitar is available for download as a *Guitar Pro* file. Guitar Pro is a proprietary desktop application which allows multi-track tablature viewing, editing, and synthesis of a wide variety of stringed and percussive instruments.[7] Other popular tablature websites and search engines include *Songsterr*[8], *911tabs*[9], *Chordie*[10], and *Guitare Tab*[11]. Website analytics for each of these sites over the past year are compared in Figure 1.6 using the metric of unique visitors from the United States per month. This information was compiled from the public analytics data from *Compete*.[12]

With the large database of transcriptions available on these tablature web sites, evaluating the quality of each tablature is important for ordering query results and moderating tablature uploads. Ultimate Guitar has a five-star rating system in which different tabla-

[6] http://www.ultimate-guitar.com
[7] http://www.guitar-pro.com
[8] http://www.songsterr.com
[9] http://www.911tabs.com
[10] http://www.chordie.com
[11] http://www.guitaretab.com
[12] http://www.compete.com/us

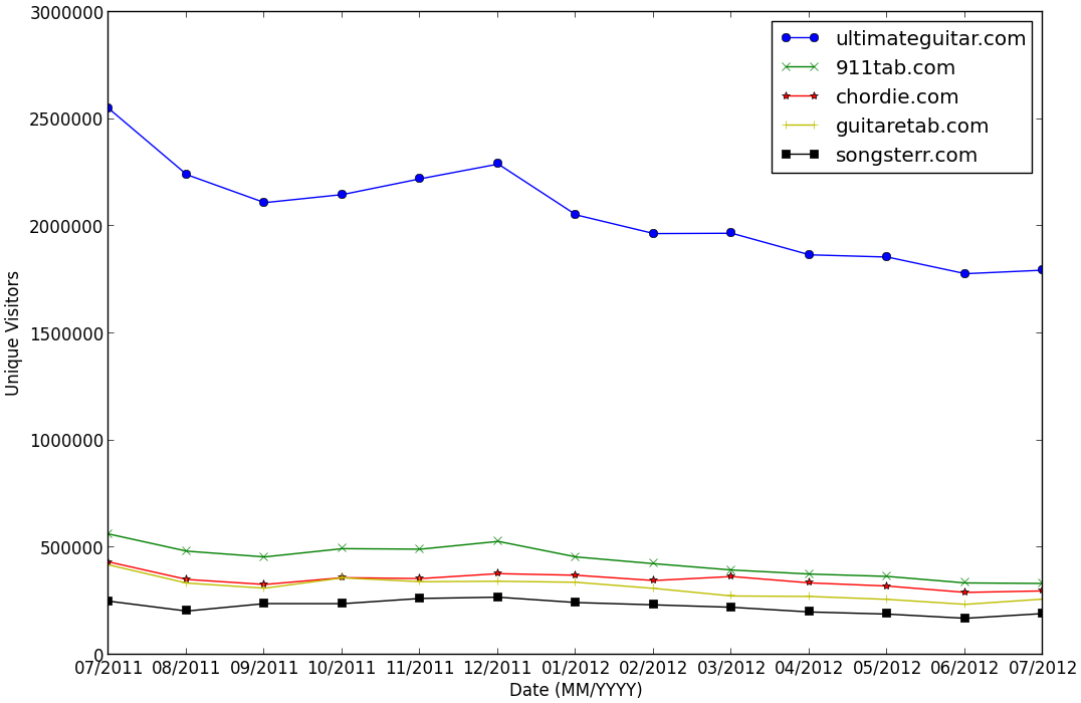


Fig. 1.6: Unique visitors, with a United States IP address, per month to various tablature websites.

ture arrangements of a piece of music can be rated by users according to its correctness and formatting.

1.1.4 Tablature Engraving Online

Since the inception of tablature notation, the process of manually transcribing tablature from an audio recording or live performance has only changed in the method of musical engraving. In the Renaissance, tablature was handwritten using ink and parchment. Before the introduction of personal computers, music engraving involved physically etching musical symbols into metal plates, which were then covered in ink and imprinted onto paper using a printing press. In the modern digital age, the term *music engraving* now refers to the typesetting and display of musical symbols on the computer. In order to differentiate music engraving on the computer from the archaic method of music engraving using metal plates, the term *digital music engraving* will hereinafter refer to music engraving on the computer.

There are several methods of digitally engraving tablature in the web browser. The most simplistic method involves displaying tablature as plain text in the web browser, as shown in Figure 1.5. More sophisticated digital music engraving methods render musical symbols on a digital drawing surface such as the hypertext markup language (HTML) canvas or use alternate multimedia web technologies such as Adobe Flash to display musical symbols. Another method of digital music engraving in the web browser involves defining the shape and position of musical symbols on the page as scalable vector graphics (SVGs), which describe images as a set of lines, curves, and transformations.

Several tools exist to reduce the amount of time required by users to digitally engrave tablature in the web browser. The method of digital music engraving governs the user input required to display the tablature. Displaying the tablature as plain text requires the author to input an ASCII character for each tablature symbol. To counteract this time-consuming process, some tablature websites such as Ultimate Guitar provide an ASCII tab template file that is filled with empty systems of tablature. With this template, the author can focus on the symbolic content of the tablature instead of on tablature formatting. Another tool to facilitate tablature entry is the desktop application Guitar Pro. Guitar Pro provides a graphical interface where the user can enter tablature symbols and export the tablature arrangement to various formats, including plain text format, which can then be displayed in the web browser. To render tablature symbols using the HTML canvas or as SVGs, there exist several digital music engraving libraries such as *VexTab*[13] and *AlphaTab*[14], which perform the low-level drawing commands to produce an aesthetically pleasing digital tablature engraving. These libraries require the user to input a list of formatted keywords and numbers that describe the tablature to be displayed. Although tools exist to reduce the amount of time required for digital tablature engraving, the process of manual tablature entry is still a tedious and demanding process.

1.2 Project Overview

In response to the time-consuming process of manual transcription, this thesis will focus on automating the task of guitar tablature transcription to facilitate tablature entry online. The objective of this thesis is to review state-of-the-art algorithms for polyphonic transcrip-

[13] `http://www.vexflow.com/vextab`
[14] `http://www.alphatab.net`

tion and guitar tablature arrangement and to develop a unifying web-based framework that facilitates the combination of these algorithms. A state-of-the-art polyphonic transcription algorithm (Zhou and Reiss 2008) has been implemented along with a new guitar tablature arrangement algorithm that generates instrument-specific tablature arrangements. The web-based guitar tablature transcription framework, hereinafter referred to as *Robotaba*, is used to create a web application that combines the implemented algorithms to automatically perform tablature transcriptions of guitar recordings and display the resulting tablature in the web browser. The implemented web application will provide a novel and accessible tool for guitarists and a centralized repository of guitar tablature transcriptions for music researchers.

A formal assessment of the implemented guitar tablature transcription web application is performed by evaluating the implemented polyphonic transcription and guitar tablature arrangement algorithms independently. A new ground-truth dataset consisting of synthesized guitar recordings that are each aligned with the correct polyphonic transcription of the recording has been compiled to evaluate the implemented polyphonic transcription algorithm. Additionally, a new ground-truth dataset consisting of human-arranged guitar tablature has been compiled to evaluate the implemented guitar tablature arrangement algorithm.

1.3 Thesis Organization

This thesis is organized as follows: the next chapter provides a literature review of polyphonic transcription algorithms, guitar tablature arrangement algorithms, and also reviews commercial guitar tablature transcription systems. Chapter 3 presents the design of the web-based guitar tablature transcription framework Robotaba and describes the implemented polyphonic transcription and guitar tablature arrangement algorithms. Chapter 4 presents the compiled ground-truth datasets and describes the methodology for evaluating the output of the implemented polyphonic transcription and guitar tablature arrangement algorithms. The results of this evaluation are presented and discussed in Chapter 5. The work is concluded in Chapter 6.

Chapter 2

Literature Review

The task of automatic guitar tablature transcription can be described as a function f that takes an audio signal x as input and outputs guitar tablature y. This function can be decomposed into two composite functions: a polyphonic transcription function g and a guitar tablature arrangement function h, such that $h(g(x)) = y$ (Figure 2.1). The polyphonic transcription function takes an audio signal as input and outputs a sequence of note events z. Each note event has a pitch, an onset time, and a duration. The guitar tablature arrangement function takes a sequence of note events as input and outputs a string and fret combination for each note in the input sequence.

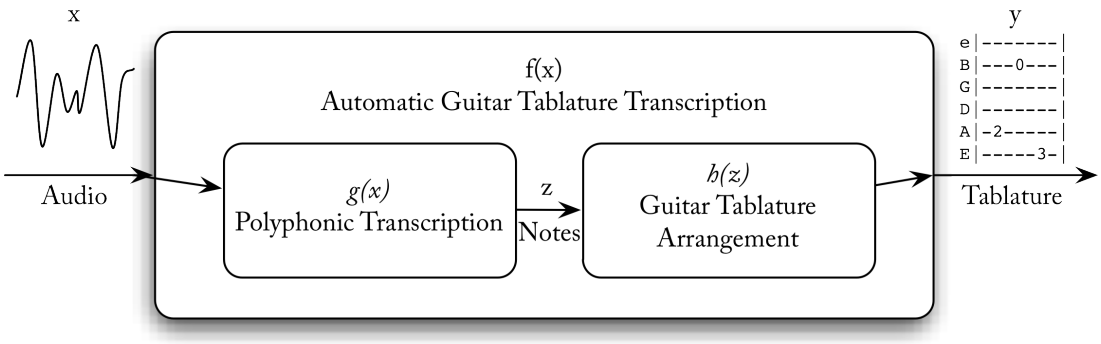


Fig. 2.1: Function decomposition of automatic guitar tablature transcription.

This chapter will provide an overview of algorithms that can be used to automatically transcribe tablature from a guitar recording. Section 2.1 reviews different approaches to the

problem of polyphonic transcription, followed by an examination of algorithms for guitar tablature arrangement in Section 2.2. Section 2.3 presents guitar-specific transcription algorithms and Section 2.4 introduces guitar tablature transcription systems that analyze data other than audio. Finally, Section 2.5 reviews commercial products that perform automatic transcriptions or aid humans in the process of manual transcription.

2.1 Polyphonic Transcription

Automatic music transcription refers to the process of extracting musical information from an audio signal in order to generate a symbolic musical score. The analysis of the input audio signal may be realtime or offline. In realtime, where the audio signal is input from a live performance of a musician, the analysis is causal—the transcription of note events at time t_0 only depends on the input audio signal $x(t) \quad \forall\ 0 \leq t \leq t_0$. Depending on the transcription algorithm being used, the offline analysis of an audio recording may be causal or non-causal since the transcription of note events at time t_0 may have access to the entire input audio signal $x(t) \quad \forall\ t \geq 0$.

Automatic music transcription can be divided into two categories: monophonic transcription and polyphonic transcription. Monophonic transcription algorithms are only capable of transcribing note events from input audio signals where one instrument plays a single note at a time. On the other hand, polyphonic transcription algorithms attempt to transcribe note events from input audio signals where several notes occur simultaneously. Polyphonic audio signals may result from one instrument playing multiple notes at the same time, or from multiple instruments (with similar or different timbres) that sound simultaneously. Though monophonic transcription can be considered a subproblem of polyphonic transcription, monophonic transcription algorithms use different analysis techniques, which exploit the fact that only one note occurs at a time, yielding simpler and more robust algorithms (Plumbley et al. 2002).

Polyphonic transcription algorithms output the pitch, onset time, and duration of notes occurring in an input audio signal. To estimate the pitch of a note, multiple fundamental frequency estimation algorithms are used. Fundamental frequency f_0 is defined as the lowest frequency of a periodic waveform and is the inverse of the fundamental period T_0 (Oppenheim et al. 1997, 17–8). Overtones f_k occur at integer multiples of the fundamental

frequency

$$f_k = kf_0 \quad \forall\, k \in \mathbb{N}^+. \tag{2.1}$$

Fundamental frequency is the physical phenomenon that corresponds to the perceptual phenomenon of pitch. To estimate the pitch of a note in Western music following an equal-tempered scale, fundamental frequency estimates are quantized to the frequency of the nearest pitch. For example, consider a note with a fundamental frequency estimate of $f_0 = 435.20$Hz. The fundamental frequency lies between the pitch G$^\sharp$4, having a fundamental frequency of 415.3Hz, and the pitch A4, having a fundamental frequency of 440Hz. In this case the note is assigned the pitch A4 because $|f_0 - 440| < |f_0 - 415.3|$.

The remaining temporal features of a note event, specifically the note onset time and duration, are estimated using an onset detection algorithm and an offset detection algorithm. In the literature, offset detection is alternatively referred to as *note tracking* since these algorithms track pitch estimates across analysis frames of the audio signal, from the note onset time until the pitch estimate can no longer be found in the audio signal. The accuracy of note offset time estimates are less important than onset time estimates because the former exhibit less perceptual importance than the latter (Costantini et al. 2009).

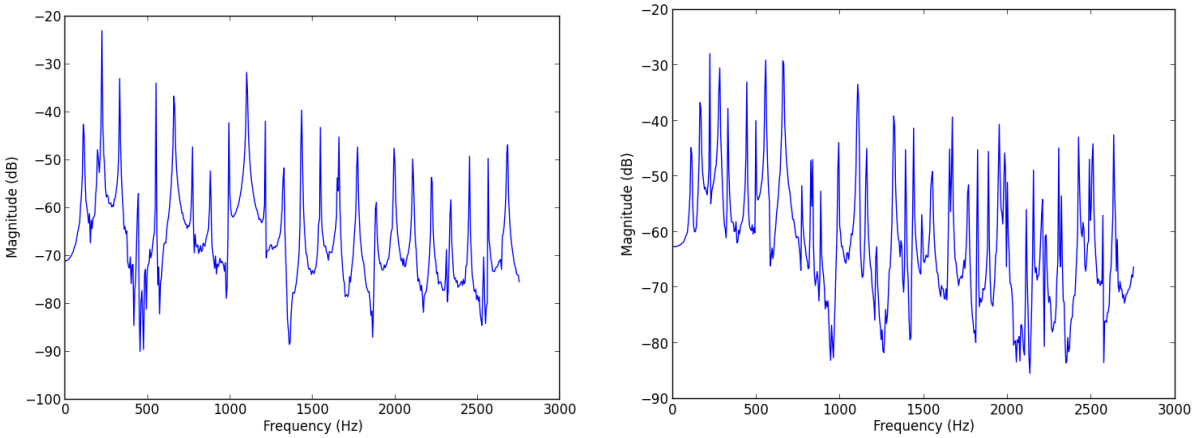
Although the transcription of monophonic musical passages is considered a solved problem (Klapuri 2004) as a result of accurate monophonic transcription systems (Slaney and Lyon 1990; Maher and Beauchamp 1994; Cheveigné and Kawahara 2002), the transcription of polyphonic music with no limitations on the number of instruments, the type of instruments, and the degree of polyphony[1] is still an open problem (Benetos et al. 2012). Detecting multiple fundamental frequencies in a polyphonic signal is a difficult task because it is ambiguous whether a peak in the magnitude spectrum at a particular frequency bin "is a fundamental or a harmonic, or both" (Fiss and Kwasinski 2011) due to coinciding harmonics. For example, let f_a and f_b be fundamental frequencies such that $f_a \neq f_b$. Harmonics of these fundamental frequencies overlap when

$$mf_a = nf_b \quad \forall\, m, n \in \mathbb{N}^+ \tag{2.2}$$

(Bonnet and Lefebvre 2003).

[1]The term *degree of polyphony* will be used to refer to the number of notes occurring simultaneously.

For a visual comparison of the complexity of fundamental frequency estimation versus multiple fundamental frequency estimation, consider the magnitude spectrum of a pluck of the open A string on a guitar in standard tuning displayed in Figure 2.2(a). The first local maximum is approximately $f_0 = 110$Hz and overtones (local or global maxima) are located at roughly integer multiples of the fundamental frequency. On the other hand, consider the magnitude spectrum displayed in Figure 2.2(b) of an A major chord comprised of five notes sounding simultaneously. The magnitude spectrum has no apparent structure because the frequency spectra of individual notes are overlapping.



(a) Frequency spectrum of the pluck of the open A string on a guitar in standard tuning.

(b) Frequency spectrum of the strum of an A major chord on a guitar.

Fig. 2.2: Comparison of the frequency spectrum of a pluck versus a strum on an acoustic guitar with steel strings.

Many music researchers are searching for a general solution to the problem of polyphonic transcription. Therefore, the majority of proposed polyphonic transcription systems are instrument and genre agnostic. These transcription systems aim to transcribe a wide range of harmonic musical instruments in a variety of playing styles (Benetos et al. 2012).

In an attempt to constrain the parameters of this difficult problem, more specialized transcription systems have been introduced that require audio input with specific musical properties or *a priori* knowledge of the musical passage being transcribed. The first polyphonic transcription system for duets yielded promising results by imposing constraints on the frequency range and timbre of the input instruments involved and the intervals

between simultaneously performed notes (Moorer 1975). In contrast to the more mature field of speech recognition where the majority of practical systems are language, gender, or speaker dependent (Huang et al. 2001), instrument and genre-specific polyphonic transcription systems are significantly less represented in the MIR literature in comparison to their general counterparts (Benetos et al. 2012). In favour of specialized polyphonic transcription systems, Martin (1996) commented that "the importance of a structured domain is that it allows the transcribing agent to exploit the structure, thereby reducing the difficulty of the task".

The music information retrieval evaluation exchange (MIREX)—an annual evaluation of state-of-the-art MIR algorithms using the same datasets and metrics—started evaluating polyphonic transcription algorithms in 2007.[2] Submitted polyphonic transcription algorithms are evaluated on two datasets. The first dataset consists of 30 audio recordings with a variety of musical instruments and degrees of polyphony. The second dataset consists of ten piano recordings. Many of the algorithms evaluated on the piano dataset are not necessarily piano-specific transcription systems (Benetos et al. 2012).

Since the scope of this thesis is on the transcription of polyphonic music arising from a single instrument, this section will provide an overview of polyphonic transcription algorithms (and surrounding schools of thought) that performed well in the MIREX evaluations on the piano dataset from 2007–2012. Table 2.1 presents the MIREX results of the reviewed transcription algorithms using the metrics of precision p (the ratio of correctly transcribed notes to the number of transcribed notes), recall r (the ratio of correctly transcribed notes to the number of ground-truth notes), and f-measure

$$f(p, r) = \frac{2pr}{p + r}, \tag{2.3}$$

which combines precision and recall into a single metric. The reported MIREX results consider the accuracy of note pitch and onset time but disregard note duration errors.

Although this section reviews polyphonic transcription algorithms that performed well on the MIREX piano dataset (Table 2.1), being stringed instruments, the piano and the guitar have similar properties. One similarity is that both instruments comply with the inharmonicity phenomenon, which causes harmonics in the upper frequency range to be

[2] http://www.music-ir.org/mirex/wiki

Table 2.1: Polyphonic transcription systems that performed well in the 2007–2012 MIREX on the piano dataset.

Polyphonic transcription system	Precision	Recall	f-measure
Zhou and Reiss (2008)	0.738	0.777	0.757
Ryynänen and Klapuri (2005)	0.720	0.669	0.694
Poliner and Ellis (2007)	0.672	0.630	0.650
Emiya et al. (2008)	0.649	0.639	0.643
Vincent et al. (2007)	0.591	0.651	0.620
Yeh and Roebel (2011)	0.504	0.793	0.616
Benetos and Dixon (2012)	0.627	0.594	0.610
Yeh and Roebel (2010)	0.497	0.785	0.609
Benetos and Dixon (2011a)	0.663	0.532	0.590
Nakano et al. (2009)	0.541	0.539	0.540
Dessein et al. (2010)	0.425	0.738	0.539
Lee et al. (2011)	0.531	0.525	0.528
Lee et al. (2010)	0.575	0.480	0.523
Chang et al. (2008)	0.345	0.680	0.458
Raczyński and Sagayama (2009)	0.689	0.246	0.363

shifted upwards in frequency according to the formula

$$f_k = k f_0 \sqrt{1 + \beta(k^2 - 1)}, \tag{2.4}$$

where f_0 is the fundamental frequency, k is the harmonic index, and β is the inharmonicity factor (Fletcher and Rossing 1998).

Polyphonic transcription is a difficult problem with a large number of variables. There are various different approaches to the problem, many of which rely on multiple techniques to perform a transcription (Klapuri 2004). Therefore, it is difficult to categorize polyphonic transcription systems according to the methodology used. Nevertheless, the following sections aim to provide an overview of the main approaches to the problem.

2.1.1 Human Audition Modelling

As a motivation for polyphonic transcription algorithms modelled after the human audition system, Anssi Klapuri, an expert in the field of polyphonic music transcription, states

that "in music transcription, ... the problem is really not in finding fast computers but in discovering the mechanisms and principles that humans use when listening to music. Modelling perception is difficult because the world in which we live is complex and because the human brain is complex" (Klapuri 2004). A large body of multi-disciplinary research follows this school of thought, developing polyphonic transcription algorithms that replicate how it is believed that humans perform this complex task.

It is possible to pose the problem of polyphonic transcription as one of sound source separation, a complex cognitive function that humans perform with ease. Auditory scene analysis is the process by which the human auditory system separates a mixture of sounds that are interwoven in both the time and frequency domain into their constituent sources (Bregman 1990). This natural phenomenon is commonly referred to as the *cocktail party effect* (Cherry 1953), which addresses the ability of humans to focus their attention on one speaker amongst various other speakers and superfluous noise. In the context of polyphonic transcription, the deconstruction of a polyphonic signal y into a set of M monophonic signals x_i that represent the audio signal of each note in a chord

$$y(t) = \sum_{i=1}^{M} x_i(t), \tag{2.5}$$

significantly reduces the complexity of the problem. The signal of each note can then be transcribed by established monophonic transcription algorithms and combined to form the complete polyphonic transcription.

Forming a computational model of the auditory scene analysis process, Kashino and Tanaka (1993) proposed a system for the polyphonic transcription of multiple instruments. Using a bottom-up approach, frequency components present in the Fourier decomposition of a monaural audio signal were clustered into individual note hypotheses according to human cues for auditory source separation, such as inharmonicity and the proximity of harmonics in the frequency domain. Additionally, timbre recognition was performed to attribute a musical instrument to each transcribed note. The system achieved 90% accurate transcriptions on synthesized musical instrument digital interface (MIDI) files of piano and flute limited to two and three degrees of polyphony.

Similarly, Kameoka et al. (2007) proposed an approach to polyphonic transcription called harmonic temporal structured clustering, which deconstructs a polyphonic audio

signal into individual source signals by clustering frequency components. This decomposition is represented as a Gaussian mixture model (GMM) $p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$, where the number of component distributions K must be specified *a priori* and the unknown parameters of the distribution (mixing coefficients π_k, means μ_k, and covariances Σ_k) are estimated using the expectation maximization algorithm. Nakano et al. (2009) extended this algorithm to automatically estimate the degree of polyphony K present in the input signal. The results of this algorithm in the MIREX 2009 evaluation are presented in Table 2.1.

Alternatively, the decomposition of a polyphonic signal into source components as in Equation 2.5 can be further deconstructed into a summation of sinusoids for each note

$$y(t) = \sum_{i=1}^{M} \sum_{j=1}^{N} [\alpha_{i,j} \cos(j f_0^{(i)} t) + \beta_{i,j} \sin(j f_0^{(i)} t)] + \eta(t), \tag{2.6}$$

where the degree of polyphony M, the number of sinusoidal components N, the fundamental frequency $f_0^{(i)}$ of each note, and the distribution of the residual noise η is unknown. Statistical signal processing techniques can be used to estimate these unknown parameters and perform a transcription of the input signal y. Davy and Godsill (2002) staged the problem in a Bayesian statistical framework, where prior distributions for the unknown parameters are used to estimate values of these parameters that best explain the observation signal y. The system is reported to perform well on signals with up to three degrees of polyphony; however, the authors note that searching the large parameter space is on the verge of being computationally intractable.

Another stream of research seeks to model the human auditory and periphery systems that contribute to the perception of pitch, a psychoacoustical attribute that humans try to assign to almost all incoming acoustical signals (Meddis and Hewitt 1991). Meddis and O'Mard (1997) introduced the unitary model of pitch perception, where the function of the middle and inner ear is modelled by a set of bandpass filters. The output of each filter is compressed, half-wave rectified, and low-pass filtered to obtain the amplitude envelope of the signal. To estimate the perceived pitch, an autocorrelation function then looks for periodicities in the amplitude envelope and sums the result of each frequency band, mimicking the function of peripheral audio processing systems in the human brain.

Klapuri (2005) modified the proposed unitary model of pitch perception to use a technique called *harmonic selection* instead of an autocorrelation function in order to extract more reliable fundamental frequency estimates. This fundamental frequency estimation algorithm was later extended to include a hidden Markov model (HMM) for note event modelling and a musicological model to govern the transitions between note events (Ryynänen and Klapuri 2005). This polyphonic transcription system was submitted to MIREX in both 2007 and 2008 and received the results displayed in Table 2.1.

2.1.2 Salience Methods

Multiple fundamental frequency estimation algorithms that apply transformations to the input audio signal in order to emphasize the underlying fundamental frequencies will be referred to as salience methods. The first example of a salience method is the fundamental frequency salience function proposed by Klapuri (2006). The salience of a f_0 candidate is calculated as the weighted sum of the magnitude of its harmonics. In mathematical terms,

$$s(f) = \sum_{m=1}^{M} \alpha(f, m)|X(mf_s f)|, \tag{2.7}$$

where M is the number of harmonics, α is an empirically determined function that produces the weight of a harmonic, $X(f)$ is the Fourier transform of the input signal that is spectrally whitened to suppress timbral information, and f_s is the sampling rate. In theory, there exists peaks in the magnitude spectrum at integer multiples of a fundamental frequency (Equation 2.1). Therefore, the maxima of the salience function should correspond to the fundamental frequencies of notes present in the audio signal. Iterative and joint estimation algorithms that search for the maxima of this salience function are described in Section 2.1.3.

Instead of using conventional time-frequency signal representations such as the short-time Fourier transform (STFT) for polyphonic transcription, Zhou and Mattavelli (2007) developed a time-frequency analysis tool called the resonator time-frequency image (RTFI), which is implemented by a bank of first-order complex resonator filters. The RTFI can support various frequency resolutions, such as uniform or constant-Q (logarithmic) resolutions. Using the RTFI, Zhou and Reiss (2008) proposed a polyphonic transcription system that

not only received the best results to date in the MIREX evaluations on the piano dataset (see Table 2.1), but also performs in realtime.

The proposed transcription system first calculates the RTFI of the input audio signal, which is used for onset detection and multiple fundamental frequency estimation. Before processing, the average of every analysis frame in the RTFI is calculated to produce the RTFI average energy spectrum. The onset detection algorithm uses a threshold technique that considers peaks in the difference energy spectrum, which is calculated by smoothing the energy spectrum and subtracting a time-lagged version of itself to accentuate signal transients. The input audio signal is then segmented according to the detected note onsets and the multiple fundamental frequency estimation is performed in each segment.

The multiple fundamental frequency estimation algorithm begins by diverging into two paths: the extraction of harmonic components by transforming the RTFI into a relative energy spectrum, and the estimation of fundamental frequency candidates by transforming the RTFI into a pitch energy spectrum. A detailed explanation of these signal transformations can be found in Zhou et al. (2009). Information about the harmonic components and fundamental frequency candidates are combined using a rule-based approach to filter the f_0 candidates into a smaller set. The resulting set of f_0 candidates is further pruned by applying the spectral smoothing principle, which attempts to discern if harmonic peaks with large magnitudes are the result of another underlying fundamental frequency. Following the success of this time-frequency analysis tool, the RTFI has been used in other polyphonic transcription algorithms (Benetos and Dixon 2011b).

2.1.3 Iterative and Joint Estimation

For the estimation of multiple fundamental frequencies from the frequency domain of an input audio signal, two predominant methods exist in the literature: iterative and joint f_0 estimation. Iterative f_0 estimation algorithms first estimate a predominant fundamental frequency and proceed to estimate the spectrum of the fundamental frequency and overtones. The estimated spectrum is subtracted from the original spectrum and the process reiterates with the estimation of another fundamental frequency from the residual frequency spectrum. Conversely, joint f_0 estimation algorithms choose from a set of fundamental frequency candidates that, together, best describe the frequency spectrum of the input audio signal.

Klapuri (2006) proposed an iterative and a joint f_0 estimation algorithm that search for maxima in the salience function described in the previous section (Equation 2.7). In the case of iterative estimation, the fundamental frequency with the maximum salience is selected. Its spectrum is estimated, subtracted from the original spectrum, and this process is repeated on the residual spectrum. In the case of joint estimation, f_0 candidates are chosen by selecting a set of local maxima from the salience function. A search algorithm chooses a subset of fundamental frequencies according to a metric that measures the fit of an individual f_0 candidate in the context of others in the set. Since no algorithm for onset detection was presented, the proposed algorithms only provide frame-by-frame f_0 estimates. After evaluation, results showed that the algorithm performed equivalently but more efficiently than the author's previously presented f_0 estimation algorithm that modelled the human auditory system (Klapuri 2005).

Emiya et al. (2007) proposed a joint multiple fundamental frequency estimation algorithm for inharmonic instruments such as the piano and guitar. Maximum likelihood estimation was used to estimate the parameters of a weighted sum of sinusoids plus noise model similar to Equation 2.6, except that the frequency of overtones was calculated by Equation 2.4 instead of calculating multiples of the fundamental frequency as in Equation 2.1. The output of the estimation algorithm is the set of notes with fundamental frequencies that jointly maximize the likelihood function. Emiya et al. (2008) subsequently embedded this joint multiple fundamental frequency estimation algorithm in a polyphonic transcription system that first performs onset detection and subsequently tracks note events using an HMM to determine note duration. The proposed transcription system was evaluated in MIREX 2008 and received the results presented in Table 2.1.

In his doctoral dissertation, Yeh (2008) proposed a frame-by-frame joint f_0 estimation algorithm that uses the weighted sinusoid plus noise model presented in Equation 2.6. The algorithm first performs a fast Fourier transform (FFT) on each analysis frame of the input audio signal to obtain the frequency spectrum of the signal. The residual noise signal $\eta(t)$ is first estimated to distinguish harmonics of the fundamental frequencies from extraneous noise in the frequency domain. A set of f_0 candidates are jointly estimated from the frequency spectrum through the use of a scoring function that considers the physical properties of harmonic sounds (Yeh et al. 2005). To estimate the number of concurrent notes, the scoring function is applied to combinations of f_0 candidates at various degrees of polyphony. A summary of this f_0 estimation algorithm can be found in (Yeh et al. 2010).

Chang et al. (2008) embedded the frame-by-frame joint f_0 estimation algorithm proposed by Yeh (2008) in a polyphonic transcription system that uses an HMM to track f_0 candidates across analysis frames. In contrast with conventional HMM note tracking systems, which include attack, sustain, release, and silence states, the proposed note tracking system only includes attack and sustain states. The proposed algorithm does not perform onset detection for the identification of the start of a note event. Rather, note onsets are identified by a new pitch estimate occurring in the analysis of the audio signal. The polyphonic transcription system was evaluated in MIREX 2009, receiving the results presented in Table 2.1. The polyphonic transcription system was later improved by modifying the distribution of the residual noise model in the f_0 estimation algorithm (Yeh and Roebel 2010). The resulting system was submitted to MIREX 2010, receiving the results in Table 2.1. Modifications to the f_0 candidate extraction algorithm showed further improvements in both precision and accuracy (see Table 2.1) in the MIREX 2011 evaluation (Yeh and Roebel 2011).

2.1.4 Machine Learning Approaches

The transcription of a mixture of note signals present in an audio recording can be framed as a machine learning problem. Machine learning algorithms use a set of input observations to train a model that attempts to explain, predict, or classify new observations. In the case of polyphonic transcription, the input observations are gathered by extracting features from frames of an audio signal. For model training and validation, supervised machine learning algorithms for polyphonic transcription require a ground-truth (correctly labelled) dataset, which consists of a set of audio recordings annotated with the correct note events occurring in each recording. There are a variety of different machine learning algorithms, many of which have been applied to the problem of polyphonic transcription.

Non-negative Matrix Factorization

Recently, many polyphonic transcription systems have been proposed that use and expand upon non-negative matrix factorization (NMF) algorithms. NMF is a factorization method that attempts to decompose a matrix $X \in \mathbb{R}_{\geq 0}^{M \times N}$ into the product of two matrices $W \in \mathbb{R}_{\geq 0}^{M \times L}$ and $H \in \mathbb{R}_{\geq 0}^{L \times N}$ such that

$$X \approx WH. \tag{2.8}$$

To attain the decomposition, the matrices W and H are alternately updated in an iterative fashion according to the gradient of an error metric (Wang and Zhang 2012). In the context of polyphonic transcription, the columns in W contain the spectral templates for each pitch and H encodes the temporal activity of each pitch over the course of the audio signal.

Since Smaragdis and Brown (2003) first applied NMFs to the problem of polyphonic transcription, many extensions have been proposed (Vincent et al. 2007; Raczyński and Sagayama 2009; Dessein et al. 2010) that explore different constraints and learning methods. The results of these algorithms in the MIREX evaluation on the piano dataset can be found in Table 2.1.

Sparse representations with a similar structure to an NMF have also been proposed for polyphonic transcription (Lee et al. 2010; Lee et al. 2011). These algorithms form a dictionary, similar to W in the NMF, of the magnitude spectra from recordings of notes played on an instrument. Instead of the matrix H containing the temporal information of note events, it instead contains weights for each note template. The task of multiple fundamental frequency estimation is then accomplished by searching for a weighted sum of note templates to match the frequency spectrum of an input audio signal. Similar to previously mentioned approaches, an HMM is then used for the purpose of note tracking. These algorithms were evaluated in MIREX, receiving the results presented in Table 2.1. Although these algorithms formed the note dictionary from piano recordings, any instrument that generates harmonic waveforms could be used. To incorporate multiple instruments, Benetos and Dixon (2011a, 2012) proposed the use of multiple templates for each pitch and for each musical instrument considered. Using only pitch templates from a piano, these polyphonic transcription systems were evaluated in MIREX on the piano dataset, receiving the results displayed in Table 2.1.

Support Vector Machines

Pattern recognition algorithms that are trained on spectral features can be used to classify the frequency components of individual notes within the frequency spectrum of a mixture of notes. Poliner and Ellis (2005, 2006, 2007) proposed the use of 87 binary classification support vector machines (SVMs) to detect the presence of a note in each analysis frame of the STFT of an input audio signal. Through a supervised training process the SVM classifier attempts to create separating hyperplanes that maximize the distance between

training points. These hyperplanes represent the optimal (given the provided training data) decision boundary between the presence or absence of a specific note. After receiving the note candidates in each analysis frame, an HMM was used for the purposes of note tracking. The proposed algorithm (Poliner and Ellis 2007) was evaluated in MIREX 2007, receiving the results presented in Table 2.1.

Neural Networks

Neural networks have also been applied to the problem of polyphonic transcription, performing both pitch estimation and note tracking (Marolt 2000). Marolt (2001, 2004) developed the *SONIC* polyphonic transcription system that utilizes multiple neural networks to automatically transcribe piano music. First, the input audio signal is transformed into a time-frequency representation using an auditory model that mimics the functionality of the human cochlea. Each output channel of the auditory model is input into an array of 88 networks that attempt to determine the presence of the pitches A0–C8 in the input audio signal. The network for each pitch consists of a set of adaptive oscillators with centre frequencies at multiples of the fundamental frequency, which attempt to synchronize with the harmonically related partials of a musical note. For onset detection, a neural network monitors significant changes in the amplitude envelope of the audio signal. The output of this neural network is a series of impulses, which represent the presence of note onsets. The transcription system was evaluated on three synthesized and three real classical piano recordings, obtaining 92% average accuracy on the former dataset and 81% average accuracy on the latter dataset. Although the neural networks were trained on piano recordings, the transcription system could be adapted for the guitar by training on guitar recordings and limiting the maximum degree of polyphony of the output music score to six instead of ten.

Hidden Markov Models

Hidden Markov models are probabilistic models that attempt to explain a finite-state machine that produces an observation at each state transition. The model defines a matrix of probabilities that describes the policy of transitioning between states. The model also defines an emission distribution—a probability mass or density function that describes the probability of producing a specific observation given the current state. Given a sequence of

observations, an HMM attempts to uncover the underlying state sequence that produced the observations. Raphael (2002) used an HMM to transcribe the notes present in a polyphonic piano recording. A number of features were extracted from frames of the audio recording and presented as observations to the HMM. Each state of the HMM represents a combination of notes and a label that describes the temporal evolution of the chord i.e., the attack, sustain, or rest. With a state space so large, searching for an optimal state sequence becomes computationally intractable. In response to this issue the maximum degree of polyphony was limited to four, only the pitches C2–F6 were considered, and heuristics were used to prune the search space. The algorithm was evaluated on recordings of movements from Mozart piano sonatas and received 61% transcription accuracy.

2.1.5 Blackboard Systems

The blackboard problem-solving model was conceived in the field of artificial intelligence as an alternative approach to the static and sequential processing of data (Newell 1962). The term "blackboard problem solving" refers to a number of experts in different domains who work together to solve a problem on a physical blackboard. As the solution evolves, each expert adds or modifies information on the blackboard in turn or when they have a significant contribution. Realizing this metaphor, the implementation of a blackboard system requires a set of independent knowledge sources (experts) to interact through a global database (blackboard) in a manner governed by a scheduler or monitor (Figure 2.3). Blackboard systems aim to flexibly integrate hypotheses about the problem from different knowledge sources. Although no blackboard polyphonic transcription systems placed among the top algorithms evaluated on the piano dataset in the MIREX, it is still an important school of thought and will be reviewed in this thesis.

Using the blackboard problem-solving model, Martin (1996) proposed a system for the polyphonic transcription of piano music. Input to the blackboard system was formed of "tracks"—local maxima in the magnitude of the STFT of the input audio signal. The blackboard database was arranged hierarchically: tracks were grouped into partials, partials into notes, notes into intervals, and intervals into chords. The system consisted of thirteen knowledge sources with expertise in areas of acoustical physics, music theory, and garbage collection (handling incorrect or competing hypotheses). A sequential scheduler organized contributions of the knowledge sources. The proposed transcription system was capable of

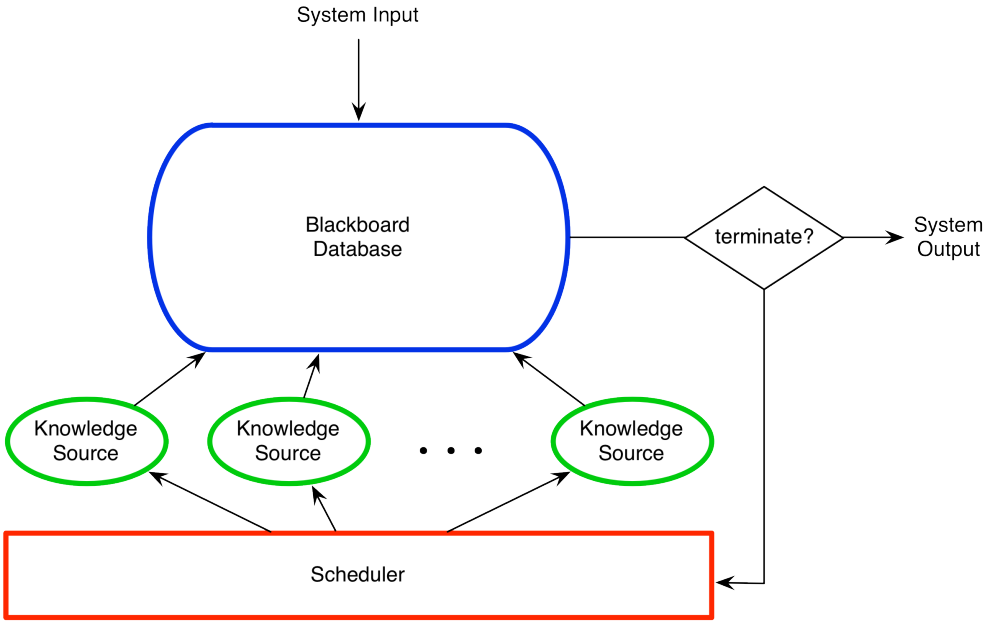


Fig. 2.3: Components and workflow of a blackboard system.

transcribing synthesized piano music where the interval between simultaneous notes was not an octave and all notes were between the pitches B3 and A5.

Martin's blackboard transcription system was later extended by Bello and Sandler (2000) to include a neural network chord recognition component that influenced the selection of note hypotheses on the blackboard. More information on the history of the blackboard problem-solving model and a detailed review of its application to polyphonic music transcription can be found in Bello (2003).

2.2 Guitar Tablature Arrangement

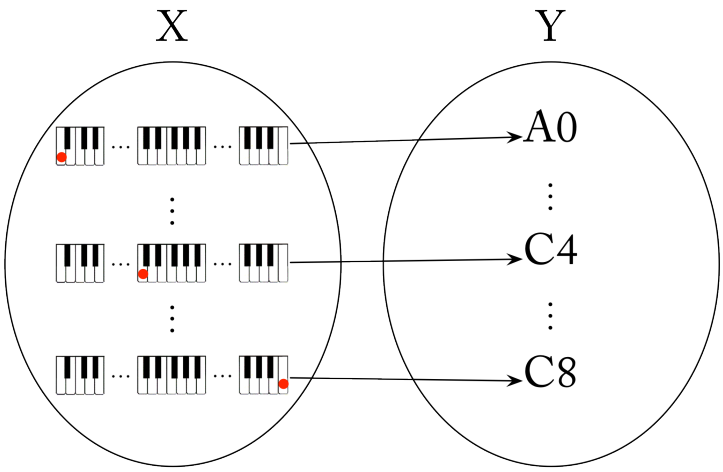
Unlike the piano which has a one-to-one correspondence between the set of physical keys and the set of possible pitches, the guitar can produce the same note in several ways, adding more ambiguity to the transcription process (Fiss and Kwasinski 2011). In mathematical terms, the mapping between the set of physical keys X and the set of possible pitches Y on a piano is a bijective function, meaning that every element of the codomain maps to

exactly one element of the domain. This bijective function is displayed in Figure 2.4(a). The mapping between the set of string and fret combinations X and the set of possible pitches Y on a guitar is a surjective function, meaning that every element of the codomain is mapped to at least one element of the domain. This surjective function is displayed in Figure 2.4(b).

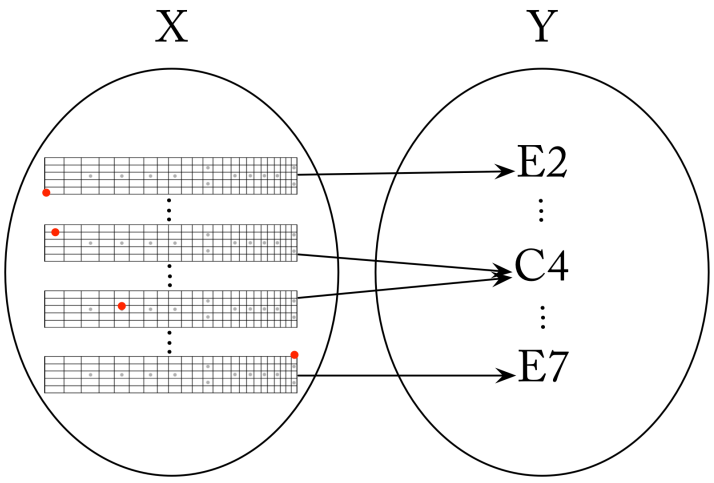
Given a sequence of note events as input, guitar tablature arrangement algorithms assign a string and fret combination to each note event according to criterion that minimizes the performance difficulty of the tablature. By studying the left-hand movements of professional classical guitar players, Heijink and Meulenbroek (2002) hypothesized that guitarists have a disposition toward instrument fingering positions that are biomechanically easy to perform. From this study emerged three complexity factors that contribute to the performance difficulty of a tablature arrangement: the position of the left hand on the guitar neck, the need to reposition the left hand within a stream of notes, and finger span. Experiments indicated that subjects favoured hand positions near the beginning of the guitar fretboard near the nut. Moreover, the subjects avoided composing arrangements that required extensive hand repositioning and large finger spans. Apart from biomechanical constraints, the authors speculated other important criteria for tablature arrangement such as cognitive and musical rules that take into consideration musical context and enforce auditory properties such as timbral characteristics of the produced sound.

Guitar tablature arrangement can be perceived as a traditional search problem, where there are many candidate string and fret combinations for each note in a musical work and the goal is to find an optimal arrangement that maximizes a quantitative metric defining a "good tab". A "good tab" refers to tablature that is generally easy to perform or has a difficulty level that is tailored to the aptness of the performer (Sawayama et al. 2006). The search space can quickly become computationally intractable since a standard 24-fret electric guitar is capable of producing the same pitch in up to six different ways (Figure 2.5), yielding an upper bound of 6^n possible tablature arrangements for a sequence of n notes. In the polyphonic case, the search space rapidly enlarges as more chords are considered. A sequence of n chords, each composed of six notes that may be played in up to six different ways, yields an upper bound of $(6^6)^n$ possible tablature arrangements.

Guitar tablature arrangement algorithms may also produce extra performance information such as note ornamentations (Miura et al. 2004) or the explicit instrument fingering for each note. The instrument fingering of a note explicitly states which finger on the left



(a) Bijective function between the domain of physical piano keys X and the codomain of possible pitches Y.



(b) Surjective function between the domain of string and fret combinations on a guitar X and the codomain of possible pitches Y.

Fig. 2.4: Function classifications of the mappings from instrument operation to pitch for the piano and guitar.

hand is used to depress the fret on a given string. For the purposes of guitar tablature

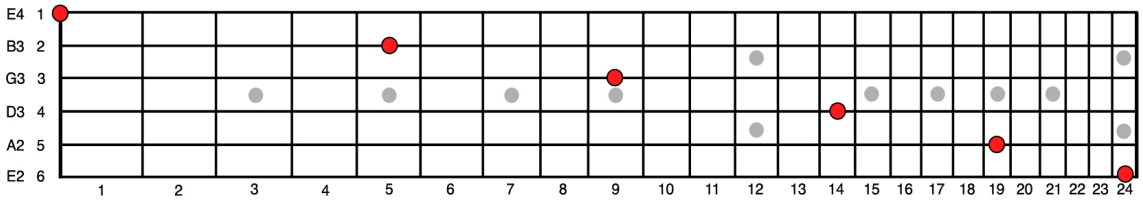


Fig. 2.5: Six different string and fret combinations that produce the pitch E4 on a 24-fret guitar in standard tuning.

arrangement, which only displays the string and fret combination required to perform a note, the inclusion of fingering information is unnecessary and can therefore be excluded.

Many methods have been proposed to search for an optimal tablature arrangement, such as traditional graph search algorithms (Section 2.2.1), constraint satisfaction (Section 2.2.2), neural networks (Section 2.2.3), and genetic algorithms (Section 2.2.4).

2.2.1 Graph Algorithms

In mathematics, a graph $G = (V, E)$ is composed of a set of vertices V that are connected by a set of edges E. In an undirected graph, each element of the set E is a set of vertices $\{u, v\}$ with cardinality two such that $u, v \in V$. In a directed graph, each element of the set E is an ordered pair of vertices (u, v) such that $u, v \in V$. In a weighted graph, each element of the set E is a tuple $(\{u, v\}, w)$ containing a pair of vertices and a weight $w \in \mathbb{R}$ associated with the edge. The weight associated with an edge typically represents the cost of traversing the edge; however, the interpretation of the weight is largely domain dependent.

In the context of guitar tablature arrangement, each vertex in the graph is associated with a candidate string and fret combination that produces the pitch of a note present in the input music score. The weight of an edge represents the "cost" associated with the transition between two finger positions. Following the study of professional classical guitarists by Heijink and Meulenbroek (2002), this transition cost should ideally reflect the biomechanical ease of the transition as well as the conformance with other cognitive and musical rules. Figure 2.6 illustrates a weighted graph of candidate string and fret combinations for a sequence of 3 notes: A4, E4, and C5. In this simple monophonic example there are already $5 \cdot 6 \cdot 4 = 120$ possible guitar tablature arrangements.

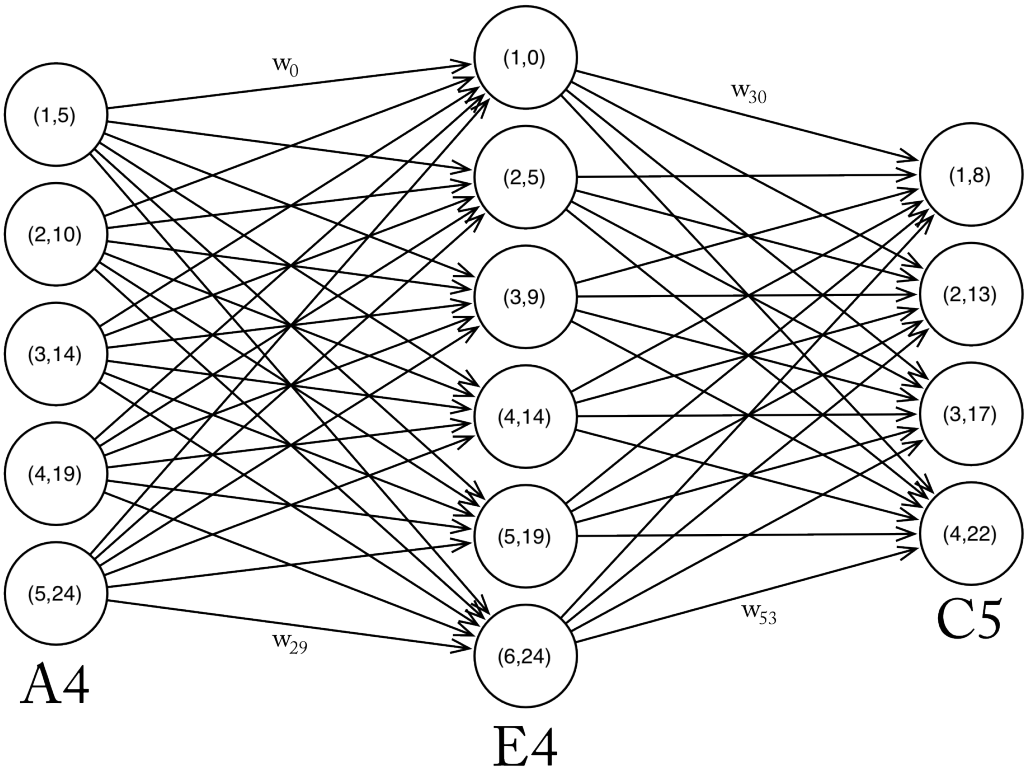


Fig. 2.6: A directed acyclic weighted graph of candidate string and fret combinations for a sequence of 3 notes: A4, E4, and C5. Some edge weights have been omitted for space purposes.

Many algorithms have been proposed to find a path through a weighted graph that incurs the minimal cost (Cormen et al. 2009). Sayegh (1989) used the Viterbi algorithm to search for an optimal path through a weighted graph of candidate string, fret, and finger combinations for notes in a monophonic musical passage. The Viterbi algorithm operates on the principle that if a vertex is part of a minimum cost path through the graph, any subset of this path is also an optimal path. If it is not optimal, then there exists an alternate intermediate path that has a lower aggregate cost and thus produces a better overall path.[3] The proposed arrangement algorithm uses a simple cost function that assigns a weight to each edge in the graph by penalizing transitions that require a change in hand position or

[3] An established application of the Viterbi algorithm is in HMMs, whereby an optimal state sequence is decoded from a sequence of observations (Rabiner 1989).

a change in string. As an extension to this simple cost function, a learning algorithm was proposed to automatically estimate the edge weights from a set of training examples. No formal evaluation of the proposed algorithm was performed.

Extending the optimal fingering estimation algorithm proposed by Sayegh (1989), Radicioni et al. (2004) advocated that the segmentation of a piece of music into phrases is an important cognitive process that affects tablature arrangement. Taking this into consideration, a variety of scores were manually segmented into musical phrases. A graph search algorithm then identified the optimal fingering positions within and between musical phrases according to a cost function that considered horizontal and vertical movement of the hand along the guitar fretboard. Horizontal movement was perceived to increase the complexity of the arrangement more than vertical movement. The proposed algorithm was later extended to accept polyphonic musical passages (Radicioni and Lombardo 2005b) by assessing the difficulty of fingering within a chord. The fingering estimation algorithm was evaluated by comparing the output of the algorithm on segments of classical guitar sonata scores to that of a human expert. Results of the experiments showed similar fingering positions to that of a professional guitarist.

Radisavljevic and Driessen (2004) proposed the use of a dynamic programming (DP) algorithm to search the graph of candidate string, fret, and finger combinations for notes in a polyphonic musical passage. DP algorithms use recursion to decompose the problem into smaller sub-problems, the solutions of which are combined to form the solution to the original problem. In the context of searching a graph, the optimal path through the graph is found by recursively finding the optimal path within a smaller search space. Static and transition cost functions were proposed, which measured the biomechanical difficulty of the hand position required to form a chord and the difficulty of transitioning between fingering positions, respectively. Many features, such as the average fret location and the number of frets between fingers, contributed to each cost function. Each feature was weighted to determine the relative importance of each criterion. Instead of tuning the feature weights by hand, the authors proposed a technique called "path difference learning". This technique uses gradient descent with respect to the feature weights to minimize the difference between the optimal path output by the DP algorithm and a set of training examples acquired from published tablature. The algorithm was trained using seven selected excerpts from classical guitar scores, receiving 97% accuracy when evaluating on the same dataset.

2.2.2 Constraint Satisfaction Approach

The constraint satisfaction problem consists of finding a set or range of values that a variable can assume without violating a set of constraints. Applying this problem-solving framework to the tablature arrangement of individual guitar chords, Radicioni and Lombardo (2005a) proposed that a variable represents a note within a chord and the domain of each variable is a string, fret, and left-hand finger combination. A set of constraints are imposed on the variables to restrict the possible combinations of fingering positions. These constraints enforce that a string can only play one note at a time, that fingers further along the fretboard depress higher fret numbers, and that a maximum finger span is not exceeded. A depth-first search strategy with backtracking is used to combine fingering positions and return a set of solutions that satisfy the imposed constraints. In the case that there are multiple solutions, the solutions are ranked based on the biomechanical ease of performance. To evaluate the proposed algorithm, human experts wrote three possible chord fingerings for 34 chords, ranked by preference. The constraint satisfaction algorithm produced chord fingerings that agreed 67% of the time with the preferred chord fingerings of a human expert. The algorithm achieved 97% accuracy in comparison to human experts when only considering string and fret combinations. Although the accuracy is quite high, it should be emphasized that the proposed algorithm only generates string and fret combinations for notes in a single chord and does not consider transitions between notes or chords, although this feature is proposed in the future work.

2.2.3 Neural Network Arrangements

Neural networks have also been used to generate guitar tablature that more closely resembles human arrangements. Tuohy and Potter (2006a) proposed the use of a three-layer neural network that sequentially calculates string and fret combinations for each note in an input music score. A potential problem with processing one note at a time is that important contextual information is lost, which adversely affects the tablature arrangement. To remedy this, the authors included contextual information of surrounding notes as input features to the neural network. A local search algorithm then passes over the tablature generated by the neural network and for each note determines if the arrangement can be improved by using a different string and fret combination according to the fitness function used in the genetic algorithm proposed by Tuohy and Potter (2006b). To train the network,

a dataset was constructed from an online repository of human-arranged guitar tablature. Evaluating the algorithm on the training dataset, the output tablature was 94% congruent with the published tablature.

2.2.4 Genetic Algorithms

The search for a guitar tablature arrangement that minimizes performance difficulty may be accomplished by a genetic algorithm (GA). As discussed at the beginning of Section 2.2, the search space for the guitar tablature arrangement problem rapidly expands as more chords and notes are added to the input music score. GAs are particularly suited for exploring large search spaces to find adequate solutions when exhaustive search algorithms become infeasible due to computational intractability (Tuohy and Potter 2005). This section will describe the function of GAs in detail and explain how they are employed to generate guitar tablature arrangements.

A genetic algorithm (GA) is a stochastic optimization technique that aims to replicate the process of natural evolution. A GA iteratively refines a set of possible solutions—referred to as the *population*—by assessing the strength of each individual (chromosome) in the population. Each chromosome consists of a string of genes that define the individual. The assessment of each individual is performed by a fitness function that evaluates an individual in the population against a set of criteria defining an optimal individual. The optimization process operates for a predetermined number of iterations (generations) or until a termination condition is met e.g., an individual has been found that is "fit" enough for the purposes of the application. At the heart of the optimization (evolutionary) process is the concept of natural selection.

Natural selection is the process of selecting individuals from the current population that will contribute to the successor population. The methodology for selection is a critical factor in determining the fitness of the successor population. In the field of artificial intelligence, specifically the underlying field of reinforcement learning, an important trade-off exists between exploitation and exploration (Sutton and Barto 1998). Exploitation involves the recurring performance of an action that is known to yield a high reward, whereas exploration involves the performance of an action that yields a reward with an unknown distribution that may have a higher mean reward than the current exploitative action. In the context of GAs, an exploitative selection process would involve always mating the most fit individuals

in a population, whereas a selection process that sporadically chooses other individuals for reproduction may result in a more fit individual. Taking the balance of exploration versus exploitation into consideration, GAs typically use a stochastic selection process where the probability distribution of selecting an individual for reproduction is influenced by the individual's fitness. Consequently, fitter individuals are more likely to reproduce but are not always the ones selected for reproduction. Another function to ensure exploration is gene mutation, which modifies a random gene of an individual with low probability. Figure 2.7 provides an overview of the evolutionary process of a GA.

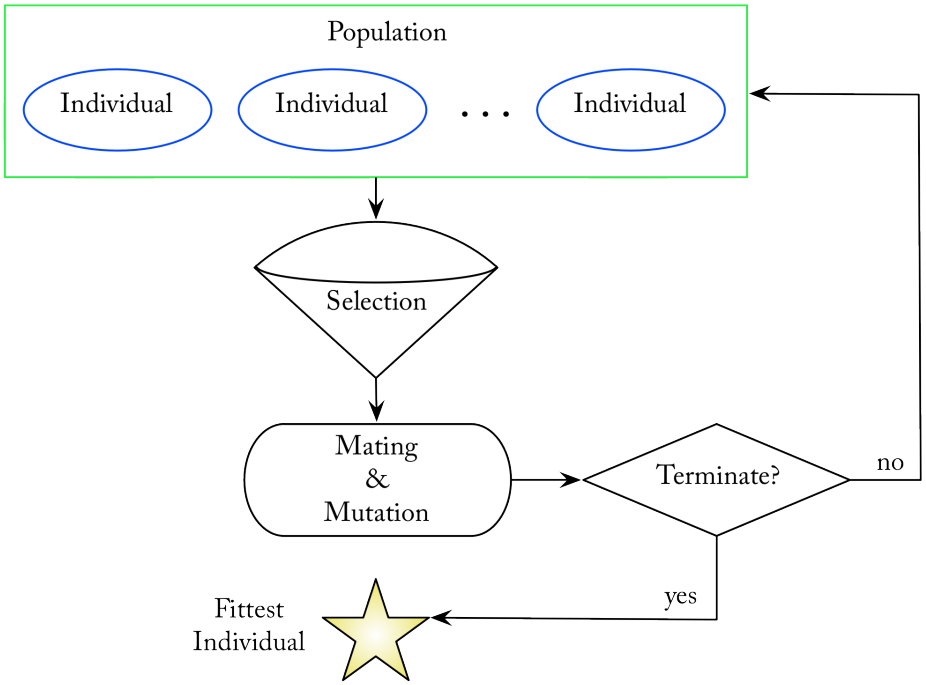


Fig. 2.7: An overview of the evolutionary process af a genetic algorithm.

In a GA, the transfer of genetic material from parents to offspring is similar to the natural phenomenon. Natural reproduction follows the principle that 50% of the genetic makeup of each parent is transferred to the offspring. In other words, the genetic makeup of the child has equal contributions from each parent. The mating function in a GA may enforce this, or allow variable genetic contributions from each parent. This is accomplished by randomly choosing a crossover point in the chromosome and "twisting" the genes about this locus. As an example, the mating algorithm for individuals with a gene sequence of

length five is illustrated in Figure 2.8. The mating function might also include a variable number of crossover points.

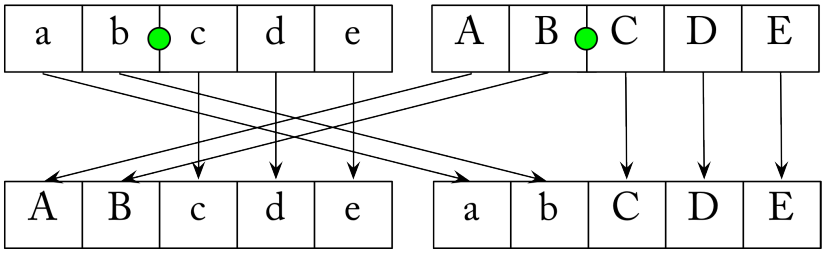


Fig. 2.8: Mating of two parents with one crossover point in the middle of the chromosome.

To apply a GA to the problem of guitar tablature arrangement, each component of the GA must first be defined. A gene represents a note or chord on the guitar and contains the string and fret combinations required to perform the note or chord. A chromosome is a sequence of genes that represents a candidate tablature arrangement. The population consists of many candidate tablature arrangements, which undergoes iterative mating and mutation to produce a more evolved population. An example of two tablatures mating is presented in Figure 2.9. After a certain number of generations, the tablature arrangement with the highest fitness is selected. Similar to the algorithms previously mentioned in this section, the fitness function of the GA typically evaluates the biomechanical ease of performing and transitioning between notes or chords.

In the literature, GAs have been applied to monophonic musical passages (Rutherford 2009), where a gene represents the string and fret combination for a single note, and also to polyphonic musical passages (Tuohy and Potter 2005; Tuohy and Potter 2006b), where a gene represents the fretboard positions required to perform a chord. These polyphonic tablature arrangement algorithms were later extended to include fingering estimation through the use of a neural network (Tuohy and Potter 2006c).

There are both advantages and disadvantages of using a GA for guitar tablature arrangement. The most significant advantage is that GAs produce multiple solutions to a problem. For example, a GA with an initial population size of 400 will produce the same amount of possible tablature arrangements. One can then retrieve the top n tablature arrangements by ranking the individuals in the final population by fitness and returning

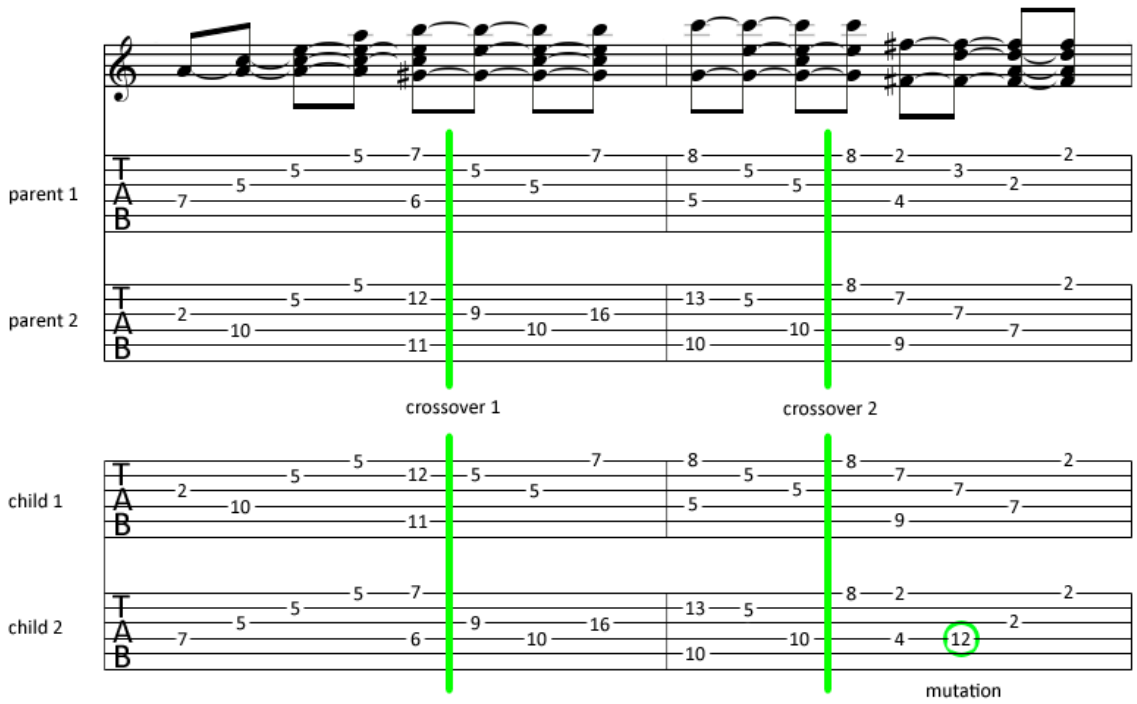


Fig. 2.9: Two parent guitar tablatures mating with two crossover points and a gene mutation in the second child.

the first n arrangements. The most significant disadvantage of using a GA is that there is no guarantee that the algorithm will converge to a global maximum in the optimization space. However, solutions found at local maxima in the optimization space may still offer "good" tablature arrangements.

2.3 Guitar-specific Transcription Systems

Instead of combining a general-purpose polyphonic transcription algorithm with a guitar tablature arrangement algorithm, guitar-specific transcription systems directly estimate the tablature arrangement by taking into consideration features of the instrument such as spectral properties of produced sounds, physical and mechanical properties of the instrument, or human performance constraints such as maximum polyphony and finger span.

Using a physical model of a plucked guitar string, Traube and Smith (2001) developed a signal processing technique for estimating the plucking point and fingering locations on

a guitar string from an audio recording. The technique revolves around the fact that the spectral envelope of a produced sound differs according to the plucking position on the string. Plucking a string close to the bridge of the guitar produces a tone that has relatively more high frequency components than a pluck close to the fretboard. There are two unknown variables: the plucking position of the string and the position of the fret that is depressed. To estimate the plucking position relative to the left-hand fingering point, the magnitude spectrum of a frame of the audio recording is compared to a set of ideal magnitude spectra calculated using the physical guitar model at different plucking positions on the string. By assuming that the right-hand fingers pluck the strings around the middle of the guitar tone hole, the remaining unknown variable (the left-hand fingering point) can be estimated. To evaluate the algorithm, a dataset of monophonic audio recordings of single plucks at different distances from the bridge of the guitar on both open and fretted strings was created. On this dataset the algorithm could accurately identify the plucking point on an open string, but struggled to identify the plucking point when the string was fretted.

Barbancho et al. (2009) presented an algorithm that estimates the pitch, string number, and, as a byproduct, the fret number, from an audio recording of a guitar. Onset detection is first performed, followed by monophonic pitch estimation using spectral peak selection of the power spectrum. Using a variety of time and frequency domain features of the audio signal, a Fisher linear discriminant is used to estimate the string the note was played on. Given the estimated pitch and string number, there exists only one candidate fret number that can produce the estimated pitch. From this information, tablature can be transcribed. The proposed algorithm received results with high variability (from 26.7%–100% accuracy) on a number of monophonic recordings of isolated guitar notes that were played on both electric and acoustic guitars with nylon and steel strings and with different playing techniques.

In order to transcribe tablature from polyphonic guitar recordings, Barbancho et al. (2012) proposed the use of an HMM to estimate the fingering of guitar chords. Each state of the HMM is a chord with a specific fingering configuration. In total, the HMM is capable of differentiating between 330 different chord fingerings. The pitch saliency function described in Section 2.1.2 by Klapuri (2006) is calculated on frames of the input guitar recording. These features are posed as observations to the HMM, which estimates the underlying chord state that contains the string and fret combinations used to perform the chord. On

a training dataset of 22 guitar recordings and a test dataset of 14 guitar recordings, the algorithm obtained 87% accuracy on average. However, the guitar transcription system is only capable of estimating a discrete number of chords and can not estimate monophonic notes.

In summary, guitar-specific tablature transcription algorithms are still in their infancy. As with instrument and genre-specific polyphonic transcription systems, guitar-specific tablature transcription algorithms are less prevalent in the literature in comparison to general-purpose transcription systems even though there is proof that "developing music transcription systems for more narrowly targeted contexts can lead to significantly improved performance" (Barbancho et al. 2012).

2.4 Alternate Transcription Methods

Various techniques for guitar tablature transcription have been proposed which process information other than an audio recording: computer vision systems (Section 2.4.1) analyze video recordings of a guitar performance; multi-modal systems (Section 2.4.2) analyze both video and audio recordings simultaneously; and augmented guitars (Section 2.4.3) rely on specialized hardware and sensors to acquire additional information about the guitar performance.

2.4.1 Computer Vision Systems

For the application of guitar tablature transcription, computer vision systems aim to use inexpensive cameras that are accessible to users (typically web cameras) to perform finger tracking or hand-shape analysis on a video recording of a guitarist during a performance. In order to detect the string and fret combinations performed by the guitarist, computer vision systems attempt to estimate from the video recording the position of frets and strings on the fretboard as well as the fingertip positions of the guitarist.

For the purposes of music education, Motokawa and Saito (2006) presented an augmented reality system that assists novice guitar players by superimposing a virtual hand over the fretboard to show how to play a given chord. The fretboard detection algorithm requires a square-shaped marker to be attached above the fretboard. The system does not implement finger position tracking to verify that the student forms the correct hand

position to play the chord. Although the augmented reality system was not capable of producing tablature, it provided a foundation for future computer vision guitar transcription systems.

Using an inexpensive video camera mounted to the neck of the guitar, Burns and Wanderley (2006) introduced a prototype system for capturing the finger positions of the left hand of a guitarist. The position of the frets and strings are determined by accentuating and grouping vertical (frets) and horizontal (strings) lines in the video recording. Fingertips, having a rounded shape in comparison to fingers, are tracked by applying an algorithm that looks for circles of a given radius in the video recording. To infer the fret and string being played, the detected fingertip positions are matched to the closest detected fret and string position on the fretboard. Movement segmentation is also performed to disregard fingering positions when transitioning between notes or chords. The proposed transcription system has some limitations: the mounted video camera could only capture the first five frets of the guitar and the top-down camera perspective hindered the transcription performance (Burns 2007). Scarr and Green (2010) extended this system by removing the fretboard-mounted video camera and instead pointed a video camera directly at the guitarist to detect the frets depressed during the performance.

With only one camera, it is difficult to determine which fingers are pressing down on a string and which are hovering (Kerdvibulvech and Saito 2008). To remedy this problem, Kerdvibulvech and Saito (2008) pointed two video cameras at the guitar at different angles. In order for the system to track finger positions, the guitarist was required to wear multi-coloured markers on each finger. Using an approach similar to Motokawa and Saito (2006), an augmented reality marker was used to detect the position of the fretboard. As a result of the stereo cameras, the system was capable of detecting the x, y, and z coordinates of each finger; however, no tablature output was generated. Since coloured markers were attached to the guitarist's fingertips, the accuracy of the finger position tracking algorithm was influenced by the colour of the background in the video recording.

2.4.2 Multi-modal Systems

Both computer vision and audio analysis guitar transcription algorithms can benefit from procuring additional information about the problem. Multi-modal systems attempt to merge computer vision and audio analysis algorithms. The main goal is to develop a

symbiotic relationship between the audio and video analysis algorithms to increase the performance of the system as a whole. For instance, analysis of the video recording may resolve ambiguities in the analysis of the audio recording, and vice versa. In the field of speech recognition, combining auditory and visual analysis of speech has shown promising results (Chibelushi et al. 2002).

Quested et al. (2008) presented a prototype polyphonic transcription system which uses the analysis of video recordings of a guitarist to supplement the analysis of audio recordings. The computer vision system seeks to locate the position of the performer, guitar neck, and left hand of the guitarist. The set of possible notes that can be produced at a specific time in the video recording is generated using the calculated spatial information of the guitarist's hand in relation to the fretboard. The resulting set of notes is used to constrain the possible fundamental frequency candidates during analysis of the audio recording. The algorithm for audio analysis has yet to be completed. To fuse the audio and video analysis, the authors plan to detect note onsets in the audio recording and analyze the video recording at the time of the note attack. Although the proposed system does not necessarily produce tablature, it provides an example of a multi-modal system that seeks to accomplish the first subtask of the guitar tablature transcription problem.

In response to the previously mentioned prototype which aims to accomplish the task of polyphonic transcription, Paleari et al. (2008) presented a multi-modal prototype system capable of guitar tablature transcription. The audio analysis component uses a monophonic pitch detection algorithm to estimate the pitch of a single note at any given time in the audio recording. A MIDI file is created which contains the pitch, onset, and duration of each estimated note. The computer vision component detects the position of the guitarist's hand in relation to the fretboard. For each note event in the MIDI file, the position of the hand is used to determine the correct fret and string combination that produced the note. Their results show that 89% of notes in the test data set, consisting of several 30 second videos, could be assigned a fret and string combination without ambiguity. As a consequence of using a monophonic pitch estimation algorithm, the described system is incapable of transcribing guitar chords.

Hrybyk and Kim (2010) proposed a multi-modal system that is capable of automatically identifying the chords in audio and video recordings of a guitar performance. Conventionally, chroma vectors are the audio feature of choice for the task of chord recognition (Fujishima 1999); instead, the authors use a technique called *Specmurt analysis* for poly-

phonic pitch detection. The result of this analysis is used to determine the chord scale (e.g., C Major). To resolve the ambiguity of the fingering used to perform the chord, video of the guitarist is analyzed to determine the chord voicing (open, barred, or first inversion). Coloured dots were placed on the fretboard to detect the position of the neck of the guitar, which, in accordance with past research that used coloured markers, places constraints on the computer vision system (Kerdvibulvech and Saito 2008). The results of an experiment where three guitarists were asked to perform various chords showed 61% accuracy of both chord scale and voicing using audio analysis alone, 33% accuracy with video alone, and 93% accuracy using both audio and video analysis.

In summary, computer vision and multi-modal guitar transcription systems are still in their infancy. Many of the proposed prototype systems impose constraints on the analysis process, such as requiring coloured markers to be placed on the guitar fretboard or fingers, or requiring multiple video cameras. Additionally, many of the proposed systems have severe limitations, only working with specific camera angles and background colours. Most importantly, only two of the reviewed systems produce tablature (Hrybyk and Kim 2010; Paleari et al. 2008). The former system is only capable of transcribing guitar chords, while the latter is only capable of transcribing single notes.

2.4.3 Augmented Guitars

Augmented musical instruments, also known as hyperinstruments, are acoustic or electric instruments that are extended by the installation of additional sensors (Miranda and Wanderley 2006). This section will review tablature transcription systems proposed in the literature that require the installation of additional sensors to the guitar.

Standard electric guitars use a "pickup", a type of sensor called a *magnetic transducer*, to convert the vibration of each string into an electrical signal. Single-coil pickups (Figure 2.10(a)) have one magnetic pole piece per string that lies directly underneath the string it is responsible for sensing. Dual-coil pickups, often referred to as "humbuckers" (Figure 2.10(b)), offer two magnetic pole pieces per string to increase the signal to noise ratio. The signal generated by each transducer is summed together and connected to the output jack of the guitar to be amplified or recorded.

Analyzing audio from the output jack of the guitar (potentially polyphonic music) instead of the recordings of individual strings (monophonic music) removes the ability

(a) Array of single-coil pickups on a Fender Stratocaster.

(b) Array of EMG-81 active dual-coil covered pickups on an LTD F-400FM.

Fig. 2.10: Example of single-coil and dual-coil guitar pickups.

of interactive systems to process and analyze the signal of individual strings (Reboursière et al. 2010) and thus complicates the task of automatic transcription (O'Grady and Rickard 2009). To circumvent this issue, O'Grady and Rickard (2009) installed a Roland GK-3[4] hexaphonic pickup to an electric guitar to capture and output the signal of each string separately (Figure 2.11(a)). By analyzing the recording of each guitar string separately, the polyphonic transcription problem becomes one of monophonic transcription, which is a solved problem (Klapuri 2004). Moreover, the ambiguity of which string and fret combination produces a note is resolved by having access to the audio recording of each string. Once the string is known, only one fret is capable of producing the note. In order to estimate the pitch of a note event (and as a by-product, the fret number), non-negative matrix factorization is used to compare the audio recording of a string to a set of templates representing all notes on the fretboard. Currently the system outputs a MIDI file containing note pitch, onset, and duration information; however, the authors plan to extend the proposed system to write tablature. The transcription system was evaluated

[4] `http://www.roland.com/products/en/GK-3`

by synthesizing the generated MIDI file and qualitatively comparing the resulting audio to the input guitar recording. Although the system produces accurate transcriptions, there are several limitations: the user must purchase a hexaphonic guitar pickup and a computer audio interface with at least six inputs. Additionally, before using the system the guitarist must provide training data by recording every note on the guitar twice—a time-consuming process that may discourage musicians from using the system.

(a) Hexaphonic pickup (Roland GK-3) installed on a Fender Deluxe Player Stratocaster (O'Grady and Rickard 2009).

(b) Sensor mount and reflective surface installed on a Cameo acoustic guitar (Fitzgerald et al. 2011).

(c) A Ztar Mark III V3 MIDI guitar controller designed by Starr Labs.

Fig. 2.11: Examples of guitars fit with special-purpose hardware for the purposes of automatic guitar tablature transcription.

As an alternative to a hexaphonic magnetic transducer pickup, Fitzgerald et al. (2011) proposed a guitar transcription product called *Guitar-2-Tab*, which requires the installation

of an infrared transmitter and receiver underneath each string in order to detect the string oscillations (Figure 2.11(b)). Each infrared sensor sends an audio signal to an Arduino microprocessor which is responsible for transcribing the tablature. The generated tablature is then saved as a text file on a memory card for future reference. This product was designed as part of a university term project and was never manufactured. However, the technical report provides a detailed description of the hardware and schematics required for manual assembly of the product.

Although not an augmented instrument by definition, MIDI guitars provide an example of special-purpose hardware that can be used to accomplish the task of automatic guitar tablature transcription. An example of a MIDI guitar controller can be seen in Figure 2.11(c)[5]. Verner (1995) proposed a method of extracting the fingering information from a performance on a MIDI guitar in realtime by assigning a separate MIDI channel to each guitar string. When a fret is depressed and a string is plucked on a MIDI guitar, the output is a MIDI note event which encodes the MIDI channel (string) associated with the note. Given a note and the string that the note is played on, the fret number can be ascertained. Although the resulting tablature is accurate, users of MIDI guitars report false note detections, performance difficulties in comparison to a standard guitar, and problems synchronizing the output of each string (Verner 1995).

Although they are accurate, guitar tablature transcription systems that require the installation of special-purpose hardware are expensive and inconvenient for users. Furthermore, some musicians may lack the technical savvy required to perform the hardware installation themselves, yielding transcription systems that are inaccessible to a certain demographic of people.

2.5 Commercial Transcription Systems

Apart from applying a polyphonic transcription algorithm to a guitar recording and manually applying a guitar tablature arrangement algorithm to the output, there are very few guitar tablature transcription systems available to the public. Kramer Guitars[6] sold special-purpose guitar transcription hardware called *Pitchrider*, developed by IVL Technologies. Similar to the approach taken by O'Grady and Rickard (2009), the Pitchrider

[5] `http://www.starrlabs.com/index.php?route=product/category&path=59_75`
[6] `http://www.kramerguitars.com`

provided a hexaphonic pickup which sent the signal of each string to a hardware interface that converted notes played on the guitar into MIDI note events. The resulting MIDI data could then be converted to tablature in a method similar to that used by Verner (1995). The Pitchrider is no longer being manufactured.

Software for guitar tablature transcription is just as scarce. To the author's knowledge, the only commercial software that performs guitar tablature transcription is *Guitarmaster*[7], a desktop application by RoboSens Ltd. that converts a polyphonic guitar recording into MIDI notation and has the option to view a tablature arrangement of the transcription in plain text format.

As for commercial polyphonic transcription programs, *Melodyne*[8] by *Celemony*[9] provides an editor that allows users to modify the pitch of individual notes in a polyphonic audio recording using a technology the company calls "direct note access". Once the audio recording has been analyzed, a MIDI file can be exported which contains the estimated note events present in the recording. To produce guitar tablature, the output MIDI file can then be input into a guitar tablature arrangement algorithm.

Transcribe![10] by *Seventh String*[11] is another commercial program that provides audio analysis tools to help musicians perform manual music transcriptions rather than automatically extracting note events from the input audio signal. The graphical interface provides a spectrum display that highlights peaks in the frequency domain and allows the user to control the playback speed without modifying the pitch content of the audio file. A thorough list of polyphonic transcription and transcription aid programs can be found on the *Seventh String* website[12].

For guitar tablature arrangement, *Guitar Pro* is a popular desktop application for editing guitar tablature which allows users to import a MusicXML or MIDI file and the software will automatically produce a guitar tablature arrangement for the sequence of input notes.

[7] http://www.guitarmaster.co.uk
[8] http://www.celemony.com/cms/index.php?id=products_editor
[9] http://www.celemony.com
[10] http://www.seventhstring.com/xscribe
[11] http://www.seventhstring.com
[12] http://www.seventhstring.com/resources/transcription.html

Chapter 3

Robotaba Guitar Tablature Transcription Framework

The previous chapter demonstrated that a large number of polyphonic transcription and guitar tablature arrangement algorithms have been proposed. However, no frameworks have been developed to facilitate the connection of polyphonic transcription and guitar tablature arrangement algorithms to produce a start-to-finish automatic guitar tablature transcription system. Moreover, after a new algorithm is developed, evaluated, and published in a conference or journal, the code has no immediately available vessel to be used by the large community of guitarists on the Internet.

To facilitate the creation of guitar tablature transcription web applications in which polyphonic transcription and guitar tablature arrangement algorithms can be embedded, a web-based guitar tablature transcription framework has been designed and implemented. A software framework is a reusable platform that allows developers to easily implement and extend the standard structure of an application. Using a framework, developers can focus on algorithmic design instead of focusing on low-level implementation details of the application. A good metaphor for a software framework is scaffolding. Suppose a construction crew shows up to work on a building and the scaffolding is already constructed for them. They immediately begin work on the building instead of constructing the scaffolding, an important task that needs to be done before work can begin but detracts from the time and energy spent on the actual project.

The implemented web-based guitar tablature transcription framework, entitled *Robotaba*, is written in the Python programming language. The framework is open source and version controlled using a git repository.[1] As a proof of concept, a state-of-the-art polyphonic transcription algorithm and a guitar tablature arrangement algorithm have been implemented and embedded in a web application using the Robotaba transcription framework.

This chapter will describe the design and implementation of Robotaba (Section 3.1) and the use of this framework to create a guitar tablature transcription web application (Section 3.2).

3.1 Framework Design

The preceding chapter formally described the guitar tablature transcription task as a function that can be decomposed into two independent functions: a polyphonic transcription function and a guitar tablature arrangement function. Guitar tablature transcription is then performed by applying the guitar tablature arrangement function to the output of the polyphonic transcription function. Emulating this structure, Robotaba is composed of three modules: a polyphonic transcription module, a guitar tablature arrangement module, and a guitar tablature engraving module. The architecture of Robotaba is displayed in Figure 3.1. Given an input audio file, the polyphonic transcription module generates a symbolic music file containing estimates of the note events occurring in the audio recording. Given an input symbolic music file containing a sequence of note events, the guitar tablature arrangement module calculates and appends a guitar string and fret combination to each encoded note event. The guitar tablature engraving module is used to display tablature encoded in a symbolic music file in the web browser.

From this modular design arises three benefits: first, the polyphonic transcription, guitar tablature arrangement, and guitar tablature engraving modules can be used independently or together. Used independently, an input file is sent directly to a module for processing, which returns a result instead of passing the output to the next module in the workflow. Using each module in sequence, guitar tablature can be generated from an input audio file and displayed in the web browser. Second, the modular design facilitates algorithm interchangeability. Assuming an algorithm produces valid output, it can be plugged

[1] `http://github.com/gburlet/robotaba`

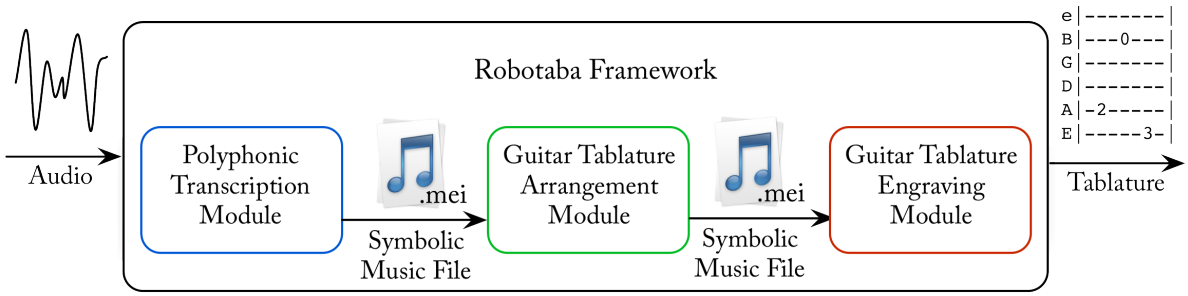


Fig. 3.1: Architecture of the Robotaba framework.

into a module without disturbing the functionality of surrounding modules. As a result, the transcription framework can accommodate new state-of-the-art polyphonic transcription or guitar tablature arrangement algorithms without substantial changes to the web application. Third, the use of a single symbolic music file format for data interchange between modules promotes polyphonic transcription and tablature arrangement algorithms to adhere to a common interface. Robotaba uses the 2012 release of the music encoding initiative (MEI)—an extensible markup language (XML) file format that encodes symbolic music notation in a hierarchical fashion (Hankinson et al. 2011).

The interaction between Robotaba modules required to generate a tablature arrangement from an audio recording and display the resulting tablature in the web browser is displayed in Figure 3.2 as a unified modeling language (UML) sequence diagram (Fowler 2003). The function and proper interpretation of UML sequence diagrams are explained in Appendix B.1. In the presented UML sequence diagram, the *PitchDetect* object represents the polyphonic transcription module, the *Tabulate* object represents the guitar tablature arrangement module, the *Engraver* object represents the guitar tablature engraving module, and the *Transcription* object essentially acts as a conductor who enforces the sequence of operations required to transcribe tablature from a guitar recording. To begin the transcription process, a message is passed to the Transcription object along with an audio file to be transcribed. The Transcription object forwards this message to the PitchDetect object, which is responsible for executing the polyphonic transcription algorithm on the audio recording and returning the resulting symbolic music file to the Transcription object. A message is subsequently passed to the Tabulate object, which is responsible for executing the guitar tablature arrangement algorithm on the provided symbolic music file.

The Tabulate object then requests the Engraver object to display the generated tablature arrangement in the web browser, and subsequently returns the resulting symbolic music file to the Transcription object, ending the sequence of interactions.

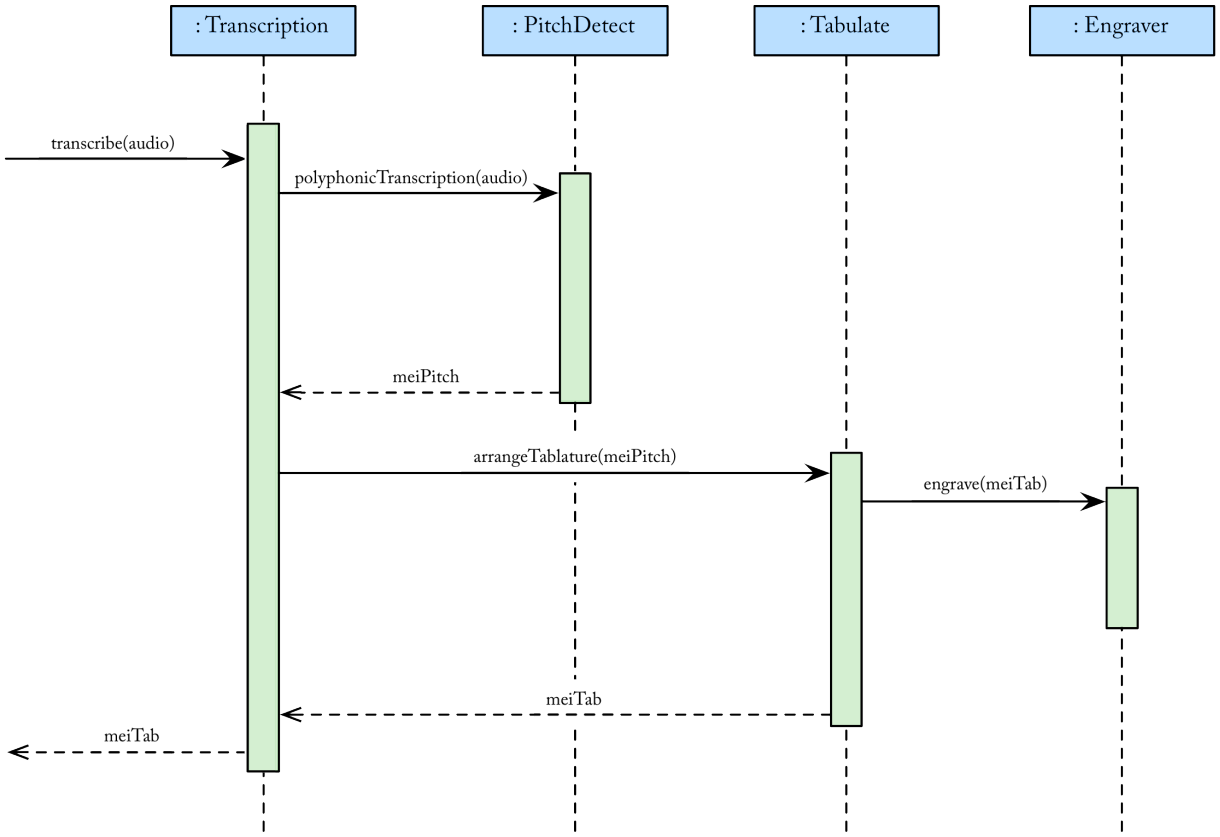


Fig. 3.2: UML sequence diagram displaying the interaction of modules within the Robotaba framework to produce a guitar tablature arrangement from an audio recording.

Alternatively, the Transcription object may be circumvented to perform polyphonic transcription and guitar tablature arrangement independently. To perform the task of polyphonic transcription only, the audio file is sent directly to the PitchDetect object for processing. To perform the task of guitar tablature arrangement only, a symbolic music file is sent directly to the Tabulate object for processing. To perform the task of guitar tablature engraving only, a symbolic music file is sent directly to the Engraver object for processing.

3.1.1 User Interface

A user interface is provided to allow the user to choose a transcription workflow and enter processing parameters. Four workflows are available to the user: a polyphonic transcription workflow, a guitar tablature arrangement workflow, a digital tablature engraving workflow, and a guitar tablature transcription workflow. Each of these workflows require the user to upload a file to be processed and to enter additional information, described in this section.

Music Metadata

Robotaba requires the user to input metadata for uploaded files so that they may be catalogued and searched. The metadata required for a musical work includes the title, artist, and copyright holder. When uploading an audio file, a web form requests the user to manually enter the appropriate metadata. When uploading a symbolic music file, which often provides metadata for the musical work encoded within, the file is parsed to retrieve the appropriate information. If this information is absent, a web form requests the user to enter the appropriate metadata, which is then injected into the uploaded symbolic music file. Furthermore, for each MEI file produced at subsequent steps in the selected workflow, the user-entered metadata is automatically injected into the symbolic music file.

Guitar Model

The polyphonic transcription module and guitar tablature arrangement module, described in the following sections, make use of specific parameters of the user's guitar. To gather this information, Robotaba provides a form for the user to specify the number of frets, the tuning, and capo position of their guitar. The function of a guitar capo is explained in Appendix A. The tuning is specified by entering the pitch name and octave of each open-string pluck[2] of the guitar.

3.1.2 Polyphonic Transcription Module

The polyphonic transcription module is responsible for estimating the note events occurring in an input audio file. The architecture of the polyphonic transcription module is illustrated in Figure 3.3. Specifically, the polyphonic transcription module accepts an audio file as

[2]An open-string pluck refers to a pluck of a string without depressing any frets.

input. This audio file is passed to the polyphonic transcription algorithm embedded in the module. The polyphonic transcription algorithm is responsible for generating an MEI file containing the estimates of note events occurring in the input audio file. Any polyphonic transcription algorithm that generates an MEI file which encodes the estimated note events may be inserted into the polyphonic transcription module. The resulting symbolic music file is optionally post-processed to ensure the estimated pitches of the note events are capable of being performed on the user's guitar. When the polyphonic transcription module is used independently, the user is able to toggle post-processing of the symbolic music file. Otherwise, post-processing is enabled by default.

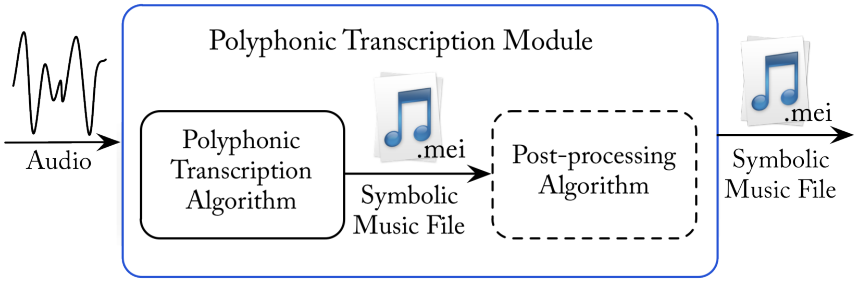


Fig. 3.3: Architecture of the polyphonic transcription module.

A significant advantage of optionally post-processing the output symbolic music file is that polyphonic transcription algorithms that are not guitar-specific may be embedded in the polyphonic transcription module. Furthermore, guitar-specific polyphonic transcription algorithms may also benefit from having access to parameters of the specific guitar which is producing the sound.

Symbolic Music File Post-processing

If post-processing is enabled, the polyphonic transcription module imposes two constraints on the output symbolic music file. First, the estimated pitches of note events that are outside of the range of the user's guitar should be discarded or transposed. The user is able to select whether notes are to be discarded or transposed. If the erroneous note events are to be discarded, they are simply removed from the symbolic music file. If the erroneous note events are to be transposed within range of the guitar, some calculations must be

performed. To calculate the lower pitch bound of the guitar, the pitch produced by an open pluck of the thickest gauged string is retrieved from the tuning parameter provided by the user. This pitch is then raised by c semitones, where $c \in \mathbb{N}$ describes the capo position on the guitar. A capo position of zero denotes that no capo is used. To calculate the upper pitch bound of the guitar, the pitch produced by an open pluck of the thinnest gauged string is retrieved from the tuning parameter provided by the user. This pitch is then raised by n semitones, where $n \in \mathbb{N}^+$ describes the number of frets on the guitar. The note transposition algorithm systematically lowers or raises the octave of the note until it resides within the pitch range of the user's guitar.

Second, the degree of polyphony of the remaining note event estimates must be limited to six—the maximum possible polyphony of a standard guitar. To enforce this, the post-processing algorithm orders the note event estimates within a chord by ascending pitch. Starting from the highest pitch, the number of notes necessary to yield a polyphony less than or equal to six are discarded.

3.1.3 Guitar Tablature Arrangement Module

The guitar tablature arrangement module is responsible for assigning a guitar string and fret combination to each note occurring in an input symbolic music file. The architecture of the guitar tablature arrangement module is illustrated in Figure 3.4. Specifically, the guitar tablature arrangement module accepts an MEI file as input, which is first pre-processed to ensure the pitches of the encoded note events are capable of being performed on the user's guitar. This pre-processing step is mandatory and follows the same algorithm as the post-processing step used in the polyphonic transcription module described in the previous section. The resulting MEI file is passed to the guitar tablature arrangement algorithm embedded in the module. Any guitar tablature arrangement algorithm that generates an MEI file that encodes the string and fret combinations required to perform the sequence of input notes may be inserted into the guitar tablature arrangement module.

3.1.4 Guitar Tablature Engraving Module

The guitar tablature engraving module is responsible for parsing an MEI file containing a sequence of note events that have each been assigned a guitar string and fret combination and displaying the encoded tablature in the web browser. The architecture of the guitar

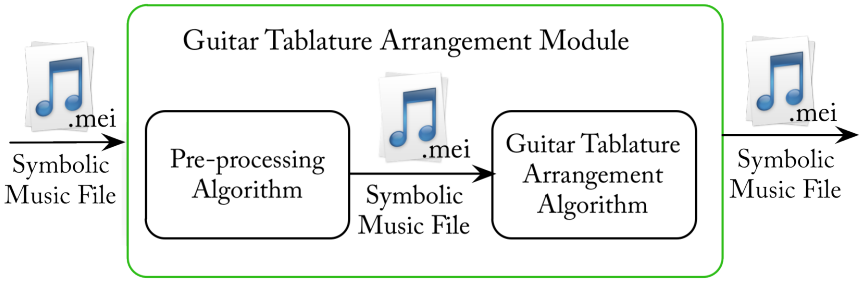


Fig. 3.4: Architecture of the guitar tablature arrangement module.

tablature engraving module is illustrated in Figure 3.5. The structure of this module differs from the polyphonic transcription and guitar tablature arrangement modules in that a tablature rendering algorithm does not have to be implemented and inserted into the module. In order to display tablature in the web browser, this module uses the digital guitar tablature engraving library *AlphaTab* to render tablature symbols on the HTML canvas element.[3] The most significant advantage of rendering tablature on the HTML canvas instead of using alternative multimedia display technologies such as Adobe Flash is that the majority of available web browsers, operating systems, and devices are capable of viewing the HTML canvas.

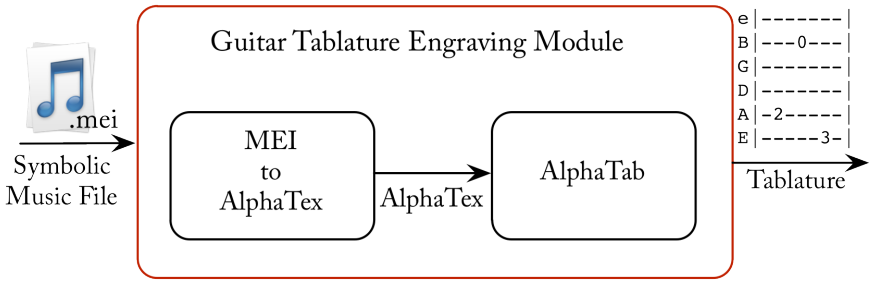


Fig. 3.5: Architecture of the guitar tablature engraving module.

AlphaTab parses drawing scripts called *AlphaTex*, in which structured keywords inform the rendering engine about the contents of the tablature and how it should be displayed.

[3] `http://www.alphatab.net`

When an MEI file is passed to the tablature engraving module, the contents of the file are converted to an AlphaTex drawing script to be rendered by AlphaTab in the web browser. Figure 3.6 illustrates an excerpt of an MEI file that is displayed using AlphaTab.

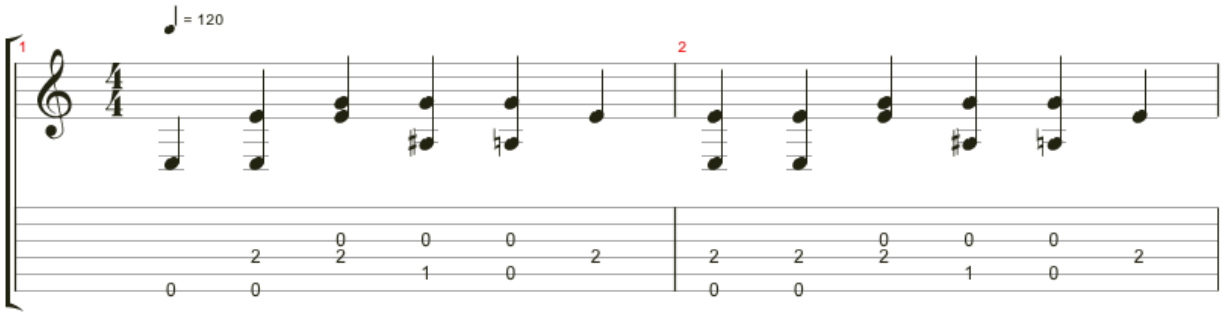


Fig. 3.6: An excerpt of an MEI encoding of "Enter Sandman" by Metallica, displayed using the *AlphaTab* digital guitar tablature engraving library.

3.1.5 Technical Details

Now that the design of the transcription framework and its modules have been presented, this section will describe the technical details and mechanics of Robotaba. Specifically, the underlying relational database and details regarding the implementation of the framework will be discussed.

Database Schema

Robotaba maintains a relational database that is responsible for storing references to uploaded audio and symbolic music files as well as the corresponding metadata for these files. An index is constructed on the title and artist fields in order for the user to rapidly query files uploaded to or generated by the web application. Although the metadata for a musical work is often encoded within a symbolic music file, this information is replicated in the database. At the expense of data redundancy, queries will execute faster because the metadata encoded in each symbolic music file on the hard disk does not have to be searched at query time. The database also encodes the relationship between input and output files. In other words, the database encodes the workflow or path of modules the input file was sent through to generate the output file. The database is also responsible for recording a

time stamp when a module begins processing an input file and when the module returns an output file. With this data, researchers can gather statistics regarding the efficiency of their algorithms on each input file processed and compare these runtime statistics to other algorithms.

The database schema for Robotaba is presented in Figure 3.7 as an entity relationship (ER) diagram (Chen 1976). The purpose and proper interpretation of an ER diagram is explained in Appendix B.2. The ER diagram depicts eight interrelated entities. The `Transcription` entity models a complete guitar tablature transcription. This involves sending an input audio file to the polyphonic transcription module (modelled by the `PitchDetect` entity), and passing the output symbolic music file to the guitar tablature arrangement module (modelled by the `Tabulate` entity). An audio file is represented by the `Audio` entity; a symbolic music file in the MEI file format that contains note event estimates is represented by the `MeiPitch` entity; and a symbolic music file in the MEI file format that encodes notes with assigned string and fret combinations is represented by the `MeiTab` entity. Each of these files must be associated with metadata describing the musical work, represented by the `MetaMusic` entity. The `GuitarModel` entity represents a model of the user's guitar.

Each entity has attributes which provide information about the entity. Attributes associated with the `GuitarModel` entity describe the number of frets, tuning, and capo position of the user's guitar. The `Audio`, `MeiPitch`, and `MeiTab` entities have attributes that encode the path to the physical file on the hard disk and the time in which the file was uploaded. Attributes associated with the `MetaMusic` entity describe the title, artist, and copyright metadata of a musical work. The `PitchDetect` entity has an attribute `sanitize`, which describes whether the symbolic music file post-processing algorithm should discard, transpose, or leave notes outside of the range of the user's guitar. Similarly, the `Tabulate` entity has the same attribute, although it describes whether the symbolic music file pre-processing algorithm should discard or transpose notes.

To describe the relationship between entities, crow's foot notation is used. An explanation of crow's foot notation is provided in Appendix B.2. A discussion of the important entity relationships follows. The `Transcription` entity must invoke one `PitchDetect` entity followed by one `Tabulate` entity. However, the inverse relationship differs. The `PitchDetect` and `Tabulate` entities do not necessarily have to be invoked by the `Transcription` entity, since the polyphonic transcription and guitar tablature arrange-

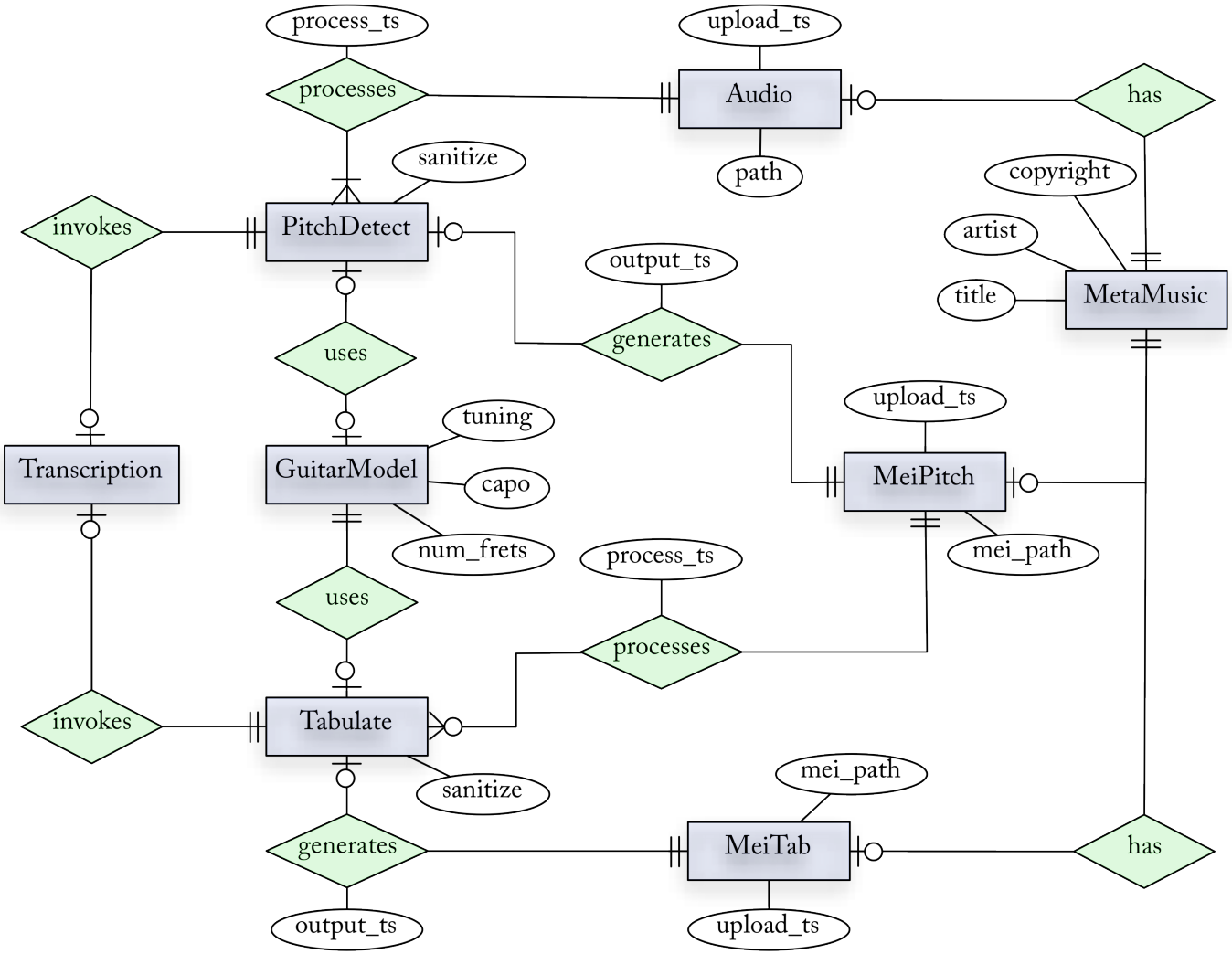


Fig. 3.7: Entity Relationship diagram for the Robotaba framework. For information on how to interpret this diagram, see Appendix B.2.

ment modules may be used independently. The `PitchDetect` entity processes exactly one audio file and generates exactly one MEI file containing note event estimates represented by the `MeiPitch` entity. The `Tabulate` entity processes exactly one MEI file containing a sequence of note events and generates exactly one MEI file that encodes the produced guitar tablature arrangement.

Relationships may also have attributes. For example, the *processes* relationship connecting the `PitchDetect` and `Audio` entities has an attribute that describes the time that processing of the audio file began. As well, the *generates* relationship connecting the `PitchDetect` and `MeiPitch` entities has an attribute that describes the time that processing finished and the output symbolic music file was generated. By subtracting these two time stamps, the runtime of the polyphonic transcription module on the input audio file can be derived. This structure is mirrored for the `Tabulate` entity. Therefore, a similar method is employed for calculating the runtime of the guitar tablature arrangement module on an input symbolic music file.

Framework Implementation

Robotaba is implemented using Django, a web framework written in Python.[4] The framework facilitates rapid development of database-driven web applications by providing an "object-relational mapper" that translates Python classes called models into structured query language (SQL) commands that automatically create the proper database tables. Each entity in Figure 3.7 is implemented as a Django model. The attributes of an entity are implemented as member variables of the Django model. Moreover, the relationships between entities can be enforced by specifying these relationships in the Django model. An example of a Django model that represents the `Audio` entity is displayed in Listing 3.1. By virtue of using the Django web framework, Robotaba also provides a graphical user interface for administrators of the web application to view and edit the contents of the database without requiring knowledge of SQL.

Another important aspect of the Django web framework are Django views. Django views are functions that return a HTML response to the client web browser and are called when a specific uniform resource locator (URL) is accessed. Within the function, the database can be queried or a server-side process can be triggered. Robotaba uses Django views to

[4] `http://www.djangoproject.com`

Listing 3.1: Django model for the Audio entity

```
class Audio(models.Model):
    fk_mid = models.ForeignKey(MetaMusic)
    upload_ts = models.DateTimeField(auto_now_add=True)
    audio_file = models.FileField(upload_to="audio")
```

send uploaded files through its modules. For example, when accessing the relative URL: "`/transcribe/<aid>/`" in Robotaba, a function is called that passes the uploaded audio file with the database identifier `<aid>` to the polyphonic transcription module to begin the guitar tablature transcription process. Similar functionality exists for the polyphonic transcription, guitar tablature arrangement, and guitar tablature rendering modules.

Symbolic Music Encoding Transformations

To interface with commercial music applications, such as *Guitar Pro*, that are not capable of reading MEI files, a program for converting MEI to MusicXML files and MusicXML to MEI files has been written in the Python programming language.[5] Although both MusicXML and MEI encode music symbols using XML, MusicXML adheres to a different encoding schema than MEI. As the structure of these files can be quite complex and there are many different elements and relationships to account for, only a subset of elements are translated between the file formats.

Using the implemented file format converter, Robotaba allows symbolic music files to be uploaded or downloaded in both the MEI and MusicXML file format. When a MusicXML file is uploaded, it is immediately converted to an MEI file before being saved on the hard disk and passed to a module in Robotaba for processing. Similarly, when a symbolic music file is requested for download in the MusicXML file format, the MEI file on the hard disk is converted to a MusicXML file and returned to the user. By allowing MusicXML files to be uploaded, tablature that has been manually entered into the Guitar Pro desktop application can be exported in MusicXML and uploaded to a web application that uses the Robotaba framework, where it may be displayed and shared online. Moreover, tablature arrangements that have been produced by a web application that uses Robotaba can be downloaded as a MusicXML file, edited in Guitar Pro, and re-uploaded to the website.

[5] `http://github.com/gburlet/musicxml-mei-conversion`

3.2 Guitar Tablature Transcription Web Application

Using the Robotaba framework, a web application for guitar tablature transcription has been developed that incorporates the polyphonic transcription algorithm and the guitar tablature arrangement algorithm described in this section.

3.2.1 Polyphonic Transcription Algorithm

A state-of-the-art polyphonic transcription algorithm (Zhou and Reiss 2008) has been implemented. This algorithm was selected for various reasons. First, this algorithm ranked highest out of the polyphonic transcription algorithms evaluated in the MIREX on the piano dataset from 2007–2012, when considering the accuracy of pitch and note onsets only. Second, the authors claim to have tuned underlying parameters of this algorithm according to a dataset composed of both piano and guitar recordings (Zhou and Reiss 2008). Third, this algorithm is capable of performing polyphonic transcriptions in realtime. Finally, the source code[6] of this algorithm is provided under the terms of the GNU General Public License, meaning the source code can be used, modified, and redistributed without financial compensation.

The polyphonic transcription algorithm developed by Zhou and Reiss (2008) is distributed as a Vamp plugin written in the C++ programming language. A Vamp plugin[7] is an audio feature extraction module that can be "plugged into" a host application such as Sonic Visualiser[8], Sonic Annotator[9], or Audacity[10]. The host application provides a graphical user interface or command line tool to allow an audio file to be imported and parameters of the plugin to be set. The host application is responsible for preprocessing and partitioning the input audio signal into frames, which is then sent to the plugin for analysis and feature extraction. The host is also responsible for exporting the features produced by the plugin to a standard file format.

In order to embed the polyphonic transcription Vamp plugin into Robotaba, the source code was manipulated in various ways. First, the plugin was divorced from the host to

[6] http://www.vamp-plugins.org/plugin-doc/qm-vamp-plugins.html#qm-transcription

[7] http://www.vamp-plugins.org

[8] http://www.sonicvisualiser.org

[9] http://omras2.org/SonicAnnotator

[10] http://audacity.sourceforge.net

produce a standalone application. To access the standalone application from Robotaba, a Python interface was created using the Boost.Python library.[11] The Boost.Python library allows C++ classes, functions, and data structures to be wrapped and subsequently accessed from Python. These wrappers are referred to as "Python bindings". To reintroduce the functionality of the host application, an open-source Python module has been implemented that sets parameters used by the polyphonic transcription algorithm, imports an audio file, sends the audio data to the Python bindings of the polyphonic transcription Vamp plugin, and generates an MEI document containing the resulting note event estimates.[12].

Symbolic Music Encoding

To generate the MEI document containing the note event estimates, the Python bindings of libmei are used. Libmei is an open-source C++ library for reading and writing MEI files.[13] The resulting MEI document is then presented to Robotaba, which optionally post-processes the encoded note events.

In regard to the structure of the MEI document, the 2012 release of the MEI is used to encode the note event estimates. Being an XML file, the MEI document inherently encodes musical events in a hierarchical manner. For example, a note may be a member of a chord, which is a member of a staff, and so forth. Another important feature of MEI is the ability to relate musical elements in the document to their temporal location in an external audio file through the specification of timestamps. This relationship is encoded using the `<timeline>` element. An example of a note and a chord referencing timestamps in an external audio file is presented in Listing 3.2.

3.2.2 Guitar Tablature Arrangement Algorithm

An open-source guitar tablature arrangement algorithm named DarwinTab[14] has been developed, written in the Python programming language, and embedded in the Robotaba guitar tablature arrangement module. DarwinTab uses a GA to produce tablature arrangements of notes encoded in an MEI file. A GA was selected for various reasons. First,

[11] `http://www.boost.org/libs/python/doc`
[12] `http://github.com/gburlet/zhoutranscription`
[13] `http://ddmal.music.mcgill.ca/libmei`
[14] `http://github.com/gburlet/darwin-tab`

Listing 3.2: MEI example of a note and a chord referencing timestamps in an audio file.

```
<timeline id="timeline" avref="/path/to/audiofile.wav" origin="when_note">
  <when id="when_note" absolute="0.00"/>
  <when id="when_chord" absolute="0.04"/>
</timeline>

<!-- in the score encoding -->
<note id="note_1" pname="C" oct="4" when="when_note"/>
<chord id="chord_1" when="when_chord">
  <note id="note_2" pname="D" oct="4"/>
  <note id="note_3" pname="E" oct="4"/>
</chord>
```

GAs have been shown to consistently yield playable tablature even when departing from the structure of the published tablature (Tuohy and Potter 2005). Second, a GA is a stochastic search algorithm and is therefore capable of producing multiple tablature arrangements for the same input file. Therefore, users of the web application have the option of generating an alternate tablature arrangement by reprocessing the input symbolic music file. Finally, the fitness function of the implemented GA can also be used in the algorithm evaluation process to compare guitar tablature arrangements, since the fitness function quantitatively defines metrics that characterize a "good tab".

To begin, the initial population of the GA must be formed. The initial population of the GA consists of a set of candidate tablature arrangements (chromosomes) that are valid, but not necessarily possible or easy to perform. The structure of a chromosome for an input music score with one measure is illustrated in Figure 3.8. The chromosome consists of a sequence of genes. Each gene represents a pluck or a strum of the guitar.

Based on the work by Allen and Goudeseune (2011), which addresses instrument tuning in the guitar fingering problem, DarwinTab uses a guitar-specific model to produce candidate tablature arrangements that take into consideration the number of frets, tuning, and capo position of the guitar on which the tablature will be performed. The guitar-specific model is responsible for calculating a set of candidate string and fret combinations for each note encoded in the input symbolic music file. The candidate string and fret combinations for a single note are calculated by first discerning the pitch of an open pluck of each string on the guitar, which is defined by the tuning of the guitar. If a capo is placed on fret

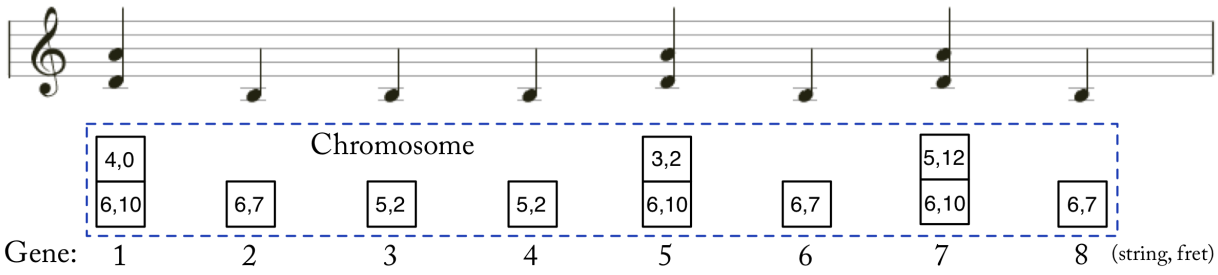


Fig. 3.8: The structure of a chromosome for a sample music score. The chromosome contains a sequence of eight genes, each consisting of a set of string and fret combinations.

number $c \in \mathbb{N}$, then c semitones are added to the pitch of an open pluck of each string. Then for each fret $f \in \{\mathbb{N} : f \leq n\}$ on the fretboard, such that $n \in \mathbb{N}^+$ is the number of frets on the guitar, f semitones are added to the pitch of each open-plucked string. If any of the resulting pitches match the pitch of the note event being processed, the string and fret combinations are added to the set of candidate fretboard positions. To produce a candidate tablature arrangement for the input music score, a string and fret combination is randomly chosen from the set of candidate string and fret combinations for each note.

Mimicking the process of natural evolution, the GA evolves the initial population of candidate tablature arrangements by selecting fit individuals from the population for mating. There are many possible methods for mate selection: Tuohy and Potter (2005, 2006c) use binary tournament selection, where two pairs of individuals are randomly selected from the population and the most fit from each pair are mated; Rutherford (2009) uses roulette-wheel selection, where the probability of an individual being selected is proportional to its fitness; and Tuohy and Potter (2006a) use rank-based selection, which selects individuals in the population based on its fitness rank with respect to other individuals. DarwinTab uses roulette-wheel selection to choose mates from the current population to form the successor population. Using conventional genetic crossover techniques, the selected parents are optionally mated with a certain probability. Otherwise, no crossover occurs and the children are genetic clones of their parents.

DarwinTab evolves tablature for a given number of generations (iterations) before terminating. In the final generation, there exists $n_{pop} \in \mathbb{N}^+$ tablature arrangements that have evolved from the initial population consisting of n_{pop} candidate tablature arrangements. For the purposes of the transcription web application, which expects a single tablature

arrangement to be returned, the candidate tablature arrangement with the highest fitness value is selected. The formula for calculating the fitness of a tablature arrangement is outlined in the following section.

DarwinTab has many operating parameters to control the evolutionary process of the underlying GA. These parameters include the population size, the maximum number of generations to run, the probability of mating, the number of genetic crossover points, and the probability of gene mutation. DarwinTab does not include note ornamentations, such as guitar bends and slides, in the resulting tablature.

Fitness Function

The fitness function of a GA is paramount to the performance of the genetic search process because it quantitatively defines the properties that contribute to a "good tab". Following the study of the left-hand movements of professional guitar players (Heijink and Meulenbroek 2002), DarwinTab incorporates into its fitness function the three biomechanical complexity factors contributing to the performance difficulty of a tablature arrangement: the position of the left hand on the guitar neck, the distance that the left hand must move to accommodate note transitions, and the finger span required to perform chords. For each note or chord in the input symbolic music score being processed, a difficulty score is produced according to each complexity factor.

For the first complexity factor, Heijink and Meulenbroek (2002) found that professional guitarists favour hand positions near the beginning of the fretboard. Therefore, notes with candidate string and fret combinations where the fret number is greater than seven are penalized to encourage tablature arrangements near the beginning of the fretboard. This fret number was chosen subjectively, according to familiarity with the instrument.

The second complexity factor emphasizes that arrangements with large fretwise-distances between consecutive notes are more difficult to perform. Therefore, for two consecutive notes with fret numbers $f_1, f_2 \in \{\mathbb{N} : f \leq n\}$ such that $n \in \mathbb{N}^+$ is the number of frets on the guitar, the difficulty score is calculated by the formula

$$\text{abs}\left(f_1 - f_2\right). \tag{3.1}$$

For a note played by depressing fret number f, followed by a chord comprised of multiple notes with the set of fret numbers g, the difficulty score is calculated by the formula

$$\text{abs}\left(f - \left(\frac{\max(g) - \min(g)}{2}\right)\right). \tag{3.2}$$

The third complexity factor ensures that chords that require excessively large finger spans are penalized. Therefore, for a chord comprised of multiple notes with the set of fret numbers g, the difficulty score is calculated by the formula

$$\max(g) - \min(g). \tag{3.3}$$

The difficulty scores for each complexity factor are aggregated across all of the notes in the symbolic music score being processed to produce a vector $\mathbf{c} \in \mathbb{R}^3$ of difficulty scores for the entire piece. These difficulty scores are weighted and summed to produce a final difficulty score $d \in \mathbb{R}^+$ for the tablature arrangement, such that

$$d = \mathbf{w} \cdot \mathbf{c} = \sum_{i=1}^{3} w_i c_i, \tag{3.4}$$

where the vector $\mathbf{w} \in \mathbb{R}^3$ contains hand-tuned weights for each difficulty score. From the final difficulty score, the fitness $f \in \mathbb{R}^+$ of an individual is calculated using the formula

$$f = \frac{1}{1+d}. \tag{3.5}$$

The goal of the GA is to search for a tablature arrangement that minimizes the difficulty score d, or equivalently, maximizes the fitness f.

The difficulty score associated with performing each note or chord in a given symbolic music score could be precomputed to reduce the number of computations required to evaluate the fitness function of a candidate tablature arrangement. This could be accomplished by creating a directed weighted graph, in which each vertex represents a candidate string and fret combination for a note or chord. Vertices that correspond to adjacent notes or chords in the symbolic music score are connected by an edge. The weight of an edge between two vertices denotes the difficulty score associated with the transition to, and performance of, the latter note or chord on the guitar. The fitness of a candidate tablature arrangement

can then be computed by tracing a path through the constructed graph that corresponds to the sequence of string and fret combinations occuring in the candidate tablature arrangement, aggregating the edge weights along this path to attain the final difficulty score d, and using Equation 3.5.

Symbolic Music Encoding

After the GA has finished evolving the tablature population, the tablature arrangement with the highest fitness is exported as an MEI document. To encode the tablature arrangement, a string and fret combination is appended to each `<note>` element encoded in the input MEI file. Each gene in the chromosome saves a reference to the unique identifier of its corresponding `<note>` element in the input MEI file. For each gene, the Python bindings of libmei are used to lookup and attach the calculated string and fret combination to the `<note>` element in the MEI document with the appropriate identifier. A sample tablature arrangement is displayed in Listing 3.3 for the notes encoded in the input MEI file presented in Listing 3.2.

Listing 3.3: MEI example of notes with appended string and fret information.

```
<note id="note_1" pname="C" oct="4" tab.string="2" tab.fret="1"/>
<chord id="chord_1">
  <note id="note_2" pname="D" oct="4" tab.string="2" tab.fret="3"/>
  <note id="note_3" pname="E" oct="4" tab.string="1" tab.fret="0"/>
</chord>
```

Chapter 4

Transcription Evaluation

The previous chapter presented the guitar tablature transcription framework *Robotaba* and its use in the creation of a web application that utilizes a polyphonic transcription algorithm in conjunction with a guitar tablature arrangement algorithm to generate and display tablature in the web browser. As a framework, Robotaba itself is not capable of performing guitar tablature transcriptions; thus, the framework itself can not be quantitatively evaluated. However, the implemented web application serves as a testament to the utility of the framework.

This chapter will focus on the procedure for evaluating the implemented web application, specifically the polyphonic transcription and guitar tablature arrangement algorithms embedded within. Given that the modular design of Robotaba allows the polyphonic transcription and guitar tablature arrangement algorithms to be used independently, the algorithms will also be evaluated independently. The polyphonic transcription algorithm will be evaluated by comparing the output of the algorithm on several synthesized guitar recordings to a dataset of correct polyphonic transcriptions. The guitar tablature arrangement algorithm will be evaluated by comparing the output of the algorithm on several symbolic music scores to a dataset of acceptable guitar tablature arrangements. Several experiments are proposed to evaluate these algorithms, the results of which are presented and discussed in the following chapter.

The structure of this chapter is as follows: Section 4.1 will present the datasets used for evaluating the algorithms. Section 4.2 and Section 4.3 will describe the experimental

methodology and metrics used in the evaluation of the polyphonic transcription and guitar tablature arrangement algorithms, respectively.

4.1 Description of Datasets

Datasets that provide correct or acceptable output alongside the input data are important for algorithm evaluation and for the training and validation of models formed by supervised learning algorithms. In the field of machine learning and MIR, such a dataset is often referred to as a ground-truth dataset. This section will present the ground-truth datasets used for evaluating the polyphonic transcription and guitar tablature arrangement algorithms.

4.1.1 Polyphonic Transcription Dataset

For the purposes of evaluating polyphonic transcription algorithms which output the pitch, onset time, and duration of each estimated note event in an input guitar recording, a ground-truth dataset should contain a set of audio recordings that are each paired with a list of the note events occurring in the recording (Bay et al. 2009).

Currently there does not exist an ubiquitous ground-truth dataset for polyphonic guitar transcription as there does for polyphonic piano transcription (Poliner and Ellis 2006). This is largely due to the fact that a Yamaha Disklavier[1] can easily create real piano recordings that are aligned with ground-truth note events (Benetos et al. 2012). However, several datasets have been compiled for monophonic and polyphonic guitar transcription. For monophonic guitar transcription, the RWC Musical Instrument Sound Data Base (Goto et al. 2003) provides isolated recordings of individual plucks of various guitars with different dynamic levels. Although the number of recorded samples is relatively large, the recordings are strictly monophonic. Another dataset of isolated guitar pluck recordings was compiled by Abesser (2012), which was gathered from the larger dataset compiled by Stein et al. (2000) consisting of guitar pluck recordings with and without digital audio effects. In this dataset, the monophonic recordings were also combined to produce polyphonic recordings; however, these recordings only spanned the first twelve frets of the guitar. Other projects have used synthesis algorithms to generate polyphonic audio signals that represent single

[1] A Disklavier is an acoustic piano that is mechanically operated by solenoids, which are typically controlled by MIDI input.

guitar chords (Gagnon et al. 2004) or have created small datasets of recordings of isolated guitar chords (Bonnet and Lefebvre 2003).

The lack of a ground-truth dataset for the polyphonic transcription of full-length guitar songs motivates the compilation of a new dataset. Toward the creation of a ground-truth dataset for polyphonic guitar transcription, this section will present a semi-automated method for harvesting the wealth of community-moderated and publicly available data present on guitar tablature websites. Using the proposed dataset creation method, a new ground-truth dataset for polyphonic guitar transcription is compiled and presented.

Dataset Creation Methodology

The semi-automated dataset creation method presented in this section focuses on the processing of *Guitar Pro* symbolic music files. Similar to other symbolic music notation file formats, the Guitar Pro file format is capable of encoding the metadata, tempo information, and symbolic music data of a musical work. The symbolic music data is organized into instrument tracks, wherein the pitch and fretboard location of notes are encoded. Due to the proprietary file format, the exact structure of information encoded in the symbolic music file is unknown. As such, the raw file can not be parsed by third-party applications to automate processing of these files. Moreover, there is no application programming interface to manipulate an encoded file and no command line tools available for batch processing of Guitar Pro files. Consequently, the graphical user interface of the Guitar Pro desktop application must be used to manipulate the downloaded files and export the encoded data into other file formats.

The Guitar Pro desktop application has many features that make it suitable for musical dataset creation. The desktop application allows Guitar Pro files to be synthesized using different instrument models and exported as an audio file. For instance, one can select a specific guitar model, an amplifier model, and a series of digital audio effects to mimic the sound of a variety of guitar configurations. The export function is also capable of translating the information encoded in the proprietary data format to the MusicXML file format. Furthermore, Guitar Pro supports a variety of different instruments apart from the guitar. Without loss of generality, the same ground-truth dataset creation method proposed in this section could be applied to other stringed instruments such as the bass guitar.

To form the ground-truth dataset for polyphonic guitar transcription, a set of Guitar Pro files is first compiled. A plethora of manual tablature transcriptions in the Guitar Pro file format are available on guitar tablature websites on the Internet, the most popular website being `www.ultimate-guitar.com` (see Figure 1.6). Each Guitar Pro file is then synthesized to create an audio file. Subsequently, a file listing the pitch, onset time, and duration of notes occurring in the synthesized audio file is created. This file will be referred to as the *ground-truth file*. The method of synthesizing and creating the ground-truth file for a single Guitar Pro file is described in detail below.

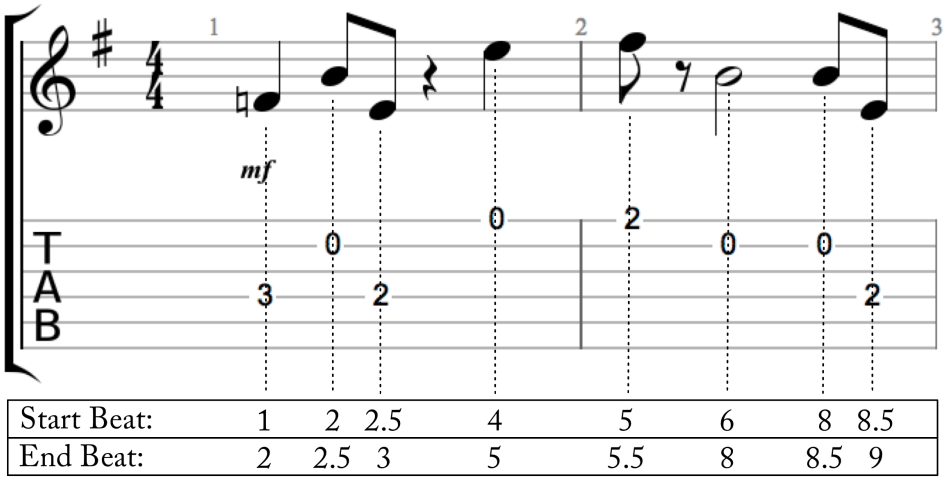
Before synthesizing the audio, a Guitar Pro file must undergo several preprocessing steps. Guitar Pro files often contain multiple instrument tracks, many of which are not guitar tracks. Extraneous tracks containing instruments such as the bass guitar, drums, and vocals are removed. Finally, all tempo, volume, and pan automations are removed from the remaining guitar track. Removing the tempo automations ensures that the entire song follows a constant tempo—a preprocessing step that is necessary for the calculation of the onset time and duration of note events in the ground-truth file creation procedure described at the end of this section.

To synthesize the preprocessed Guitar Pro file using the Guitar Pro desktop application, a guitar model, amplifier model, and desired audio effects must first be selected. Guitar Pro has a variety of presets that automatically select the guitar, amplifier, and audio effects used for synthesis. Two such presets exist for clean guitar and distortion guitar.[2] The clean guitar preset consists of a Stratocaster electric guitar model with single coil pickups, an amplifier model with default settings, and no digital audio effects. The distortion guitar preset consists of a Les Paul electric guitar model with humbucker pickups, an amplifier with default settings, and a "Screamer Overdrive" guitar pedal with default settings for the application of a distortion audio effect.

To create the ground-truth file, the Guitar Pro desktop application is used to export a MusicXML file from the Guitar Pro file that was used to synthesize the audio file. From the MusicXML file the tuning of the guitar, as well as the pitch, string and fret combination, and temporal information of each note event can be obtained. This MusicXML file is automatically processed by a program to gather and output the pitch, onset time, and duration of notes in the symbolic music file. Nichols et al. (2009) developed a MusicXML parser program written in Matlab, which was originally used to gather information from

[2]The terms *clean guitar* and *distortion guitar* are described in Appendix A.

symbolic music scores for the purposes of finding patterns in the relationships between melody, lyrics, and instrumentation.[3] The program parses a MusicXML file and returns the pitch, start beat, and end beat of each note event present in the symbolic music encoding, among other retrieved information. The pitch of the note event is explicitly encoded in the MusicXML file; however, the start and end beat of a note event within the entire music score requires some calculation. The start beat of a note is calculated by considering the time signature of each measure and the beat duration of all previous notes and rests in the score. The end beat of a note is calculated by adding the beat duration of the note to the start beat of the note. Example output of the beat calculation algorithm for two measures of music is illustrated in Figure 4.1.



Start Beat:	1	2	2.5	4	5	6	8	8.5
End Beat:	2	2.5	3	5	5.5	8	8.5	9

Fig. 4.1: Calculation of the start and end beat of each note event in an example music score containing two measures.

Several changes were made to this program to generate a ground-truth file with the desired format. First, the program was adapted to include notes that are part of chords and to consider measures that repeat multiple times. Secondly, the start and end beat calculated by the program for each note event were converted to the absolute onset and offset time, respectively, of the note event in the corresponding audio file. The onset time

[3] `http://www.music.informatics.indiana.edu/code/musicxml`

t_{onset} in seconds is calculated using the formula

$$t_{onset} = 60\frac{b_{onset}}{bpm} + 0.025, \tag{4.1}$$

where b_{onset} is the start beat of a note event, bpm is the constant tempo of the music score in beats per minute, and the constant 0.025 is the amount of silence that Guitar Pro prepends to each synthesized audio file. The offset time t_{offset} in seconds is calculated by substituting the end beat b_{offset} of a note event for b_{onset} in Equation 4.1. Finally, a text file is generated for each symbolic music file processed by the modified MusicXML parsing program. The text file contains a list of the onset times, offset times, and pitches of note events occurring in the corresponding synthesized guitar recording.

Performing the described process on a set of Guitar Pro files results in a set of audio files and a set of text files. The generated text files along with their corresponding synthesized guitar recordings form the ground-truth dataset for polyphonic guitar transcription.

Dataset Details

Using the dataset creation process presented in the previous section, a ground-truth dataset for polyphonic guitar transcription has been created. 75 Guitar Pro files were selected from the *Ultimate Guitar* Top 100 list[4] and from the *Ultimate Guitar* Fresh Tabs list[5]. The *Ultimate Guitar* Top 100 list sorts every Guitar Pro file uploaded to the website by its rating—from one to five stars—and displays the top 100 files. The *Ultimate Guitar* Fresh Tabs list is a catalogue of Guitar Pro files that have recently been uploaded to the website and is sorted by number of hits (views). Only uploaded tablature with a five-star rating agreed upon by at least ten unique users was considered for selection. Tablature was selected on the basis of its musical genre, average degree of polyphony, and tempo, in order to accumulate a set of pieces with a variety of different attributes. The majority of Guitar Pro files were drawn from the *Ultimate Guitar* Top 100 list. Using the technique described in the previous section, a ground-truth file was created for each selected Guitar Pro file.

The collected Guitar Pro files were then synthesized using the clean guitar and distortion guitar presets in Guitar Pro. If the guitar track was originally intended to be performed by an acoustic guitar, a Martin & Co. acoustic guitar with steel strings was used as the guitar

[4] `http://www.ultimate-guitar.com/top/?rating&filter=pro`
[5] `http://www.ultimate-guitar.com/tabs/index_guitar_pro.htm`

model instead of the guitar model assigned by the preset. It is hoped that providing a set of distortion guitar recordings in the ground-truth dataset will stimulate the development of polyphonic guitar transcription algorithms that are tuned to perform well on distortion guitar recordings, since the distortion guitar effect is popular in many musical genres.

A detailed overview of the compiled ground-truth dataset is presented in Table C.1 of Appendix C. The dataset consists of 75 isolated guitar tracks; 125,192 note events; 30,914 chords; an average polyphony of 2.16; and an average tempo of 112 beats per minute. The number of note events was calculated by counting the number of note elements occurring in the 75 MusicXML representations of the selected Guitar Pro files. Similarly, the number of chords was calculated by counting the number of chord elements occurring in the MusicXML files. The average polyphony was calculated by dividing the number of note events by the number of chords plus the number of note elements that are not part of a chord. The average tempo was calculated by adding the tempos encoded in the metadata of the MusicXML files and dividing by the number of pieces in the dataset. There are approximately five and a half hours of clean guitar recordings and five and a half hours of distortion guitar recordings, yielding approximately eleven hours of audio in total. The musical genre for each song was assigned by considering the genre of the artist on `www.wikipedia.org`. The distribution of the genre of songs in the ground-truth dataset is illustrated in Figure 4.2. The role of the isolated guitar track (lead or rhythm) was subjectively chosen after listening to the track in its entirety.

4.1.2 Guitar Tablature Arrangement Dataset

In the evaluation of guitar tablature arrangement algorithms, a ground-truth dataset should contain a set of symbolic music scores that encode a sequence of notes or chords along with an appropriate string and fret combination for each note in the score. Similar to other problems in the field of MIR such as mood classification, genre classification, or structural segmentation where the ground-truth annotations are open for interpretation, the ground-truth tablature for a music score is also open for interpretation. There are often several solutions to the tablature arrangement problem and the adequacy of any given arrangement is also subject to the stylistic preferences of the performer.

Similar to the problem of polyphonic guitar transcription, there exists no ubiquitous dataset for training or evaluating guitar tablature arrangement algorithms. Previous guitar

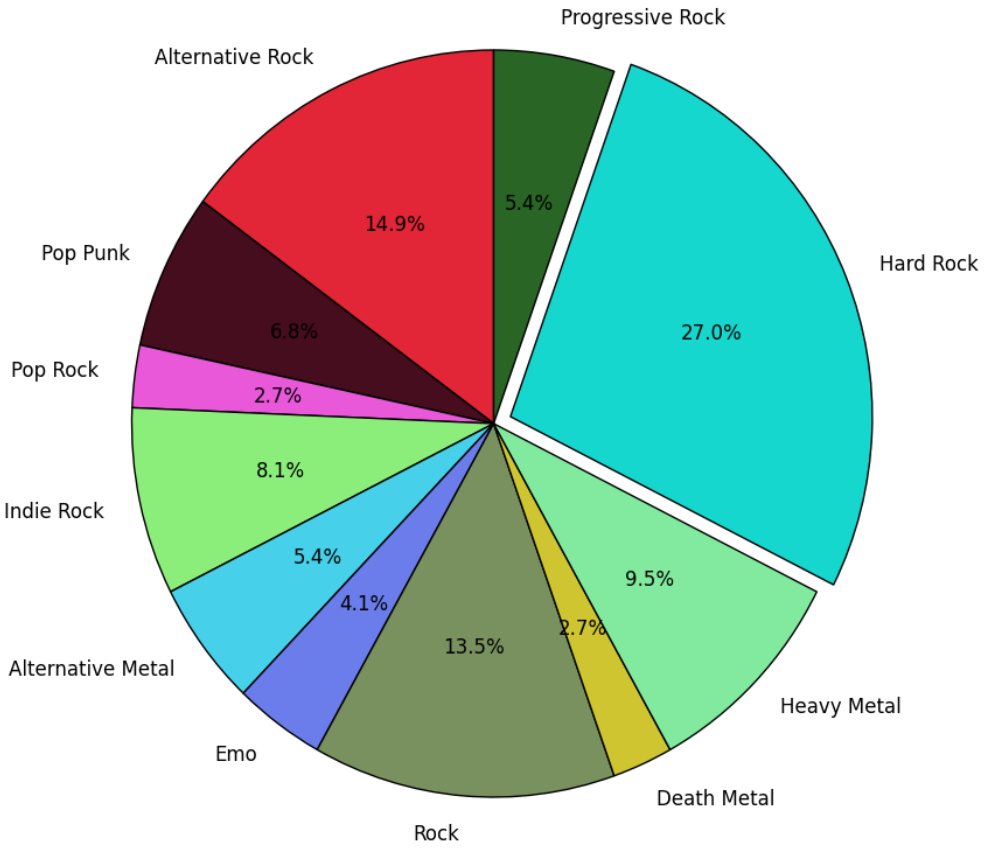


Fig. 4.2: Distribution of the genre of pieces in the compiled ground-truth dataset for polyphonic guitar transcription and guitar tablature arrangement.

tablature arrangement algorithms have been evaluated using selected excerpts from music scores that have been hand-annotated by trained guitarists (Radicioni et al. 2004; Radicioni and Lombardo 2005a; Radicioni and Lombardo 2005b; Rutherford 2009) or using selected excerpts from published tablature (Radisavljevic and Driessen 2004; Tuohy and Potter 2005; Sawayama et al. 2006; Tuohy and Potter 2006c). The largest evaluation dataset for a guitar tablature arrangement algorithm presented in the literature thus far has consisted of selected excerpts from 75 user-uploaded classical guitar tablatures obtained from `www.classtab.org` (Tuohy and Potter 2006a)—a relatively large dataset, though it is biased toward a single genre and the tablature is encoded in plain text format.

This section presents a new ground-truth dataset for guitar tablature arrangement. In comparison to the dataset presented by Tuohy and Potter (2006a), the dataset compiled

in this work incorporates a wider variety of musical genres and provides tablature in a standard symbolic music encoding format.

Dataset Details

The ground-truth dataset for guitar tablature arrangement was compiled using the same 75 Guitar Pro files that were collected and preprocessed for the polyphonic guitar transcription dataset presented in Section 4.1.1. Following previous research, which evaluates guitar tablature arrangement algorithms on selected excerpts of tablature (Radisavljevic and Driessen 2004; Tuohy and Potter 2006a), excerpts were selected from the Guitar Pro files. Excerpts were selected on the basis of overall length, the number of times the excerpt occurred throughout the entire piece, and the average polyphony of the excerpt. In general, musical motifs—salient and recurring guitar riffs—were selected from each piece because they were relatively short (no one excerpt exceeds eight measures) and they frequently recur throughout the music score. Excerpts were also selected on the basis of their average polyphony to ensure that the average polyphony of the resulting guitar tablature arrangement dataset was close to the average polyphony of the polyphonic transcription dataset. Each excerpt in Guitar Pro was exported as a MusicXML file using the Guitar Pro desktop application, which was subsequently converted to an MEI file.

An overview of the compiled dataset is presented in Table C.3 of Appendix C. The ground-truth dataset for guitar tablature arrangement consists of 75 tablature arrangements that are encoded in both the MEI and MusicXML symbolic music file formats. There are 4,845 notes; 1,143 chords; and an average polyphony of 1.94. The number of notes, chords, and the average polyphony of each symbolic music file in the ground-truth dataset was calculated using the same technique described in Section 4.1.1 for the polyphonic guitar transcription dataset. The distribution of genres for the songs in the dataset is illustrated in Figure 4.2.

4.2 Polyphonic Transcription Evaluation

The purpose of the evaluation proposed in this section is to determine the quality of transcriptions produced by the implemented polyphonic transcription algorithm (Zhou and Reiss 2008) on guitar recordings. The implemented polyphonic transcription algorithm

was evaluated in the 2008 MIREX competition on a piano dataset consisting of ten 30-second Yamaha Disklavier recordings. The algorithm received favourable results when considering the fundamental frequency and onset time of transcribed notes (Table 2.1): 0.738 precision, 0.777 recall, and an f-measure of 0.757. Although the parameters of the transcription algorithm have been hand tuned to perform well on a dataset composed of polyphonic piano and guitar recordings (Zhou and Reiss 2008), which was compiled by mixing monophonic recordings from the RWC Musical Instrument Sound Data Base (Zhou et al. 2009), the algorithm has yet to be evaluated on polyphonic guitar recordings.

Using an evaluation procedure similar to MIREX, the implemented polyphonic transcription algorithm will be evaluated using the compiled ground-truth dataset for polyphonic guitar transcription. The MIREX evaluation procedure is used because it provides a standardized method of evaluating polyphonic transcription algorithms, reports standard statistical metrics regarding the performance of an algorithm, and is established in the MIR community. MIREX evaluates polyphonic transcription algorithms by comparing the estimated onset times, offset times, and fundamental frequencies of the note events in an input audio recording to the ground-truth note events using the metrics of precision, recall, and f-measure. Before reviewing these metrics, it is important to establish the conditions that MIREX imposes on an estimated note event for it to be considered a correctly transcribed note. An estimated note event is deemed to be correctly transcribed if the fundamental frequency is within half a semitone of the ground-truth note event, the onset time is within a 50-millisecond range of the ground-truth note event, and the offset time is within 20% of the duration of the ground-truth note event. MIREX also reports the results of the polyphonic transcription algorithms when the offset time of estimated note events are disregarded and also when octave errors are disregarded. When octave errors are disregarded, an estimated note event is deemed to be correctly transcribed if the fundamental frequency is within half a semitone of integer multiples of the ground-truth fundamental frequency.

To review, precision $p \in \{\mathbb{R} : 0 \leq p \leq 1\}$ describes the ratio of correctly transcribed note events to the total number of estimated note events. Formally, precision is calculated using the formula

$$p = \frac{|G \cap E|}{|E|}, \tag{4.2}$$

such that G is the set of ground-truth note events and E is the set of estimated note events. The set intersection of G and E is the set of correctly transcribed note events. Equivalently,

precision may also be calculated using the formula

$$p = \frac{tp}{tp + fp}, \tag{4.3}$$

such that tp (true positive) is the number of correctly transcribed notes and fp (false positive)[6] is the number of estimated notes that are not present in the ground truth. An excessive number of false positives suggests that a more conservative threshold for note onset estimation should be considered.

Recall $r \in \{\mathbb{R} : 0 \leq r \leq 1\}$ describes the ratio of correctly transcribed note events to the total number of ground-truth note events. Formally, recall is calculated using the formula

$$r = \frac{|G \cap E|}{|G|}, \tag{4.4}$$

such that G is the set of ground-truth note events and E is the set of estimated note events. Equivalently, recall may be calculated using the formula

$$r = \frac{tp}{tp + fn}, \tag{4.5}$$

such that tp is the number of correctly transcribed notes and fn (false negative)[7] is the number of ground-truth note events that have not been correctly transcribed. A false negative can result from an onset estimation error, an offset estimation error, or an error in the fundamental frequency estimation of a note event.

Precision and recall are symbiotic statistics that should be conjunctively interpreted to assess the performance of a polyphonic transcription algorithm. If recall is the only metric considered, a polyphonic transcription algorithm that outputs every possible pitch at regular intervals would receive 100% recall even though the generated transcription is far from the desired transcription. If precision is the only metric considered, a polyphonic transcription algorithm that outputs a single correct note event would receive 100% precision, although the generated transcription is again far from the desired transcription. The f-measure statistic seeks to combine precision and recall into a single metric. The traditional f-measure (Equation 2.3) weights precision and recall equally.

[6] Using terminology from the field of statistics, a false positive is referred to as a type I error.
[7] Using terminology from the field of statistics, a false negative is referred to as a type II error.

Using a similar evaluation methodology as MIREX, two experiments will be conducted. The first experiment will report the precision, recall, and f-measure of the polyphonic transcription module on each guitar recording when the offset time of estimated note events are disregarded. The second experiment will report the precision, recall, and f-measure of the polyphonic transcription module on each guitar recording when both the offset time and the octave of each estimated note event are disregarded. Similar to MIREX, the onset time of an estimated note event is deemed correct if it is within a 50-millisecond range of the onset time of the ground-truth note event. As a subtle difference to the MIREX evaluation procedure, estimated fundamental frequencies are quantized to the nearest pitch so that they may be compared to the pitch of note events in the ground-truth file. The pitch of an estimated note event is correct if both the pitch name and the octave are the same as the corresponding ground-truth pitch. In the case of the second experiment where the octave of a note is disregarded, the pitch of an estimated note event is deemed correct if the pitch name is equal to the pitch name of the ground-truth note event. For both experiments, the window and hop size of the polyphonic transcription algorithm will be set to the default value of 441 samples, as it was in the 2008 MIREX evaluation (Zhou and Reiss 2008).

In each evaluation experiment the algorithm will be embedded in the polyphonic transcription module of Robotaba in order to prune estimated note events that exceed the maximum chord polyphony of six and are outside of the pitch range of the guitar model (Section 3.1.2). The guitar model is constructed by assuming a 24-fret guitar and the tuning and capo position are retrieved from the symbolic music file corresponding to the guitar recording being processed.

The proposed experiments seek to confirm or refute several hypotheses about the performance of the polyphonic transcription algorithm on guitar recordings. The first hypothesis is that the precision and recall of the polyphonic transcription algorithm will be less than that reported in the 2008 MIREX evaluation of the algorithm on the piano dataset. The reasoning behind this hypothesis is twofold. First, the recordings in the compiled ground-truth dataset exhibit a variety of ornamentation such as pitch bends, slides, palm muting, dead notes, hammer-ons, pull-offs, and right-hand tapping (see Appendix A for a description of these techniques). Pitch bends are especially problematic for fundamental frequency estimation algorithms, since the pitch of a note event in the ground truth may be encoded several semitones lower than its sounding pitch, which gradually rises over time. Notes performed by hammer-ons, pull-offs, and tapping are especially problematic for onset esti-

mation algorithms, since the amplitude of the attack of these notes is less than a plucked note (Ozaslan et al. 2010). Second, the MIREX piano dataset contains a total of five minutes of audio, whereas the compiled ground-truth dataset contains approximately five and a half hours of audio of various genres, lending more variability to the audio being transcribed.

The second hypothesis is that the average precision and recall of the polyphonic transcription algorithm on the distortion guitar recordings will be less than the average precision and recall of the polyphonic transcription algorithm on the clean guitar recordings. The reasoning behind this hypothesis is based on the properties of the distortion guitar effect, which was originally obtained by increasing the gain of vacuum tube amplifiers past normal operating limits. Since its inception, numerous digital signal processing methods have attempted to mimic this distortion effect (Dailey 2013, 179–205). The application of a distortion guitar effect to an audio signal results in the modification of the amplitude of harmonics in the frequency domain (Dailey 2013, 188). In the case of harmonic distortion, the amplitude of overtones at integer multiples of the input frequency are affected, whereas inharmonic distortion affects the amplitude of overtones at odd integer multiples of the input frequency. The modification of the frequency content of the guitar signal as a result of the application of a distortion audio effect could negatively influence the performance of the polyphonic transcription algorithm.

4.3 Guitar Tablature Arrangement Evaluation

There is no standardized method for evaluating guitar tablature arrangement algorithms as there is for polyphonic transcription algorithms, which use the MIREX evaluation model. However, two evaluation methods are predominantly used in the literature, which compare tablature generated by guitar tablature arrangement algorithms to human-arranged tablature. The most common method of comparison involves subjective evaluation, whereby the generated tablature arrangements and the ground-truth tablature arrangements are performed or analyzed by a guitarist, who comments on the comparative difficulty and style of each arrangement (Radicioni and Lombardo 2005b; Tuohy and Potter 2006b; Tuohy and Potter 2006a). Another commonly used method of comparison calculates the percentage of string and fret combinations in the generated tablature arrangements that are consistent with published tablature (Radisavljevic and Driessen 2004). As noted by Tuohy and Potter

(2005), the latter evaluation method is "inherently flawed" because "a tablature that differs from the published tablature by only one note could conceivably be unplayable while a tablature differing at every position could be just as playable".

In light of this point, tablature arrangements generated by DarwinTab will be assessed using the fitness function of the implemented GA described in Section 3.2.2. To review, the fitness function quantitatively assesses the biomechanical ease of performing a tablature arrangement according to the three biomechanical complexity factors proposed by Heijink and Meulenbroek (2002): the position of the left hand on the guitar neck, the distance the left hand must move to transition between notes, and the finger span required to perform chords. The fitness function (Equation 3.5) returns the fitness $f \in \mathbb{R}^+$ of a tablature arrangement, such that for any two tablature arrangements of a symbolic music score, the superior arrangement is that with the maximum fitness. As a result of assessing tablature arrangements using the fitness function, a generated tablature arrangement that diverges from the corresponding human-arranged tablature is not penalized if it is still biomechanically easy to perform.

In the evaluation of DarwinTab, guitar tablature arrangements will be generated for each symbolic music score in the ground-truth dataset and the fitness of each arrangement will be calculated. However, the fitness of a tablature arrangement is a number that is difficult to interpret without contextualization because it is derived from the biomechanical difficulty score of performing a tablature arrangement, which has many contributing factors (Equation 3.4).

Addressing this issue, two methods were considered to contextualize the fitness values of the generated tablature arrangements. First, the calculated fitness value for each generated tablature arrangement could be compared to the fitness value of the ground-truth tablature arrangement as well as a different tablature arrangement of the same piece that received a lower user-rating on `www.ultimate-guitar.com`. If the fitness value of the generated tablature arrangement lies within the fitness range of the low-rated and high-rated human-arranged tablature, it can be concluded that the guitar tablature arrangement algorithm generates tablature arrangements that are of equivalent quality as those generated by humans. This method was attempted, but was discarded due to the variability in tablature transcriptions of the same song (e.g., inserted or deleted notes), and the lack of alternate tablature arrangements for particular pieces. Second, the calculated fitness value for each generated tablature arrangement could be normalized with respect to the fitness

of the ground-truth tablature arrangement. With this transformation, the normalized fitness $f' \in \mathbb{R}^+$ is interpreted as the biomechanical ease of performing a generated tablature arrangement relative to the ground-truth tablature arrangement. Specifically, $f' = 1$ indicates that the generated tablature arrangement is of equal performance difficulty as the ground-truth arrangement; $f' > 1$ indicates that the generated tablature arrangement is relatively easier to perform than the ground-truth arrangement; and $f' < 1$ indicates that the ground-truth tablature arranged by a human is easier to perform than the automatically generated tablature.

Using the second method described, the normalized fitness value will be calculated for each tablature arrangement generated by DarwinTab on the symbolic music scores in the ground-truth dataset. Given that the tablature arrangements in the ground-truth dataset are created by humans[8] and have a five-star rating agreed upon by at least ten unique users on `www.ultimate-guitar.com`, the fitness values of the ground-truth tablature arrangements are a sufficient standard to compare to.

DarwinTab has many different parameters that control the underlying GA: the population size n_{pop}, the number of generations n_{gen}, the number of crossover points n_x, the probability of individuals mating p_{mate}, and the probability of gene mutation p_{mutate}. Ideally, numerous parameter configurations would be tested and the combination of parameters that result in tablature arrangements with an equal performance difficulty as the human-arranged tablature would be selected. However, certain parameter configurations drastically increase the computation time required by DarwinTab to generate tablature arrangements for the pieces in the ground-truth dataset. For this reason, three experiments will be conducted to evaluate DarwinTab with different parameter combinations. Each experiment consists of producing a single guitar tablature arrangement for each symbolic music score in the ground-truth dataset and calculating the normalized fitness values of the resulting tablature arrangements.

Outlined in Table 4.1 are the parameter combinations used in each experiment. The parameters were chosen based on preliminary experiments on individual symbolic music scores sampled from the ground-truth dataset. The preliminary experiments involved generating tablature arrangements for pieces with a low average polyphony and pieces with a high average polyphony using a variety of different parameter configurations. These

[8] According to the tablature submission requirements on `www.ultimate-guitar.com`, "[the] tab must be ear-transcribed (you listen to the song, then tab out how you think it is played)".

experiments showed that the selected population size had a significant impact on the resulting normalized fitness values. Therefore, the three experiments cover a wide range of population sizes. In the preliminary experiments, the fitness value of the most elite chromosome in the population was observed for each generation. The number of generations parameter was selected based on the average generation that the fitness of the elite chromosome stabilized. Additional generations were added to account for the stochastic nature of the GA, which causes fluctuations in the number of generations required for converging on a solution. The number of crossover points parameter was selected by considering the average length of symbolic music scores in the ground-truth dataset. The probability of mating parents was set quite high to promote the rapid evolution of the population. The probability of mutation was selected to be around 0.03, as proposed by Tuohy and Potter (2006b).

Table 4.1: DarwinTab parameters for each evaluation experiment.

Experiment	n_{pop}	n_{gen}	n_x	p_{mate}	p_{mutate}
1	500	250	4	0.9	0.03
2	2000	200	4	0.8	0.04
3	4000	200	4	0.85	0.035

Tablature arrangements generated by DarwinTab will also be compared to those generated by two commercial algorithms provided by the Guitar Pro and Sibelius desktop applications. Both arrangement algorithms are closed source; though, after experimentation with several sample music scores, the algorithms seem to produce tablature arrangements deterministically. Using these commercial algorithms, a tablature arrangement will be generated for each symbolic music score in the ground-truth dataset. The fitness of the resulting tablature arrangements will be computed and normalized with respect to the fitness of the ground-truth tablature arrangements.

Chapter 5

Results and Discussion

The previous chapter presented the compiled ground-truth datasets for polyphonic guitar transcription (Section 4.1.1) and guitar tablature arrangement (Section 4.1.2). Several experiments were proposed to evaluate the implemented polyphonic transcription algorithm and the implemented guitar tablature arrangement algorithm on the compiled ground-truth datasets.

The experiments proposed in the previous chapter have been conducted and the results will be reported in this chapter, followed by a discussion of the results. Specifically, Section 5.1 will present the experimental results of the polyphonic transcription algorithm on the compiled ground-truth dataset. Several conjectures were made regarding the results of these experiments, which will be addressed through an analysis of the results. Section 5.2 will present the experimental results of the guitar tablature arrangement algorithm and compare the generated tablature arrangements to tablature arranged by humans and tablature arranged by the commercial reference algorithms provided by *Guitar Pro* and *Sibelius*.

5.1 Polyphonic Transcription Evaluation

To review, four experiments have been performed to evaluate the implemented polyphonic transcription algorithm on the guitar recordings in the ground-truth dataset:

Experiment 1

Calculate the precision, recall, and f-measure of the polyphonic transcription algo-

rithm on the clean guitar recordings in the ground-truth dataset, considering the accuracy of note pitch and onset time only.

Experiment 2

Calculate the precision, recall, and f-measure of the polyphonic transcription algorithm on the clean guitar recordings in the ground-truth dataset, considering the accuracy of note onset time and pitch name only. Octave errors are ignored.

Experiment 3

Calculate the precision, recall, and f-measure of the polyphonic transcription algorithm on the distortion guitar recordings in the ground-truth dataset, considering the accuracy of note pitch and onset time only.

Experiment 4

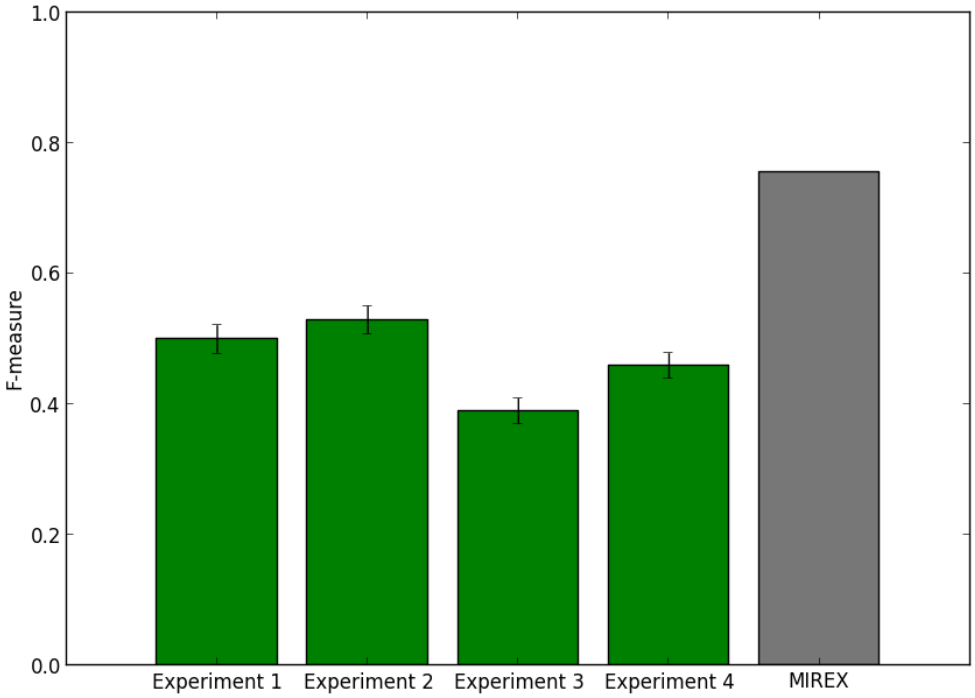
Calculate the precision, recall, and f-measure of the polyphonic transcription algorithm on the distortion guitar recordings in the ground-truth dataset, considering the accuracy of note onset time and pitch name only. Octave errors are ignored.

Similar to the MIREX experiments, the onset time of a note event is considered acceptable if it lies within 50 milliseconds of the onset time of the corresponding ground-truth note event. For each experiment the two parameters of the polyphonic transcription algorithm—window size and hop size—are both set to the default value of 441 samples (Zhou and Reiss 2008).

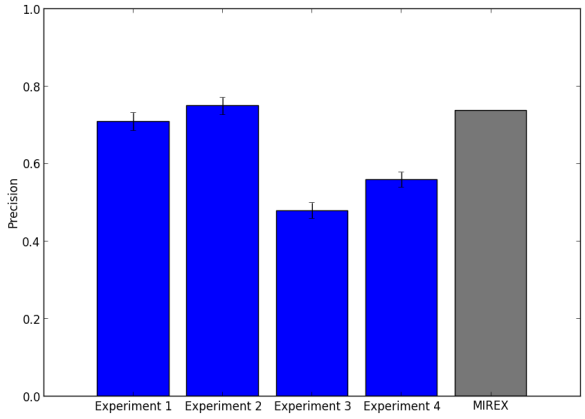
5.1.1 Results

The results of each experiment are presented in Table C.2 of Appendix C. The precision, recall, and f-measure of the polyphonic transcription algorithm on each guitar recording in the ground-truth dataset is reported. The ID field in Table C.2 references the ID field in Table C.1 of the appendix, which provides metadata for each piece of music. For each experiment, the average f-measure is displayed in Figure 5.1(a), the average precision in Figure 5.1(b), and the average recall in Figure 5.1(c).

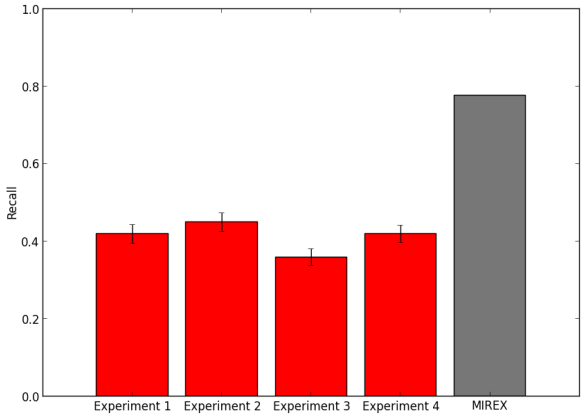
Another result to report is the performance of the polyphonic transcription algorithm on pieces with different degrees of polyphony. To produce this result, the ground-truth dataset was partitioned into five groups: the first partition contained 39 pieces, each with



(a) Average f-measure of the polyphonic transcription algorithm.



(b) Average precision of the polyphonic transcription algorithm.



(c) Average recall of the polyphonic transcription algorithm.

Fig. 5.1: Average metrics across the ground-truth dataset of each of the four experiments conducted to evaluate the implemented polyphonic transcription algorithm. Also displayed are the results of the algorithm on the piano dataset reported by MIREX, which considers the pitch and onset time of note events (similar to Experiment 1).

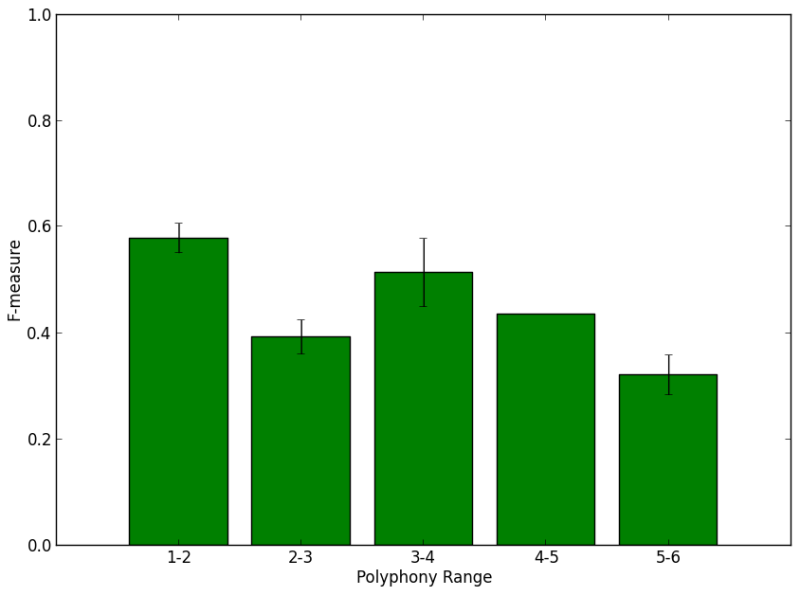


Fig. 5.2: Average f-measure of the polyphonic transcription algorithm, considering note pitch and onset time, across clean guitar recordings in the ground-truth dataset partitioned by average polyphony. There is no error bar for polyphony range 4–5 because only one piece is in this range.

an average polyphony $\in [1, 2)$; the second partition contained 24 pieces, each with an average polyphony $\in [2, 3)$; the third partition contained nine pieces, each with an average polyphony $\in [3, 4)$; the fourth partition contained one piece having an average polyphony $\in [4, 5)$; and the fifth partition contained two pieces, each with an average polyphony $\in [5, 6]$. Figure 5.2 displays the average f-measure of the polyphonic transcription algorithm, considering note pitch and onset time, on the clean guitar recordings in each of the six partitions of the ground-truth dataset.

Yet another result to report is the performance of the polyphonic transcription algorithm on pieces with different genres. To produce this result, the ground-truth dataset was partitioned into eleven genre groups: pop punk, emo, rock, heavy metal, hard rock, progressive rock, alternative rock, pop rock, indie rock, alternative metal, and death metal. The distribution of the pieces in the dataset with these genres is presented in the previous chapter (Figure 4.2). Figure 5.3 presents the average f-measure of the polyphonic transcription algorithm, considering the accuracy of note pitch and onset time, on the clean guitar recordings in the compiled ground-truth dataset partitioned by genre.

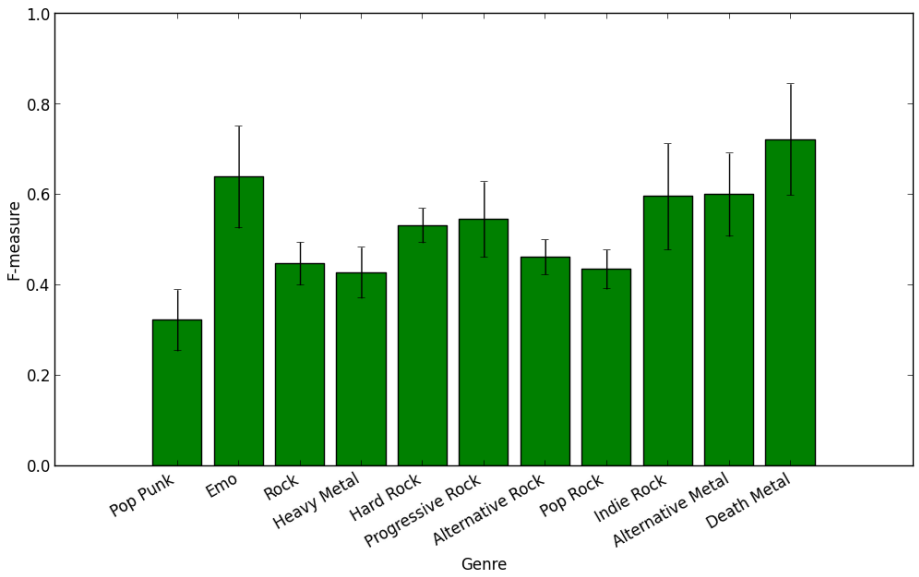


Fig. 5.3: Average f-measure of the polyphonic transcription algorithm, considering note pitch and onset time, across clean guitar recordings in the ground-truth dataset partitioned by genre.

5.1.2 Discussion

In the previous chapter, two hypotheses were made regarding the performance of the polyphonic transcription algorithm on the compiled ground-truth dataset. The first hypothesis postulates that the polyphonic transcription algorithm will perform worse on the compiled dataset of clean guitar recordings in comparison to the performance of the algorithm on the MIREX piano dataset. The second hypothesis posits that the polyphonic transcription algorithm will perform better on clean guitar recordings than guitar recordings with a distortion audio effect applied. An analysis of the experimental results presented in this section will address these hypotheses and comment on additional findings.

To address the first hypothesis, the results of the first experiment are compared to the results of the algorithm reported by MIREX on the MIREX piano dataset. Both evaluation experiments consider the accuracy of note pitch and onset time, while disregarding note duration, on audio recordings with no audio effects applied. Referencing Figure 5.1, which presents the average precision, recall, and f-measure of Experiment 1 alongside the average precision, recall, and f-measure reported by MIREX, the algorithm in fact performed worse

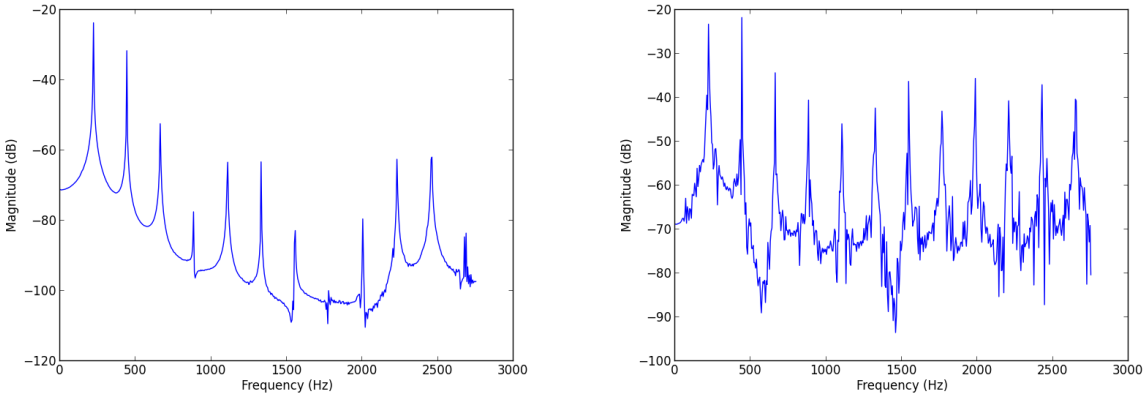
on the clean guitar recordings in the compiled ground-truth dataset in comparison to the performance of the algorithm on the MIREX piano dataset, with respect to all metrics.

One possible explanation for the degraded transcription performance is that the guitar recordings in the compiled ground-truth dataset contain ornamentation such as slides, bends, hammer-ons, hammer-offs, palm-muting, and dead notes (see Appendix A for a description of these guitar techniques). The piano is incapable of replicating many of these ornaments; for example, when processing piano recordings the polyphonic transcription algorithm is never presented notes with fluctuating pitch because a pianist can not bend the piano strings as a guitarist can bend the guitar strings. An ancillary experiment that evaluates the transcription algorithm on synthesized audio recordings of the symbolic music scores in the ground-truth dataset with the ornamentation removed could provide a definitive answer to this question.

Another possible explanation for this result is that the dataset used in this thesis contains different pieces than the MIREX piano dataset. The compiled dataset is substantially larger, with approximately five and a half hours of audio versus the five minutes of audio in the MIREX piano dataset. Furthermore, the genre of the pieces in the guitar dataset are primarily rock, metal, and their derivative genres, whereas the genre of the pieces in the MIREX piano dataset are classical, containing pieces such as Ludwig van Beethoven's Piano Sonata No. 8 in C minor and Wolfgang Mozart's Piano Sonata No. 13 in B-flat major. The fact that the dataset used in this thesis consists of different pieces than the MIREX piano dataset prevents drawing the conclusion that the evaluated polyphonic transcription algorithm performs better on piano recordings than guitar recordings.

To address the second hypothesis, the results of Experiment 1 and Experiment 2, which process the clean guitar recordings in the ground-truth dataset, are compared to the results of Experiment 3 and Experiment 4, which process the guitar recordings with a distortion audio effect applied. Confirming the hypothesis that the application of a distortion audio effect to guitar recordings will degrade automatic polyphonic transcription performance, the precision, recall, and f-measure of Experiment 1 is greater than that of Experiment 3 and the precision, recall, and f-measure of Experiment 2 is greater than that of Experiment 4, as presented in Figure 5.1. When forming this hypothesis in the previous chapter, it was speculated that a contributing factor to the reduced transcription performance could be the modification of the relative amplitudes of harmonics in the frequency domain of the guitar signal caused by the distortion audio effect. Figure 5.4 displays the frequency

domain of a guitar pluck of the note A3 synthesized using *Guitar Pro* with and without a distortion audio effect applied. This figure provides evidence that the relative amplitudes of harmonics are in fact modified when a distortion audio effect is applied—a factor that may have contributed to the reduced transcription performance on the distortion guitar recordings.



(a) Portion of the magnitude spectrum of a guitar signal without a distortion audio effect applied.

(b) Portion of the magnitude spectrum of a guitar signal with a distortion audio effect applied.

Fig. 5.4: Comparison of the magnitude spectrum of a guitar pluck of the note A3 synthesized with and without a distortion audio effect applied.

Several other interesting conclusions can be drawn from the experimental results of the polyphonic transcription algorithm. Observing Figure 5.1(b) and Figure 5.1(c), the recall reported by each experiment is strictly less than the precision, whereas the MIREX results of the algorithm on the piano dataset report the recall (0.777) to be slightly greater than the precision (0.738). Having high precision and low recall means that many notes were missed; however, of the notes that were estimated, they were quite accurate. This result suggests that the evaluated polyphonic transcription algorithm has a note onset detection algorithm that is too conservative, at least for the pieces evaluated in the compiled dataset.

As one might expect, the precision, recall, and f-measure of the polyphonic transcription algorithm strictly increases when disregarding note octave errors. Observing Figure 5.1, the precision, recall, and f-measure of Experiment 1 is strictly less than that of Experiment 2, and the precision, recall, and f-measure of Experiment 3 is strictly less than that of Experiment 4. An interesting observation is that there are marginally more octave errors in

transcriptions of distortion guitar recordings than transcriptions of clean guitar recordings. Figure 5.1(a) shows a 3% increase in f-measure between Experiment 1 and Experiment 2, whereas there is a 7% increase in f-measure between Experiment 3 and Experiment 4.

In regard to the influence of polyphony on transcription performance, previous research in the area of automatic music transcription has noted decreased accuracy in multiple fundamental frequency estimation as the polyphony of the audio signal increases (Klapuri 2006). This inverse relationship between the accuracy of multiple fundamental frequency estimation and polyphony can be attributed to the increasingly convoluted mixture of signal information in both the time and frequency domain of the audio signal as the polyphony increases. In a similar fashion, Figure 5.2 displays a downward trend in f-measure as polyphony increases.

Considering the influence of genre on polyphonic transcription performance, Figure 5.3 displays the average f-measure of the polyphonic transcription algorithm across clean guitar recordings in the ground-truth dataset, partitioned by genre. The genre *pop punk*, characterized by its fast tempo, rapid guitar strumming, and frequently palm-muted guitar riffs, received the lowest f-measure of 0.32. On the contrary, the genre *death metal*, also characterized by its fast tempo and frequently palm-muted guitar riffs, received the highest f-measure of 0.72; however, only two pieces of this genre were present in the compiled dataset. Overall, the performance of the polyphonic transcription algorithm varies greatly across pieces of different genres, suggesting that genre is a factor influencing the performance of the polyphonic transcription algorithm. However, there may exist a spurious relationship between genre and transcription performance, which prevents the conclusion that genre directly influences transcription performance. For example, the tempo of a piece may be a confounding variable that affects the genre and the transcription performance. Moreover, specifying the genre of a piece is a highly subjective process, making it difficult to draw any conclusions regarding the influence of genre on transcription performance.

Another important step of the analysis process is investigating the input that an algorithm performs poorly on as well as the input that an algorithm performs well on. To provide insight into the possible attributes of a guitar recording that contribute to a poor transcription, the ten pieces in the ground-truth dataset that received an f-measure < 0.3 in Experiment 1 were grouped together and analyzed. These pieces either have an above average tempo, an above average degree of polyphony, or both. For example, the rhythm guitar recording of "Johnny B. Goode" by Chuck Berry (ID: 17 in Table C.2 and Table C.1)

is characterized by its fast strumming pattern and rapid chord changes, receiving the lowest f-measure of 0.11. On the other hand, the almost monophonic lead guitar recording of "Mr. Brightside" by The Killers (ID: 43 in Table C.2 and Table C.1), received the highest f-measure in Experiment 1 of 0.95. However, by counterexample it can not be concluded that guitar recordings with low polyphony will yield a transcription with high precision and recall. The lead guitar recording of the heavy-metal song "Unholy Confessions" by Avenged Sevenfold (ID: 6 in Table C.2 and Table C.1), characterized by its fast and often palm-muted guitar riffs, has a low average polyphony (1.24) yet received an f-measure of only 0.42. From this analysis, one can conclude that there are many factors that influence transcription performance.

5.2 Guitar Tablature Arrangement Evaluation

To review, three experiments have been conducted to evaluate the implemented guitar tablature arrangement algorithm *DarwinTab*:

Experiment 1

Calculate the normalized fitness of the guitar tablature arrangement generated by DarwinTab for each symbolic music score in the ground-truth dataset using the following parameters: $n_{pop} = 500$, $n_{gen} = 250$, $n_{nx} = 4$, $p_{mate} = 0.9$, $p_{mutate} = 0.03$.

Experiment 2

Calculate the normalized fitness of the guitar tablature arrangement generated by DarwinTab for each symbolic music score in the ground-truth dataset using the following parameters: $n_{pop} = 2000$, $n_{gen} = 200$, $n_{nx} = 4$, $p_{mate} = 0.8$, $p_{mutate} = 0.04$.

Experiment 3

Calculate the normalized fitness of the guitar tablature arrangement generated by DarwinTab for each symbolic music score in the ground-truth dataset using the following parameters: $n_{pop} = 4000$, $n_{gen} = 200$, $n_{nx} = 4$, $p_{mate} = 0.85$, $p_{mutate} = 0.035$.

5.2.1 Results

The normalized fitness values for each experiment are reported in Table C.4 alongside the normalized fitness values of tablature arrangements produced by the Guitar Pro and Sibelius reference algorithms for each symbolic music score in the ground-truth dataset.

Table 5.1 presents the median (denoted by $\hat{\mu}$) and standardized median absolute deviation (denoted by $\hat{\sigma}$) of the distribution of normalized fitness values for each algorithm. The median and standardized median absolute deviation are used to describe the central tendency and dispersion of the distribution, respectively. They are referred to as robust descriptive statistics because they are less sensitive to outliers present in small datasets in comparison to the sample mean and sample standard deviation. Robust descriptive statistics are used to compensate for the handful of outliers present in the experimental results. For example, the tablature arrangements of the pieces "Carry On Wayward Son" by Kansas (ID: 42 in Table C.4) and "Animal I have Become" by Three Days Grace (ID: 75 in Table C.4) generated by each algorithm received a normalized fitness value of 9.00 and 4.54, respectively, which appear to be abnormally high in comparison to the rest of the results.

Table 5.1: Statistics of the approximately normal distribution of normalized fitness values for the tablature generated by each tablature arrangement algorithm.

Statistic	Experiment 1	Experiment 2	Experiment 3	Guitar Pro	Sibelius
$\hat{\mu}$	0.78	1.01	1.09	1.03	1.00
$\hat{\sigma}$	0.46	0.57	0.49	0.11	0.33

The median normalized fitness $\hat{\mu}$ of each algorithm is calculated by sorting the normalized fitness values calculated for each generated tablature arrangement and selecting the value in the middle of this list. The standardized median absolute deviation $\hat{\sigma}$ of each algorithm is found by first calculating the absolute value of the residual of each normalized fitness value f_i' from the median normalized fitness $\hat{\mu}$. The median of these residuals is then calculated and multiplied by a scale factor κ. In mathematical terms,

$$\hat{\sigma} = \kappa \cdot \text{median}_i(|f_i' - \hat{\mu}|), \tag{5.1}$$

such that

$$\kappa = \frac{\int_{-\infty}^{\frac{3}{4}} e^{-\frac{t^2}{2}} dt}{\sqrt{2\pi}} \approx 1.4826 \tag{5.2}$$

for normally distributed data. By the central limit theorem, which states that a distribution becomes increasingly more Gaussian as the sample size increases (Wackerly et al. 2007, 370–7), it can be assumed that the distribution of normalized fitness values for each algorithm approximately follows a normal distribution.

Presenting the experimental results in a different way, Figure 5.5 displays a box-and-whisker plot of the normalized fitness values of the tablature arrangements generated by each algorithm. For each algorithm, a box-and-whisker is formed by partitioning the normalized fitness values into two groups at the median value. The median is then calculated for each resulting group to form four partitions called *quartiles*. The last value in each partition is given the value q_1, q_2, q_3, and q_4, respectively. The interquartile range $q_r = q_3 - q_1$ forms the box and the whiskers stretch out to the last value in the outer quartiles that are not outliers. A normalized fitness value is considered an outlier if $f_i' < q_1 - 1.5q_r$ or $f_i' > q_3 + 1.5q_r$, and is displayed as a cross symbol above or below the whiskers. The solid line between the interquartile range represents the median normalized fitness value.

Another result to report is the performance of the evaluated guitar tablature arrangement algorithms on music scores with different degrees of polyphony. To this end, the ground-truth dataset was partitioned into four groups: the first partition contained 47 pieces, each with an average polyphony $\in [1, 2)$; the second partition contained 16 pieces, each with an average polyphony $\in [2, 3)$; the third partition contained nine pieces, each with an average polyphony $\in [3, 4)$; and the fourth partition contained three pieces, each with an average polyphony $\in [5, 6]$. There were no excerpts in the ground-truth dataset with an average polyphony $\in [4, 5)$. Figure 5.6 presents the median normalized fitness of the tablature arrangements generated by DarwinTab (using the parameters described in Experiment 3), Guitar Pro, and Sibelius on the symbolic music scores in the ground-truth dataset partitioned by average polyphony. Notice that for each symbolic music score in the polyphony range 5–6, Guitar Pro generated tablature arrangements with a normalized fitness value of one. Therefore, no error bars are displayed in this case.

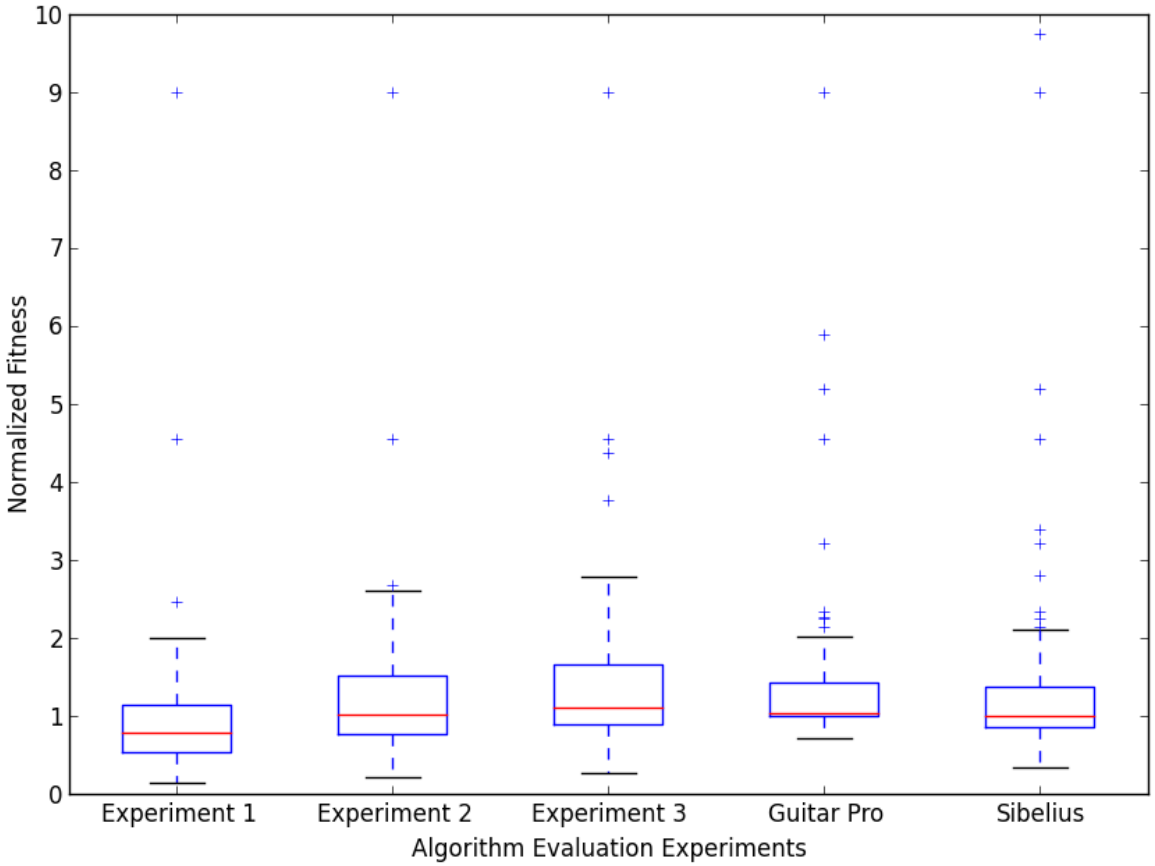


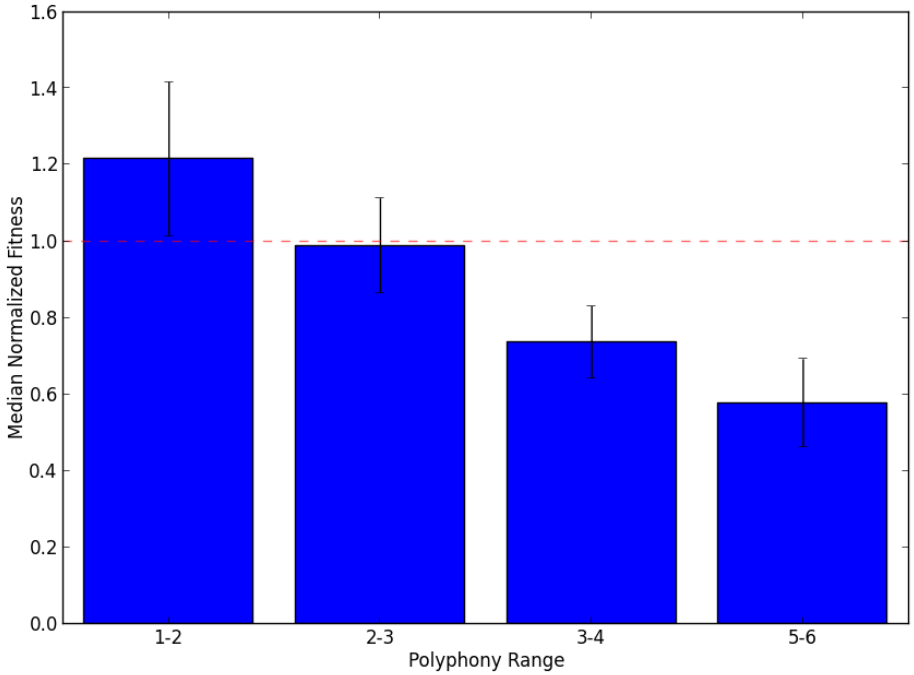
Fig. 5.5: Box-and-whisker plot of the normalized fitness values of the tablature arrangements generated by each evaluated tablature arrangement algorithm.

5.2.2 Discussion

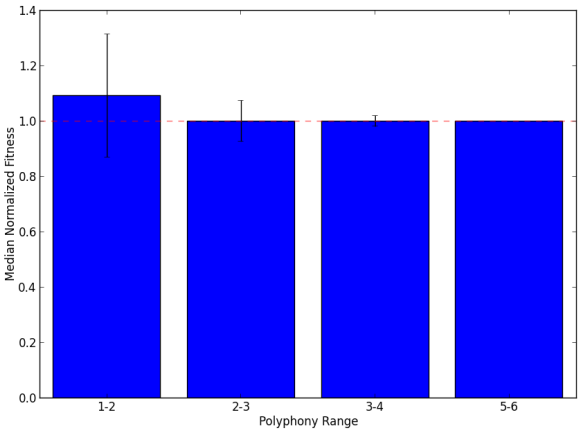
Several conclusions can be drawn from the presented experimental results. Using the parameter configuration of Experiment 3, DarwinTab generates guitar tablature arrangements that, on average, are of similar performance difficulty[1] as the ground-truth tablature arrangements as well as those produced by the reference algorithms. Table 5.1 reports a median normalized fitness of 1.09, 1.03, and 1.00 for DarwinTab, Guitar Pro, and Sibelius, respectively.

However, the variance in the performance difficulty of tablature arrangements generated by DarwinTab is larger than that of the reference algorithms (see Figure 5.5). In

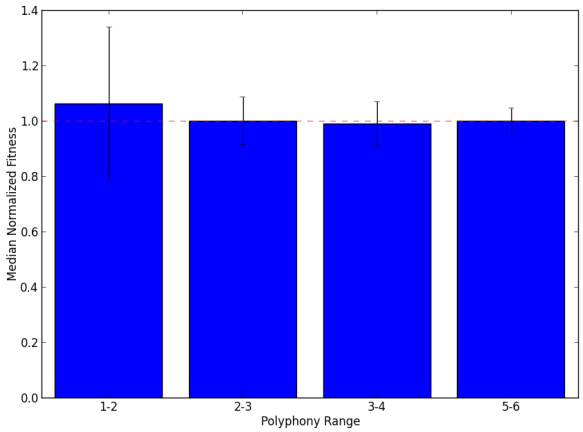
[1]The term *performance difficulty* refers to the biomechanical difficulty of performing a tablature arrangement, as assessed by the fitness function.



(a) Median normalized fitness of the implemented guitar tablature arrangement algorithm with the parameters described in Experiment 3.



(b) Median normalized fitness of the Guitar Pro tablature arrangement algorithm.



(c) Median normalized fitness of the Sibelius tablature arrangement algorithm.

Fig. 5.6: Median normalized fitness of the guitar tablature arrangement algorithm, alongside two commercial reference algorithms, across symbolic music scores in the ground-truth dataset partitioned by average polyphony. No symbolic music scores in the dataset had an average polyphony in the range 4–5.

Experiment 3, DarwinTab generated tablature arrangements with a median normalized fitness of 1.09 and a standardized median absolute deviation of 0.49 (Table 5.1). Therefore, DarwinTab generates tablature arrangements with normalized fitness values which approximately follow the Gaussian distribution $N(\hat{\mu} = 1.09, \hat{\sigma} = 0.49)$ displayed in Figure 5.7. By the empirical rule, 68.2% of tablature arrangements generated by DarwinTab are estimated to have normalized fitness values that lie within one standard deviation of the mean (0.60–1.58). Comparing this to the reference algorithms, Guitar Pro reported a median normalized fitness of 1.03 and a relatively small standardized median absolute deviation of 0.11 (Table 5.1). By the empirical rule, 68.2% of tablature arrangements generated by Guitar Pro are estimated to have normalized fitness values within the range 0.92–1.14. Sibelius reported a median normalized fitness of 1.00 and a median absolute deviation of 0.33 (Table 5.1). By the empirical rule, 68.2% of tablature arrangements generated by Sibelius are estimated to have normalized fitness values within the range 0.67–1.33.

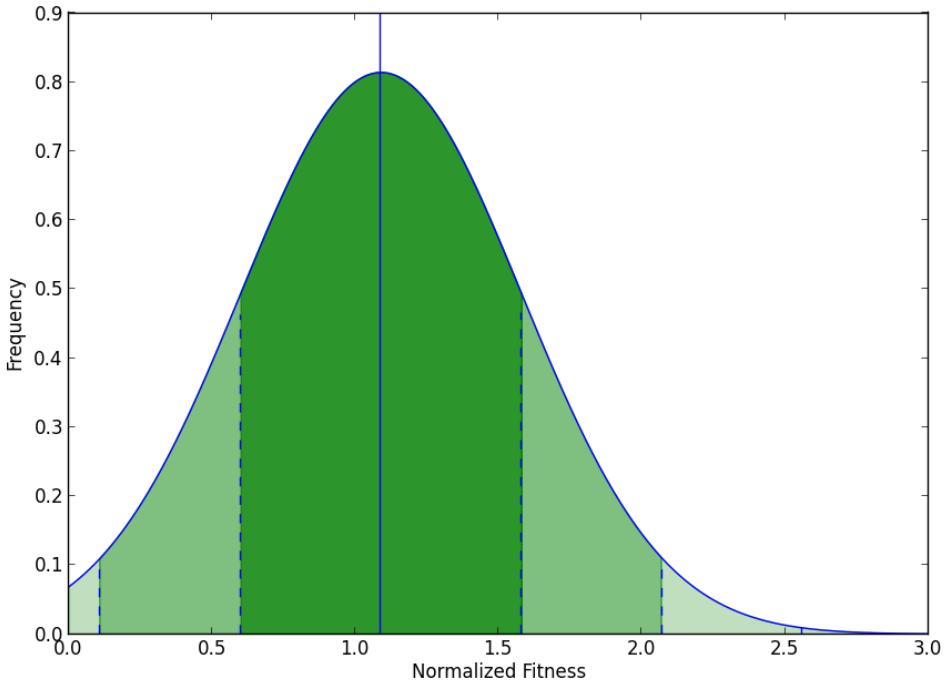
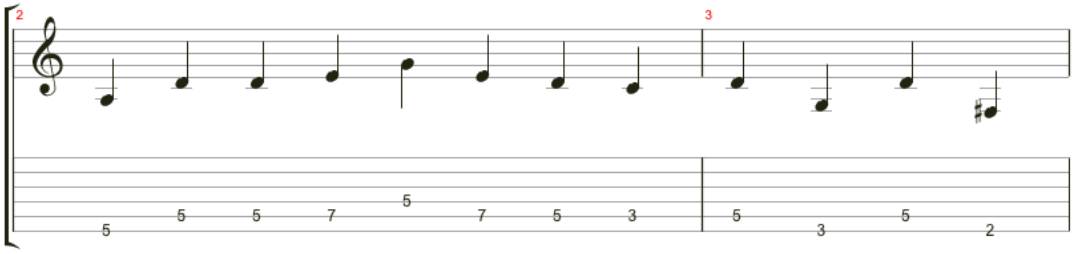


Fig. 5.7: Estimated Gaussian distribution $N(\hat{\mu} = 1.09, \hat{\sigma} = 0.49)$ of the normalized fitness values generated by DarwinTab in Experiment 3. The solid vertical line at $y = 1.09$ indicates the mean of the distribution. The dashed vertical lines at $y = 1.09 \pm 0.49k$ indicate the standard deviations.

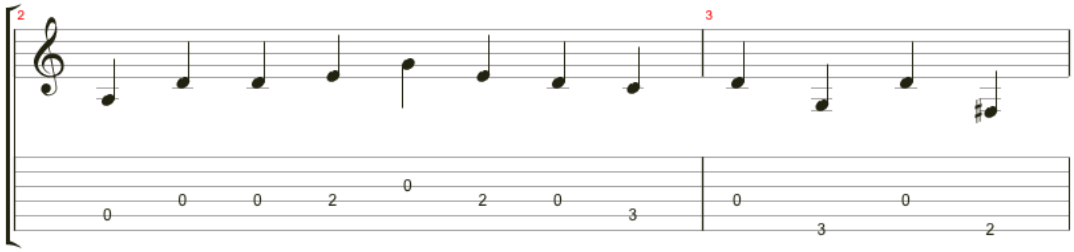
Another interesting result is the rise in the median normalized fitness from Experiment 1 to Experiment 3 (Table 5.1 and Figure 5.5), which signifies that the population size of the underlying GA in DarwinTab is an important factor in tablature arrangement performance. A population size of 500 in Experiment 1 yields a substandard median normalized fitness of 0.78. Increasing the population size to 2000 in Experiment 2 yields a very acceptable median normalized fitness of 1.01. Further increasing the population size to 4000 in Experiment 3 yields a marginal increase in median normalized fitness. However, the marginal increase in the median normalized fitness between Experiment 2 and Experiment 3 does not necessarily suggest that there is no benefit in further increasing the population size of the GA. Consider Figure 5.6(a), which shows a steady increase in the performance difficulty of the tablature arrangements generated by DarwinTab as the average degree of polyphony increases, whereas the reference algorithms consistently generate tablature arrangements with performance difficulties similar to the ground-truth arrangements. A potential explanation for this result is that the initial population of the GA does not provide enough genetic variability for scores with high degrees of polyphony. The results suggest that a population size of 4000 is sufficient for producing tablature arrangements for symbolic music scores with an average degree of polyphony below three, while the algorithm could benefit from a larger population size for scores with an average degree of polyphony above three.

Another result worth investigating is the normalized fitness of tablature arrangements generated by the evaluated algorithms that are considered outliers. Figure 5.5 shows that all of the outliers lie above the fourth quartile in the box plots. This means that either the corresponding ground-truth tablature arrangements are in fact biomechanically more difficult to perform or that, as propositioned by Heijink and Meulenbroek (2002), there are other factors contributing to a "good tab" apart from biomechanical difficulty. These factors might include rules that consider the musical context and rules that enforce certain timbral characteristics of plucked notes. Therefore, it is reasonable to assume that certain human-arranged tablatures sacrifice biomechanical ease of performance to accommodate the auditory or stylistic preferences of the arranger. This assumption is supported by the tablature arrangements generated for the piece "Carry On Wayward Son" by Kansas (ID: 42 in Table C.4). Each evaluated tablature arrangement algorithm converged to the same solution for this piece, receiving a normalized fitness value of 9.00. Figure 5.8 compares two measures of the generated tablature to the ground-truth tablature arrangement. In contrast

with the hand-arranged tablature, the generated tablature inserts open-string plucks wherever possible. When performed on the guitar, both tablatures have a subjectively equal difficulty level, though the timbre greatly differs. For notes with more than one candidate string and fret combination, the timbral characteristic of the performed note becomes a factor in the selection of the fretboard location (Tuohy and Potter 2005).



(a) Two measures of the ground-truth tablature arrangement of the piece "Carry on Wayward Son" by Kansas.

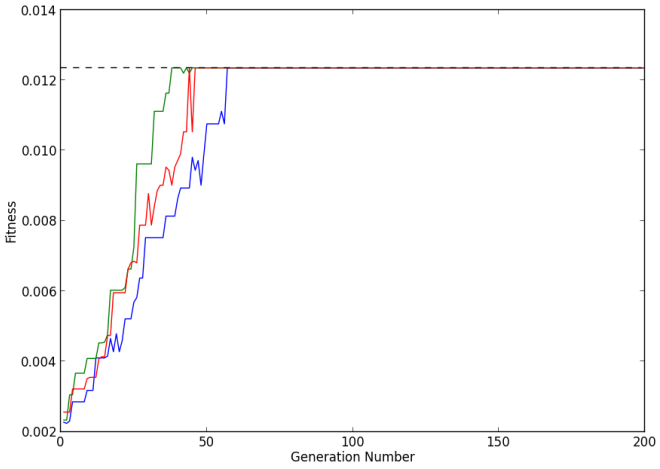


(b) Two measures of the tablature generated by DarwinTab, Guitar Pro, and Sibelius for the piece "Carry on Wayward Son" by Kansas.

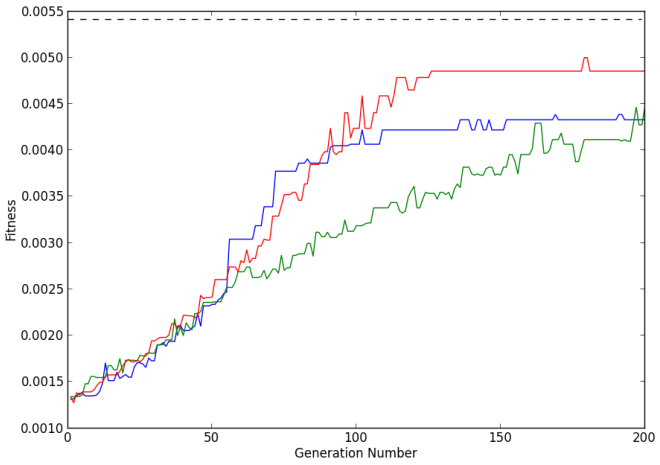
Fig. 5.8: Generated tablature arrangement compared to the hand-arranged tablature. The tablature is rendered using the AlphaTab digital tablature engraving library.

Another important issue surrounding the implemented guitar tablature arrangement algorithm is the computational runtime of the underlying GA. The runtime of Experiment 1 was approximately ten hours. The runtime of Experiment 2 was approximately 54 hours. The runtime of Experiment 3 was approximately 110 hours, meaning that the average time to produce a tablature arrangement was approximately one and a half hours. The experiments were run on a machine with a 2GHz CPU and 8GB of main memory. In comparison, the reference algorithms generated each tablature arrangement virtually instantaneously. The short runtime of these algorithms suggests that a neural network or a graph search algorithm with heuristics is used.

In practise, the runtime of DarwinTab could be reduced by specifying a GA termination condition, which would return the elite tablature arrangement in the population when the fitness value becomes stagnant over a sequence of generations. Figure 5.9 displays the fitness values of the elite tablature arrangement for two pieces—the first having a low average polyphony of 1.63 (Song ID: 24 in Table C.3), and the second having a high average polyphony of 5.00 (Song ID: 7 in Table C.3)—using the parameter configuration of Experiment 3 ($n_{pop} = 4000$, $n_{gen} = 200$, $n_{nx} = 4$, $p_{mate} = 0.85$, $p_{mutate} = 0.035$). Three runs of the GA were performed to account for the stochastic nature of the algorithm. In Figure 5.9(a), each run of the GA converged to the fitness of the ground-truth tablature arrangement around the 50^{th} generation. In Figure 5.9(b), the elite fitness value in two of the three runs stabilized around the 125^{th} generation. These results suggest that in certain cases the computational runtime of DarwinTab could be reduced by as much as 75% if the GA is terminated at the point of convergence.



(a) Fitness values of the elite tablature in the GA population over the course of 200 generations for the tablature arrangement of the piece "Tears in Heaven" by Eric Clapton.



(b) Fitness values of the elite tablature in the GA population over the course of 200 generations for the tablature arrangement of the piece "While My Guitar Gently Weeps" by The Beatles.

Fig. 5.9: Three runs of DarwinTab on two symbolic music scores using the parameter configurations of Experiment 3. The dashed line denotes the fitness of the ground-truth tablature arrangement.

Chapter 6

Conclusion

After introducing tablature notation in Chapter 1, this thesis presented an overview of several approaches to the problem of automatic guitar tablature transcription. Specifically, several polyphonic transcription algorithms (and surrounding schools of thought) that performed well in the MIREX evaluation suite were reviewed, followed by a review of guitar tablature arrangement algorithms. Chapter 3 introduced the implemented guitar tablature transcription framework, the implemented polyphonic transcription algorithm (Zhou and Reiss 2008), and the implemented guitar tablature arrangement algorithm entitled *DarwinTab*. Two new ground-truth datasets, gathered from manual transcriptions posted by users on `www.ultimate-guitar.com`, were compiled to evaluate the implemented algorithms. Chapter 4 described the creation and contents of these datasets and the design of the experiments for evaluating the implemented polyphonic transcription and guitar tablature arrangement algorithms.

Chapter 5 presented and discussed the experimental results. It was found that the implemented polyphonic transcription algorithm exhibited reduced performance on the compiled dataset of guitar recordings in comparison to the results of the algorithm reported by MIREX on a small dataset of piano recordings. It was also found that the application of a distortion audio effect to the guitar recordings significantly decreased the performance of the polyphonic transcription algorithm. The implemented guitar tablature arrangement algorithm was also evaluated. The tablature arrangements generated by DarwinTab were compared to human-arranged tablature and tablature arranged by two commercial reference algorithms. It was found that DarwinTab generated tablature with a performance difficulty

that, on average, coincided with the performance difficulty of tablature arranged by humans and the reference algorithms. However, it was found that the variance in performance difficulty of tablature generated by DarwinTab was higher than those of the reference algorithms.

6.1 Summary of Contributions

The most significant contribution of this thesis is the design and implementation of the open-source web-based guitar tablature transcription framework, entitled *Robotaba*. The framework facilitates the rapid development of guitar tablature transcription web applications, providing a vessel for music researchers to publicize their polyphonic transcription and guitar tablature arrangement algorithms, while allowing researchers to focus on algorithm development instead of application development. As part of Robotaba, an open-source program has been implemented to convert MusicXML files to MEI files and MEI files to MusicXML files.

As a proof of concept, a guitar tablature transcription web application has been developed using the Robotaba framework. An open-source polyphonic transcription application has been implemented which uses the state-of-the-art polyphonic transcription algorithm proposed by Zhou and Reiss (2008). Furthermore, an open-source guitar tablature arrangement application has been implemented. Although GAs have been applied to the guitar tablature arrangement problem before (Tuohy and Potter 2005; Tuohy and Potter 2006b), DarwinTab extends these algorithms to produce guitar-specific tablature arrangements by considering the number of frets, tuning, and capo position of the guitar on which the tablature is intended to be performed.

Another important contribution to the field of MIR is the ground-truth dataset for polyphonic guitar transcription and the ground-truth dataset for guitar tablature arrangement, which can be used for training machine-learning algorithms or for algorithm evaluation. The polyphonic guitar transcription dataset consists of 150 synthesized guitar recordings, totalling approximately 11 hours of audio, which have been semi-automatically annotated. The guitar tablature transcription dataset consists of 75 hand-arranged tablatures encoded in the MEI and MusicXML symbolic music notation file formats.

6.2 Future Work

Now that a framework has been constructed to allow polyphonic transcription and guitar tablature arrangement algorithms to be combined to generate guitar tablature transcriptions directly from an audio recording, more work can be done to improve the algorithms themselves. Though the implemented polyphonic transcription algorithm performs in realtime, the runtime required for DarwinTab to find an adequate tablature arrangement for a piece of music is perhaps longer than the average user is willing to wait. For example, using the largest GA population size tested, the average time required for DarwinTab to generate a tablature arrangement for a symbolic music score with an average of 64 notes is approximately one and a half hours. Future work will explore extensions to alternate tablature arrangement algorithms that are capable of rapidly generating arrangements, such as neural networks (Tuohy and Potter 2006a).

Since correctly annotated datasets are an extremely valuable resource in the music research community, more energy will also be directed towards increasing the size of the compiled ground-truth datasets using the dataset creation methodology described in Chapter 3. Moreover, the creation of an alternate polyphonic transcription ground-truth dataset that contains no guitar ornamentation in the synthesized audio recordings could prove useful to the research community. As well, this alternate dataset could be used to provide a definitive answer to the question of whether or not guitar ornamentation significantly degrades transcription performance. It is hoped that these datasets will stimulate future research in the area of automatic guitar tablature transcription.

Appendix A

Guitar Terminology

The purpose of this appendix is to describe guitar-specific terminology used in this thesis. The following terms include physical parts or accessories of the guitar, guitar note ornamentations, and other commonly used phrases in the guitar community.

Frets

Metal dividers embedded in the fretboard of the guitar that are strategically spaced to enforce an equal-tempered division of the octave. Depressing a string over a particular fret changes the length of the string permitted to vibrate, changing the sounding pitch.

Capo

A device that is clipped onto the fretboard and raises the pitches of the open strings of the guitar. Figure A.1 displays an acoustic guitar with a capo placed on the second fret, which raises the pitches of the open guitar strings by two semitones.

Bend

A type of note ornamentation whereby the guitarist drags a stopped (depressed) string vertically along the fretboard, stretching the string and consequently raising the sounding pitch.

Hammer-on

A type of note ornamentation whereby the guitarist sharply depresses a fret occurring further along the fretboard of a currently depressed fret, without plucking the string with the right hand.

Fig. A.1: An acoustic guitar with a capo placed on the second fret.

Pull-off

A type of note ornamentation whereby the guitarist rapidly removes his or her finger from a fret occuring further along the fretboard of a currently depressed fret, without plucking the string with the right hand.

Slide

A type of note ornamentation whereby the guitarist gradually moves their finger along a string between two fret positions, resulting in a gradual change in pitch.

Palm Muting

A type of note ornamentation whereby the palm of the right hand of the guitarist gently touches the strings, resulting in a dampened sound.

Dead Notes

A muted note with no discernible pitch. A dead note is intended to be more percussive sounding than melodic and is performed by lightly touching a string with the fretting hand and plucking the string with the plucking hand.

Right-hand Tapping

A type of note ornamentation whereby the guitarist performs hammer-ons and pull-offs with fingers on their right hand instead of their left hand.

Clean Guitar

A guitar signal in which no audio effects are applied.

Distortion Guitar

A guitar signal with a distortion audio effect applied.

Appendix B

Software Engineering Diagrams

The purpose of this appendix is to introduce standard software engineering diagrams used to convey relationships between events or objects. Section B.1 introduces the unified modeling language (UML) sequence diagram (Fowler 2003), and Section B.2 introduces the entity relationship (ER) diagram (Chen 1976).

B.1 UML Sequence Diagram

The purpose of a UML sequence diagram is to illustrate the necessary sequence of interactions between objects to perform a task. Figure B.1 provides a simple UML sequence diagram as an example. At the top of the sequence diagram are a series of rectangles with dashed lines extruding from the bottom. These rectangles represent the objects that are capable of sending messages and performing operations in response to received messages. Messages passed between objects are represented by a series of arrows. A solid arrow represents a message that requests another object to perform an operation. A dashed arrow represents a message that returns information to another object. The vertical dimension of the diagram conveys the time sequence of interactions between objects, where messages are arranged from top to bottom. In other words, if an arrow appears above another arrow, the former message is passed before the latter. In the provided example, there is one `Audio` object that represents a digital audio recording. Starting from the top, the first message requests the number of samples in the audio file, which the object calculates and returns as a second message.

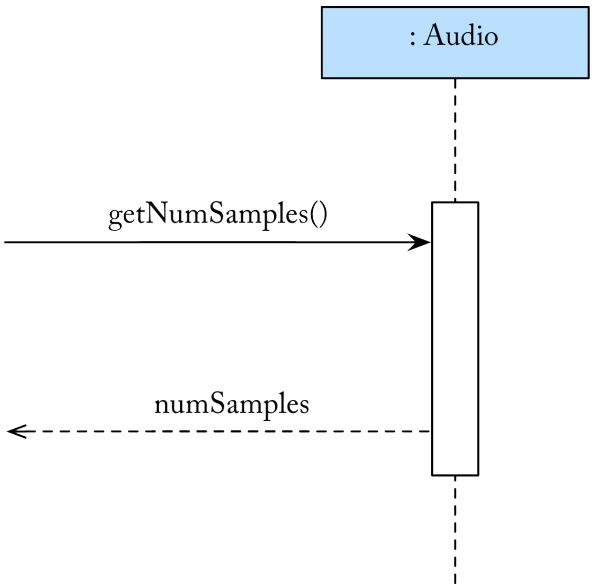


Fig. B.1: Simple example of a UML sequence diagram.

B.2 Entity-Relationship Diagram

An ER diagram illustrates the structure and relationships between data in a database. In an ER diagram, rectangles are used to represent entities, ovals are used to represent attributes of an entity or relationship, and a rhombus with a pair of connecting lines is used to represent the relationship between entities. An entity represents a physical object, event, or idea. Typically an entity is represented by a table in a relational database. Attributes describe properties of an entity. The attributes associated with an entity become the fields of the database table. Relationships describe how entities are related to each other. Depending on the complexity of the relationship between entities, a relationship may also correspond to a table in the database. The cardinality of the relationship between entities, e.g., one-to-one, one-to-many, or many-to-many is depicted using crow's foot notation, which is outlined in Table B.1. Summarizing this information, a simple ER diagram is displayed in Figure B.2. There are two entities shown: a `Guitar` entity and a `String` entity. The `Guitar` entity has two attributes: colour and weight. The relationship between the two entities is read from left to right—"the guitar has one or more strings"—and from

right to left—"a string is part of zero or one guitar", since a string may exist apart from a guitar.

Table B.1: Crow's foot notation to specify the cardinality of relationships between entities in an entity relationship diagram.

Cardinality	Notation
zero or one	
one and only one	
zero or many	
one or many	

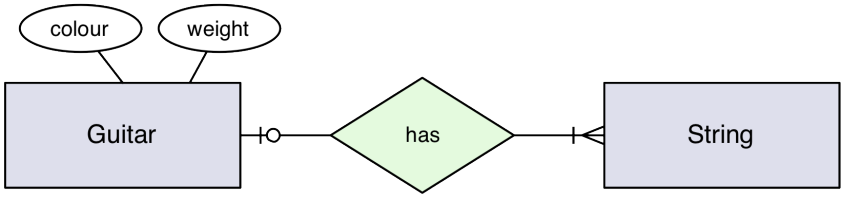


Fig. B.2: Simple example of an Entity Relationship diagram.

Appendix C

Detailed Description of Datasets and Results

This appendix will provide a detailed description of the pieces within the compiled ground-truth dataset for polyphonic guitar transcription (Table C.1) and guitar tablature arrangement (Table C.3). Also found in this appendix are detailed results of the polyphonic transcription evaluation experiments (Table C.2) and the guitar tablature arrangement evaluation experiments (Table C.4) for each song in the ground-truth datasets. The tables begin on the following page.

Table C.1 – Polyphonic Guitar Transcription Dataset

ID	Title	Artist	Genre	Role	Tempo	Length	Notes	Chords	Polyphony
1	Kryptonite	3 Doors Down	Alternative Rock	Lead	100	3:43	1143	165	1.65
2	Back in Black	AC/DC	Hard Rock	Lead	96	4:10	1747	362	2.00
3	Back in Black	AC/DC	Hard Rock	Rhythm	96	4:10	1637	458	2.69
4	Dream On	Aerosmith	Hard Rock	Rhythm	80	4:18	980	235	1.52
5	Dream On	Aerosmith	Hard Rock	Lead	80	4:18	724	225	1.55
6	Unholy Confessions	Avenged Sevenfold	Heavy Metal	Lead	95	4:39	1237	236	1.24
7	While My Guitar Gently Weeps	The Beatles	Pop Rock	Rhythm	112	4:54	4417	844	5.23
8	Yesterday	The Beatles	Pop Rock	Lead	96	0:55	237	100	2.28
9	Fallen Leaves	Billy Talent	Pop Punk	Rhythm	123	3:15	2094	617	3.04
10	Tighten Up	Black Keys	Indie Rock	Lead	108	3:24	720	145	1.67
11	Tighten Up	Black Keys	Indie Rock	Rhythm	108	3:24	1167	357	3.27
12	Paranoid	Black Sabbath	Hard Rock	Lead	164	2:44	1472	688	2.01
13	All The Small Things	Blink 182	Pop Punk	Rhythm	148	2:54	1544	539	2.34
14	Don't Fear The Reaper	Blue Oyster Cult	Hard Rock	Lead	142	4:57	1650	254	1.31
15	Nottingham Lace	Buckethead	Alternative Metal	Lead	115	6:12	2779	1196	1.92
16	Johnny B Goode	Chuck Berry	Rock	Lead	170	2:35	604	184	1.45
17	Johnny B Goode	Chuck Berry	Rock	Rhythm	170	2:35	1583	783	2.02
18	One Last Breath	Creed	Rock	Lead	120	4:12	1486	331	1.86
19	Crystal Mountain	Death	Death Metal	Lead	179	4:41	565	0	1.00
20	Crystal Mountain	Death	Death Metal	Rhythm	179	4:41	1019	125	1.22
21	Smoke On The Water	Deep Purple	Hard Rock	Lead	112	5:45	996	340	1.59
22	Hotel California	The Eagles	Rock	Lead	74	6:32	3352	811	2.80
23	Layla	Eric Clapton	Rock	Lead	120	2:37	851	138	2.02
24	Tears in Heaven	Eric Clapton	Rock	Lead	78	4:34	1065	290	1.64
25	Everlong	Foo Fighters	Hard Rock	Lead	156	4:05	3505	951	2.89
26	The Pretender	Foo Fighters	Hard Rock	Lead	90	4:16	3984	1122	3.00
27	Your Revolution Is A Joke	Funeral For A Friend	Emo	Lead	148	2:39	793	77	1.11
28	American Idiot	Green Day	Pop Punk	Rhythm	200	2:41	1908	664	2.87
29	Wake Me Up When September Ends	Green Day	Pop Punk	Lead	105	4:25	1807	485	2.01
30	Don't Cry	Guns N' Roses	Hard Rock	Lead	62	4:40	918	107	1.31

Continued on the next page...

Table C.1 – Continued

ID	Title	Artist	Genre	Role	Tempo	Length	Notes	Chords	Polyphony
31	Knockin' On Heaven's Door	Guns N' Roses	Hard Rock	Lead	64	5:41	1330	337	2.18
32	Knockin' On Heaven's Door	Guns N' Roses	Hard Rock	Rhythm	64	5:41	1304	438	2.73
33	November Rain	Guns N' Roses	Hard Rock	Lead	80	7:32	1293	279	2.35
34	Sweet Child O' Mine	Guns N' Roses	Hard Rock	Lead	128	5:39	1797	380	2.31
35	Sweet Child O' Mine	Guns N' Roses	Hard Rock	Rhythm	128	5:39	2421	676	3.07
36	Welcome To The Jungle	Guns N' Roses	Hard Rock	Lead	104	5:06	1995	504	1.91
37	Fear Of The Dark	Iron Maiden	Heavy Metal	Lead	76	7:38	912	130	1.17
38	All Along the Watchtower	Jimi Hendrix	Rock	Lead	112	4:02	3182	743	4.28
39	Little Wing	Jimi Hendrix	Rock	Lead	67	2:38	852	235	1.79
40	23	Jimmy Eat World	Emo	Lead	144	3:42	892	163	1.90
41	23	Jimmy Eat World	Emo	Rhythm	144	6:35	2442	490	1.47
42	Carry On Wayward Son	Kansas	Progressive Rock	Lead	132	4:50	1309	160	1.30
43	Mr. Brightside	The Killers	Indie Rock	Lead	148	3:41	1075	7	1.02
44	My Curse	Killswitch Engage	Alternative Metal	Lead	174	4:08	2182	681	2.02
45	Black Dog	Led Zeppelin	Hard Rock	Lead	160	4:44	1517	340	1.80
46	Stairway To Heaven	Led Zeppelin	Hard Rock	Lead	75	6:52	2999	713	2.97
47	What I've Done	Linkin Park	Alternative Rock	Lead	122	2:14	455	6	1.01
48	Sweet Home Alabama	Lynyrd Skynyrd	Hard Rock	Lead	100	4:36	1770	588	1.58
49	Sweet Dreams (Are Made Of This)	Marilyn Manson	Alternative Metal	Lead	92	4:21	831	172	1.26
50	Symphony of Destruction	Megadeth	Heavy Metal	Lead	140	3:51	1119	259	1.49
51	Enter Sandman	Metallica	Heavy Metal	Lead	120	5:38	1413	553	1.65
52	Fade To Black	Metallica	Heavy Metal	Lead	116	7:21	2031	328	1.49
53	The Unforgiven	Metallica	Heavy Metal	Lead	69	6:26	2722	542	2.28
54	Super Mario Bros. Theme	Unknown	n/a	Lead	110	3:03	720	196	1.45
55	Come As You Are	Nirvana	Alternative Rock	Lead	120	3:36	1093	221	1.63
56	In Bloom	Nirvana	Alternative Rock	Rhythm	78	4:18	2312	750	2.85
57	Crazy Train	Ozzy Ozbourne	Heavy Metal	Lead	138	4:24	1937	388	1.65
58	Another Brick In The Wall II	Pink Floyd	Progressive Rock	Rhythm	104	2:07	812	207	3.83
59	Comfortably Numb	Pink Floyd	Progressive Rock	Rhythm	65	5:21	5813	1148	5.05
60	Wish You Were Here	Pink Floyd	Progressive Rock	Lead	60	5:05	2228	564	2.73

Continued on the next page...

Table C.1 – Continued

ID	Title	Artist	Genre	Role	Tempo	Length	Notes	Chords	Polyphony
61	Hey There Delilah	Plain White Tees	Pop Punk	Lead	100	3:53	1095	355	1.48
62	Purple Rain	Prince	Rock	Lead	58	6:21	1008	226	3.09
63	Creep	Radiohead	Alternative Rock	Lead	92	3:52	990	116	1.71
64	Karma Police	Radiohead	Alternative Rock	Lead	76	4:16	1556	504	2.27
65	Killing in the Name of	Rage Against the Machine	Alternative Metal	Lead	110	4:20	1092	415	1.72
66	Californication	Red Hot Chili Peppers	Alternative Rock	Lead	96	5:18	3464	649	3.85
67	Scar Tissue	Red Hot Chili Peppers	Alternative Rock	Lead	89	3:33	2488	502	3.07
68	Angie	Rolling Stones	Rock	Lead	60	5:12	1218	304	2.88
69	Rock You Like A Hurricane	Scorpions	Hard Rock	Rhythm	120	4:21	2067	689	3.00
70	1979	The Smashing Pumpkins	Alternative Rock	Rhythm	130	4:15	3023	1009	2.99
71	Black Hole Sun	Soundgarden	Alternative Rock	Lead	52	5:29	1408	224	1.80
72	Heart In A Cage	The Strokes	Indie Rock	Lead	138	3:18	857	36	1.05
73	Heart In A Cage	The Strokes	Indie Rock	Rhythm	138	3:18	2278	525	2.58
74	Reptilia	The Strokes	Indie Rock	Lead	160	3:26	1182	129	1.78
75	Animal I have Become	Three Days Grace	Alternative Rock	Rhythm	122	3:30	724	204	2.29

Average Tempo	112 bpm
Total Audio Length	5:29:30×2 = 10:59:00
Total Number of Notes	125,192
Total Number of Chords	30,914
Average Polyphony	2.16

Table C.1: Polyphonic transcription evaluation dataset.

Table C.2 – Polyphonic Guitar Transcription Experiment Results

	Experiment 1			Experiment 2			Experiment 3			Experiment 4		
ID	Precision	Recall	f-measure	Precision	Recall	f-measure	Precision	Recall	f-measure	Precision	Recall	f-measure
1	0.62	0.33	0.44	0.69	0.37	0.48	0.56	0.41	0.47	0.65	0.47	0.55
2	0.63	0.45	0.53	0.67	0.49	0.56	0.58	0.43	0.49	0.63	0.47	0.54
3	0.69	0.48	0.56	0.74	0.51	0.61	0.60	0.43	0.50	0.66	0.47	0.55
4	0.92	0.70	0.80	0.92	0.71	0.80	0.67	0.57	0.62	0.70	0.59	0.64
5	0.66	0.90	0.76	0.69	0.94	0.79	0.49	0.74	0.59	0.59	0.89	0.71
6	0.41	0.43	0.42	0.55	0.58	0.57	0.42	0.49	0.45	0.51	0.60	0.55
7	0.97	0.23	0.37	0.99	0.23	0.38	0.62	0.25	0.36	0.66	0.27	0.38
8	0.75	0.37	0.50	0.82	0.41	0.54	0.55	0.39	0.46	0.71	0.50	0.59
9	0.74	0.51	0.60	0.77	0.52	0.62	0.65	0.48	0.55	0.69	0.51	0.59
10	0.66	0.54	0.59	0.67	0.54	0.60	0.33	0.41	0.37	0.44	0.55	0.49
11	0.95	0.88	0.91	0.97	0.90	0.93	0.29	0.44	0.35	0.45	0.68	0.54
12	0.90	0.18	0.31	0.92	0.19	0.31	0.82	0.65	0.72	0.84	0.67	0.75
13	0.68	0.14	0.23	0.74	0.15	0.25	0.51	0.18	0.26	0.77	0.27	0.40
14	0.86	0.78	0.82	0.86	0.78	0.82	0.56	0.65	0.60	0.61	0.71	0.65
15	0.74	0.80	0.77	0.77	0.82	0.79	0.54	0.59	0.56	0.63	0.68	0.65
16	0.76	0.46	0.57	0.81	0.49	0.61	0.51	0.37	0.43	0.73	0.53	0.61
17	0.25	0.07	0.11	0.26	0.08	0.12	0.24	0.12	0.16	0.32	0.16	0.21
18	0.62	0.31	0.41	0.65	0.33	0.44	0.37	0.24	0.29	0.48	0.32	0.38
19	0.87	0.92	0.90	0.87	0.92	0.90	0.86	0.96	0.91	0.87	0.98	0.92
20	0.53	0.57	0.55	0.57	0.62	0.60	0.45	0.60	0.51	0.49	0.66	0.57
21	0.36	0.45	0.40	0.44	0.54	0.48	0.23	0.31	0.27	0.39	0.52	0.45
22	0.81	0.32	0.46	0.83	0.33	0.47	0.62	0.29	0.39	0.77	0.36	0.49
23	0.79	0.32	0.46	0.79	0.32	0.46	0.35	0.28	0.31	0.50	0.40	0.45
24	0.50	0.52	0.51	0.53	0.55	0.54	0.30	0.37	0.33	0.40	0.50	0.44
25	0.60	0.20	0.29	0.73	0.24	0.36	0.49	0.23	0.31	0.60	0.28	0.39
26	0.89	0.14	0.24	0.97	0.15	0.26	0.43	0.07	0.12	0.56	0.09	0.15
27	0.89	0.86	0.88	0.89	0.86	0.88	0.84	0.86	0.85	0.85	0.87	0.86
28	0.55	0.23	0.33	0.74	0.31	0.44	0.53	0.29	0.38	0.65	0.36	0.46
29	0.28	0.11	0.16	0.29	0.12	0.17	0.21	0.09	0.13	0.24	0.10	0.14
30	0.84	0.69	0.76	0.85	0.69	0.76	0.65	0.63	0.64	0.69	0.67	0.68
31	0.85	0.40	0.54	0.86	0.40	0.55	0.61	0.38	0.47	0.66	0.40	0.50

Continued on the next page...

Table C.2 – Continued

	Experiment 1			Experiment 2			Experiment 3			Experiment 4		
ID	Precision	Recall	f-measure	Precision	Recall	f-measure	Precision	Recall	f-measure	Precision	Recall	f-measure
32	0.82	0.34	0.48	0.82	0.34	0.48	0.55	0.30	0.39	0.63	0.34	0.44
33	0.51	0.34	0.41	0.51	0.35	0.41	0.35	0.28	0.31	0.37	0.30	0.33
34	0.94	0.48	0.64	0.95	0.49	0.64	0.68	0.43	0.53	0.73	0.46	0.57
35	0.69	0.35	0.47	0.77	0.39	0.52	0.35	0.25	0.29	0.41	0.30	0.35
36	0.76	0.51	0.61	0.81	0.55	0.65	0.42	0.38	0.40	0.54	0.48	0.51
37	0.64	0.39	0.49	0.66	0.40	0.50	0.40	0.29	0.34	0.42	0.31	0.36
38	0.95	0.28	0.44	0.97	0.29	0.45	0.64	0.27	0.38	0.72	0.30	0.43
39	0.64	0.34	0.44	0.67	0.35	0.46	0.42	0.31	0.36	0.53	0.39	0.45
40	0.89	0.26	0.40	0.96	0.28	0.43	0.55	0.29	0.38	0.58	0.30	0.40
41	0.87	0.50	0.64	0.88	0.50	0.64	0.46	0.27	0.34	0.52	0.31	0.39
42	0.75	0.67	0.71	0.76	0.68	0.72	0.42	0.59	0.49	0.47	0.66	0.55
43	0.97	0.93	0.95	0.97	0.94	0.95	0.81	0.85	0.83	0.81	0.86	0.83
44	0.50	0.20	0.29	0.58	0.23	0.33	0.17	0.09	0.12	0.32	0.17	0.22
45	0.84	0.61	0.71	0.87	0.62	0.73	0.44	0.40	0.42	0.55	0.50	0.53
46	0.70	0.40	0.51	0.83	0.48	0.61	0.53	0.35	0.42	0.65	0.43	0.52
47	0.96	0.24	0.38	0.96	0.24	0.38	0.36	0.15	0.21	0.39	0.16	0.23
48	0.60	0.39	0.47	0.61	0.40	0.48	0.26	0.26	0.26	0.33	0.32	0.33
49	0.83	0.55	0.66	0.85	0.56	0.67	0.46	0.61	0.52	0.47	0.62	0.53
50	0.27	0.43	0.33	0.29	0.45	0.35	0.26	0.32	0.28	0.37	0.45	0.41
51	0.24	0.19	0.21	0.27	0.21	0.24	0.20	0.20	0.20	0.21	0.21	0.21
52	0.57	0.60	0.58	0.60	0.63	0.61	0.45	0.50	0.47	0.52	0.59	0.55
53	0.95	0.51	0.66	0.95	0.51	0.66	0.63	0.34	0.44	0.68	0.36	0.47
54	0.38	0.32	0.35	0.40	0.33	0.36	0.22	0.17	0.19	0.31	0.24	0.27
55	0.66	0.58	0.62	0.67	0.59	0.63	0.31	0.38	0.34	0.45	0.54	0.49
56	0.64	0.20	0.30	0.78	0.24	0.37	0.43	0.19	0.27	0.46	0.21	0.29
57	0.79	0.51	0.62	0.81	0.53	0.64	0.42	0.33	0.37	0.50	0.39	0.43
58	0.86	0.45	0.59	0.95	0.49	0.65	0.63	0.36	0.46	0.74	0.43	0.54
59	0.94	0.16	0.27	0.99	0.16	0.28	0.72	0.10	0.18	0.91	0.13	0.22
60	0.49	0.21	0.30	0.53	0.23	0.32	0.24	0.18	0.21	0.34	0.26	0.29
61	0.84	0.65	0.73	0.94	0.73	0.82	0.46	0.43	0.45	0.56	0.52	0.54
62	0.84	0.56	0.67	0.87	0.59	0.70	0.25	0.23	0.24	0.33	0.30	0.31

Continued on the next page...

Table C.2 – Continued

	Experiment 1			Experiment 2			Experiment 3			Experiment 4		
ID	Precision	Recall	f-measure	Precision	Recall	f-measure	Precision	Recall	f-measure	Precision	Recall	f-measure
63	0.94	0.45	0.61	0.97	0.46	0.62	0.72	0.46	0.57	0.74	0.48	0.58
64	0.84	0.56	0.68	0.88	0.59	0.71	0.68	0.47	0.56	0.78	0.54	0.64
65	0.44	0.39	0.41	0.53	0.47	0.50	0.37	0.23	0.28	0.52	0.33	0.40
66	0.87	0.23	0.36	0.89	0.24	0.37	0.52	0.19	0.28	0.61	0.22	0.33
67	0.94	0.21	0.35	0.95	0.21	0.35	0.70	0.15	0.24	0.79	0.17	0.28
68	0.46	0.26	0.33	0.50	0.29	0.37	0.27	0.17	0.21	0.38	0.24	0.30
69	0.73	0.30	0.43	0.78	0.32	0.45	0.47	0.29	0.36	0.58	0.37	0.45
70	0.68	0.16	0.26	0.79	0.18	0.30	0.34	0.09	0.14	0.42	0.11	0.18
71	0.44	0.27	0.33	0.45	0.28	0.35	0.31	0.23	0.26	0.35	0.25	0.29
72	0.93	0.48	0.63	0.94	0.49	0.64	0.86	0.63	0.72	0.87	0.64	0.74
73	0.75	0.09	0.15	0.76	0.09	0.15	0.32	0.09	0.14	0.41	0.12	0.18
74	0.86	0.42	0.57	0.87	0.42	0.57	0.53	0.45	0.49	0.55	0.46	0.50
75	0.44	0.47	0.46	0.53	0.57	0.55	0.22	0.26	0.24	0.34	0.40	0.37
Average	0.71	0.42	0.50	0.75	0.45	0.53	0.48	0.36	0.39	0.56	0.42	0.46

Table C.2: Results of the four experiments conducted to evaluate the implemented polyphonic transcription algorithm.

Table C.3 – Guitar Tablature Arrangement Dataset

ID	Title	Artist	Genre	Role	Notes	Chords	Polyphony
1	Kryptonite	3 Doors Down	Alternative Rock	Lead	30	0	1.00
2	Back in Black	AC/DC	Hard Rock	Lead	47	5	1.12
3	Back in Black	AC/DC	Hard Rock	Rhythm	89	27	2.54
4	Dream On	Aerosmith	Hard Rock	Rhythm	67	9	1.52
5	Dream On	Aerosmith	Hard Rock	Lead	44	15	1.52
6	Unholy Confessions	Avenged Sevenfold	Heavy Metal	Lead	54	4	1.08
7	While My Guitar Gently Weeps	The Beatles	Pop Rock	Rhythm	120	24	5.00
8	Yesterday	The Beatles	Pop Rock	Lead	107	44	2.23
9	Fallen Leaves	Billy Talent	Pop Punk	Rhythm	103	33	3.12
10	Tighten Up	Black Keys	Indie Rock	Lead	67	0	1.00
11	Tighten Up	Black Keys	Indie Rock	Rhythm	87	29	3.00
12	Paranoid	Black Sabbath	Hard Rock	Lead	84	37	1.87
13	All The Small Things	Blink 182	Pop Punk	Rhythm	88	28	2.75
14	Don't Fear The Reaper	Blue Oyster Cult	Hard Rock	Lead	74	24	1.85
15	Nottingham Lace	Buckethead	Alternative Metal	Lead	80	40	2.00
16	Johnny B, Goode	Chuck Berry	Rock	Lead	63	23	1.57
17	Johnny B, Goode	Chuck Berry	Rock	Rhythm	80	40	2.00
18	One Last Breath	Creed	Rock	Lead	74	3	1.04
19	Crystal Mountain	Death	Death Metal	Lead	34	0	1.00
20	Crystal Mountain	Death	Death Metal	Rhythm	48	10	1.26
21	Smoke On The Water	Deep Purple	Hard Rock	Lead	47	18	1.62
22	Hotel California	The Eagles	Rock	Lead	59	5	1.20
23	Layla	Eric Clapton	Rock	Lead	90	20	2.90
24	Tears in Heaven	Eric Clapton	Rock	Lead	44	11	1.63
25	Everlong	Foo Fighters	Hard Rock	Lead	81	25	1.45
26	The Pretender	Foo Fighters	Hard Rock	Lead	40	3	1.29
27	Your Revolution Is A Joke	Funeral For A Friend	Emo	Lead	33	4	1.14
28	American Idiot	Green Day	Pop Punk	Rhythm	68	24	2.83
29	Wake Me Up When September Ends	Green Day	Pop Punk	Lead	115	35	2.40
30	Don't Cry	Guns N' Roses	Hard Rock	Lead	54	1	1.02

Continued on the next page...

Table C.3 – Continued

ID	Title	Artist	Genre	Role	Notes	Chords	Polyphony
31	Knockin' On Heaven's Door	Guns N' Roses	Hard Rock	Lead	34	3	1.17
32	Knockin' On Heaven's Door	Guns N' Roses	Hard Rock	Rhythm	91	29	2.76
33	November Rain	Guns N' Roses	Hard Rock	Lead	46	2	1.15
34	Sweet Child O' Mine	Guns N' Roses	Hard Rock	Lead	40	2	1.18
35	Sweet Child O' Mine	Guns N' Roses	Hard Rock	Rhythm	124	30	3.10
36	Welcome To The Jungle	Guns N' Roses	Hard Rock	Lead	36	4	1.29
37	Fear Of The Dark	Iron Maiden	Heavy Metal	Lead	36	6	1.20
38	All Along the Watchtower	Jimi Hendrix	Rock	Lead	127	21	5.77
39	Little Wing	Jimi Hendrix	Rock	Lead	55	7	1.15
40	23	Jimmy Eat World	Emo	Lead	32	0	1.00
41	23	Jimmy Eat World	Emo	Rhythm	63	15	1.31
42	Carry On Wayward Son	Kansas	Progressive Rock	Lead	40	0	1.00
43	Mr. Brightside	The Killers	Indie Rock	Lead	32	0	1.00
44	My Curse	Killswitch Engage	Alternative Metal	Lead	90	36	1.67
45	Black Dog	Led Zeppelin	Hard Rock	Lead	29	2	1.12
46	Stairway To Heaven	Led Zeppelin	Hard Rock	Lead	92	27	3.07
47	What I've Done	Linkin Park	Alternative Rock	Lead	32	0	1.00
48	Sweet Home Alabama	Lynyrd Skynyrd	Hard Rock	Lead	55	9	1.34
49	Sweet Dreams (Are Made Of This)	Marilyn Manson	Alternative Metal	Lead	46	15	1.48
50	Symphony of Destruction	Megadeth	Heavy Metal	Lead	63	18	2.33
51	Enter Sandman	Metallica	Heavy Metal	Lead	43	19	1.79
52	Fade To Black	Metallica	Heavy Metal	Lead	43	9	1.43
53	The Unforgiven	Metallica	Heavy Metal	Lead	32	2	1.07
54	Super Mario Bros. Theme	Unknown	n/a	Lead	30	0	1.00
55	Come As You Are	Nirvana	Alternative Rock	Lead	48	12	1.85
56	In Bloom	Nirvana	Alternative Rock	Rhythm	66	22	3.00
57	Crazy Train	Ozzy Ozbourne	Heavy Metal	Lead	67	8	1.37
58	Another Brick In The Wall II	Pink Floyd	Progressive Rock	Rhythm	76	21	3.62
59	Comfortably Numb	Pink Floyd	Progressive Rock	Rhythm	147	27	5.44
60	Wish You Were Here	Pink Floyd	Progressive Rock	Lead	69	19	2.76

Continued on the next page...

Table C.3 – Continued

ID	Title	Artist	Genre	Role	Notes	Chords	Polyphony
61	Hey There Delilah	Plain White Tees	Pop Punk	Lead	48	16	1.50
62	Purple Rain	Prince	Rock	Lead	73	15	2.81
63	Creep	Radiohead	Alternative Rock	Lead	60	1	1.02
64	Karma Police	Radiohead	Alternative Rock	Lead	104	37	2.42
65	Killing in the Name of	Rage Against the Machine	Alternative Metal	Lead	72	24	1.50
66	Californication	Red Hot Chili Peppers	Alternative Rock	Lead	87	21	3.22
67	Scar Tissue	Red Hot Chili Peppers	Alternative Rock	Lead	43	6	1.16
68	Angie	Rolling Stones	Rock	Lead	53	17	2.12
69	Rock You Like A Hurricane	Scorpions	Hard Rock	Rhythm	60	20	3.00
70	1979	The Smashing Pumpkins	Alternative Rock	Rhythm	91	31	2.94
71	Black Hole Sun	Soundgarden	Alternative Rock	Lead	55	9	2.12
72	Heart In A Cage	The Strokes	Indie Rock	Lead	48	0	1.00
73	Heart In A Cage	The Strokes	Indie Rock	Rhythm	120	32	3.75
74	Reptilia	The Strokes	Indie Rock	Lead	32	0	1.00
75	Animal I Have Become	Three Days Grace	Alternative Rock	Rhythm	44	8	1.57
Total Number of Notes		4,845					
Total Number of Chords		1,143					
Average Polyphony		1.94					

Table C.3: Guitar tablature arrangement evaluation dataset.

Table C.4 – Guitar Tablature Arrangement Experiment Results

ID	Experiment 1	Experiment 2	Experiment 3	Guitar Pro	Sibelius
1	0.71	1.00	1.00	1.00	1.00
2	0.78	1.02	1.22	0.74	0.54
3	0.68	1.01	1.11	1.08	1.25
4	0.74	0.92	0.99	0.86	0.49
5	0.74	0.78	1.06	1.00	0.41
6	1.49	1.91	1.84	2.14	2.14
7	0.48	0.70	0.94	1.00	1.00
8	0.29	0.60	0.66	1.00	1.00
9	0.58	0.97	1.00	1.00	0.41
10	0.96	1.81	3.76	5.89	9.73
11	0.23	0.35	0.53	0.83	1.02
12	0.85	1.03	1.21	1.03	1.60
13	0.73	1.01	1.09	1.23	1.17
14	0.90	1.03	1.00	0.91	1.06
15	0.56	0.99	1.15	1.68	1.27
16	0.37	0.50	0.68	1.17	1.36
17	1.22	1.41	1.64	1.04	1.89
18	0.54	0.75	0.97	0.94	1.07
19	1.12	1.47	2.05	2.25	1.26
20	1.99	2.60	2.60	1.55	2.80
21	1.39	2.10	2.15	1.62	1.95
22	0.87	1.46	1.51	1.00	0.66
23	0.23	0.39	0.45	0.93	0.83
24	0.90	1.00	1.00	0.89	0.89
25	1.14	1.71	1.66	1.03	1.85
26	1.14	1.55	2.34	1.14	1.11
27	1.24	1.25	1.15	0.71	0.96
28	1.04	1.16	1.18	1.02	1.18
29	0.40	0.62	0.73	1.00	0.90
30	0.63	0.89	0.87	1.07	1.00
31	0.92	0.94	0.94	1.00	1.00
32	0.43	0.74	0.95	0.92	0.79
33	0.51	0.79	1.05	1.10	1.36
34	1.09	1.17	1.14	1.16	0.33
35	0.36	0.61	0.74	1.00	1.00

Continued on the next page...

Table C.4 – Continued

ID	Experiment 1	Experiment 2	Experiment 3	Guitar Pro	Sibelius
36	1.59	2.18	1.74	1.57	2.09
37	1.51	1.71	2.78	1.92	3.38
38	0.25	0.37	0.47	1.00	0.83
39	0.86	1.15	1.10	1.04	0.62
40	0.77	1.43	1.09	3.21	3.21
41	0.53	0.71	0.75	1.00	1.00
42	9.00	9.00	9.00	9.00	9.00
43	1.72	1.77	1.72	1.67	0.43
44	1.50	1.57	1.64	1.05	0.90
45	1.49	1.88	1.88	2.32	2.32
46	0.37	0.57	0.65	1.00	0.87
47	0.73	2.06	4.36	5.19	5.19
48	0.63	0.88	0.91	1.00	1.00
49	1.32	1.39	1.39	1.00	1.21
50	1.00	1.00	1.00	1.00	1.00
51	1.89	2.23	2.23	2.23	2.23
52	0.43	0.63	0.64	1.00	0.72
53	0.30	0.47	0.57	1.00	1.00
54	0.66	1.08	1.38	1.03	0.93
55	0.48	0.81	0.81	1.03	1.00
56	1.00	1.04	1.14	1.03	1.15
57	1.40	2.11	2.11	1.63	1.63
58	0.30	0.60	0.58	1.00	0.59
59	0.56	0.59	0.58	1.00	1.00
60	0.79	0.99	0.99	1.00	1.00
61	0.75	0.87	0.99	1.00	1.00
62	0.58	0.87	0.76	0.95	0.78
63	0.48	0.80	1.22	1.74	0.82
64	0.32	0.53	0.85	1.00	1.00
65	1.31	1.82	1.89	1.49	1.49
66	0.55	0.96	0.99	0.99	0.99
67	0.91	1.08	1.16	1.10	1.16
68	0.33	0.47	0.65	0.88	0.76
69	0.85	1.22	1.09	1.06	1.23
70	2.45	2.68	2.68	2.01	1.91
71	1.00	1.00	0.99	0.90	0.82

Continued on the next page...

Table C.4 – Continued

ID	Experiment 1	Experiment 2	Experiment 3	Guitar Pro	Sibelius
72	1.46	1.75	1.75	1.36	0.59
73	0.13	0.21	0.26	1.00	0.96
74	1.09	1.09	1.09	1.09	0.49
75	4.54	4.54	4.54	4.54	4.54
$\hat{\mu}$	0.78	1.01	1.09	1.03	1.00
$\hat{\sigma}$	0.46	0.57	0.49	0.11	0.33

Table C.4: Results of the three experiments conducted to evaluate the implemented guitar tablature arrangement algorithm, alongside the reference algorithms provided by Guitar Pro and Sibelius. $\hat{\mu}$ denotes the median and $\hat{\sigma}$ denotes the standardized median absolute deviation.

Bibliography

Abesser, J. 2012. Automatic string detection for bass guitar and electric guitar. In *Proceedings of the International Symposium on Computer Music Modeling and Retrieval*, London, UK, 567–82.

Allen, T., and C. Goudeseune. 2011. Topological considerations for tuning and fingering stringed instruments. *Cornell University Computing Research Repository*.

Apel, W. 1953. *The Notation of Polyphonic Music 900–1600,* 5th ed. Cambridge, MA: The Mediaeval Academy of America.

Bach, J. S. 1885. *Orchestral Suite No.2 in B minor, Ouverture (BWV 1067)*, 202. Bach-Gesellschaft Ausgabe, Band 31.1: Orchesterwerke.

Barbancho, A. M., A. Klapuri, L. J. Tardón, and I. Barbancho. 2012. Automatic transcription of guitar chords and fingering from audio. *IEEE Transactions on Audio, Speech, and Language Processing* 20 (3): 915–21.

Barbancho, I., L. J. Tardón, A. M. Barbancho, and S. Sammartino. 2009. Pitch and played string estimation in classic and acoustic guitars. In *Proceedings of the Audio Engineering Society Convention*, Munich, Germany, 1–9.

Bay, M., A. F. Ehmann, and J. S. Downie. 2009. Evaluation of multiple-F0 estimation and tracking systems. In *Proceedings of the International Society for Music Information Retrieval Conference*, Kobe, Japan, 315–20.

Bello, J. P. 2003. Towards the automated analysis of simple polyphonic music: A knowledge based approach. Ph. D. thesis, Department of Electronic Engineering, Queen Mary University of London.

Bello, J. P., and M. Sandler. 2000. Blackboard system and top-down processing for the transcription of simple polyphonic music. In *Proceedings of the COST G-6 Conference on Digital Audio Effects*, Verona, Italy.

Benetos, E., and S. Dixon. 2011a. Multiple-instrument polyphonic music transcription using a convolutive probabilistic model. In *Proceedings of the Sound and Music Computing Conference*, Padova, Italy, 19–24.

Benetos, E., and S. Dixon. 2011b. Polyphonic music transcription using note onset and offset detection. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, 37–40.

Benetos, E., and S. Dixon. 2012. Multiple-F0 estimation and note tracking for MIREX 2012 using a shift-invariant latent variable model. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2012/BD3.pdf`. Accessed April 8, 2013.

Benetos, E., S. Dixon, D. Giannoulis, H. Kirchoff, and A. Klapuri. 2012. Automatic music transcription: Breaking the glass ceiling. In *Proceedings of the International Society for Music Information Retrieval Conference*, Porto, Portugal, 1002–7.

Bonnet, L., and R. Lefebvre. 2003. High-resolution robust multipitch analysis of guitar chords. In *Proceedings of the Audio Engineering Society*, Amsterdam, The Netherlands, 1–7.

Bregman, A. S. 1990. *Auditory Scene Analysis: The Perceptual Organization of Sound*. Cambridge, MA: MIT Press.

Burns, A. M. 2007. Computer vision methods for guitarist left-hand fingering recognition. Master's thesis, McGill University.

Burns, A. M., and M. M. Wanderley. 2006. Visual methods for the retrieval of guitarist fingering. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, Paris, France, 196–9.

Carolingian, H. 990. *Antiphonarium Officii (Antiphonary for liturgy of the hours)*, 32. Cod. Sang. 390: Stiftsbibliothek St. Gallen (DOI: 10.5076/e-codices-csg-0390).

Catholic Church. 1963. *The Liber Usualis, with introduction and rubrics in English*, 276. Tournai, Belgium: Desclée.

Chang, W., A. W. Y. Su, C. Yeh, A. Roebel, and X. Rodet. 2008. Multiple-F0 tracking based on a high-order HMM model. In *Proceedings of the International Conference on Digital Audio Effects*, Espoo, Finland.

Chen, P. P. 1976. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems* 1 (1): 9–36.

Cherry, E. C. 1953. Some experiments on the recognition of speech, with one and with two ears. *Journal of the Acoustical Society of America* 25 (5): 975–9.

Cheveigné, A., and H. Kawahara. 2002. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America* 111 (4): 1917–30.

Chibelushi, C., F. Deravi, and J. Mason. 2002. A review of speech-based bimodal recognition. *IEEE Transactions on Multimedia* 4 (1): 23–37.

Cormen, T. H., C. E. Leiserson, R. L. Rivest, and C. Stein. 2009. *Introduction to Algorithms,* 2nd ed., Chapter 24: Single-source shortest paths. Cambridge, MA: MIT.

Costantini, G., R. Perfetti, and M. Todisco. 2009. Event based transcription system for polyphonic piano music. *Signal Processing* 89 (9): 1798–1811.

Dailey, D. J. 2013. *Electronics for Guitarists,* 2nd ed. New York, NY: Springer.

Davy, M., and S. J. Godsill. 2002. Bayesian harmonic models for musical signal analysis. In *Proceedings of the Valencia International meeting Bayesian Statistics VII*, Tenerife, Spain. Oxford University Press.

Dessein, A., A. Cont, and G. Lemaitre. 2010. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *Proceedings of the International Society for Music Information Retrieval Conference*, Utrecht, Netherlands.

Emiya, V., R. Badeau, and B. David. 2007. Multipitch estimation of inharmonic sounds in colored noise. In *Proceedings of the International Conference of Digital Audio Effects*, Bordeaux, France, 93–8.

Emiya, V., R. Badeau, and B. David. 2008. Automatic transcription of piano music based on HMM tracking of jointly-estimated pitches. In *Proceedings of the European Signal Processing Conference*, Lausanne, Switzerland.

Fiss, X., and A. Kwasinski. 2011. Automatic real-time electric guitar audio transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Prague, Czech Republic, 373–6.

Fitzgerald, R., J. Newell, T. King, and A. Khayr. 2011. Guitar-2-tab. Technical report, Wichita State University, http://hdl.handle.net/10057/4028. Accessed: April 8, 2013.

Fletcher, N. F., and T. D. Rossing. 1998. *The physics of musical instruments,* 2nd ed. New York: Springer.

Fowler, M. 2003. *UML distilled: A brief guide to the standard object modeling language,* 3rd ed. Boston, MA: Pearson Education.

Fuenllana, M. D. 1978. *Orphenica Lyra (Seville, 1554)*, 41. New York: Oxford University Press.

Fujishima, T. 1999. Realtime chord recognition of musical sound: A system using common lisp music. In *Proceedings of the International Computer Music Conference*, Beijing, China, 464–7.

Gagnon, T., S. Larouche, and R. Lefebvre. 2004. A neural network approach for preclassification in musical chords recognition. In *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, Monterey, CA, 2106–9.

Goto, M., T. Nishimura, H. Hashiguchi, and R. Oka. 2003. RWC music database: Music genre database and musical instrument sound database. In *Proceedings of the International Conference on Music Information Retrieval*, Baltimore, MD, 229–30.

Hankinson, A., P. Roland, and I. Fujinaga. 2011. The Music Encoding Initiative as a document-encoding framework. In *Proceedings of the International Conference on Music Information Retrieval*, Miami, FL, 293–8.

Heijink, H., and R. G. J. Meulenbroek. 2002. On the complexity of classical guitar playing: Functional adaptations to task constraints. *Journal of Motor Behavior* 34 (4): 339–51.

Hrybyk, A., and Y. Kim. 2010. Combined audio and video analysis for guitar chord identification. In *Proceedings of the International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 159–64.

Huang, X., A. Acero, and H. Hon. 2001. *Spoken Language Processing: A guide to theory, algorithm, and system development*. Upper Saddle River, NJ: Prentice Hall.

Kameoka, H., T. Nishimoto, and S. Sagayama. 2007. A multipitch analyzer based on harmonic temporal structured clustering. *IEEE Transaction on Audio, Speech, and Language Processing* 15 (3): 982–94.

Kashino, K., and H. Tanaka. 1993. A sound source separation system with the ability of automatic tone modeling. In *Proceedings of the International Computer Music Conference*, Tokyo, Japan, 248–55.

Kerdvibulvech, C., and H. Saito. 2008. Guitarist fingertip tracking by integrating a Bayesian classifier into particle filters. *Advances in Human-Computer Interaction* 2008: 1–10.

Klapuri, A. 2004. Automatic music transcription as we know it today. *Journal of New Music Research* 33 (3): 269–82.

Klapuri, A. 2005. A perceptually motivated multiple-F0 estimation method. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 291–4.

Klapuri, A. 2006. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of the International Society for Music Information Retrieval Conference*, Victoria, BC, 216–21.

Lee, C., Y. Yang, and H. Chen. 2011. Automatic transcription of piano music by sparse representation of magnitude spectra. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, 1–6.

Lee, C., Y. Yang, K. Lin, and H. Chen. 2010. Multiple fundamental frequency estimation of piano signals via sparse representation of Fourier coefficients. In *Music In-*

formation Retrieval Evaluation eXchange, http://www.music-ir.org/mirex/abstracts/2010/AR1.pdf. Accessed April 8, 2013.

Maher, R. C., and J. W. Beauchamp. 1994. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America* 95 (4): 2254–63.

Marolt, M. 2000. Adaptive oscillator networks for partial tracking and piano music transcription. In *Proceedings of the International Computer Music Conference*, Berlin, Germany.

Marolt, M. 2001. SONIC: Transcription of polyphonic piano music with neural networks. In *Proceedings of the Workshop on Current Research Directions in Computer Music*, Barcelona, Spain, 217–24.

Marolt, M. 2004. A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Transactions on Multimedia* 6 (3): 439–49.

Martin, K. D. 1996. A blackboard system for automatic transcription of simple polyphonic music. Technical Report 385, Massachusetts Institute of Technology.

Meddis, R., and M. J. Hewitt. 1991. Virtual pitch and phase sensitivity of a computer model of the auditory periphery. i: Pitch identification. *Journal of the Acoustical Society of America* 89 (6): 2866–82.

Meddis, R., and L. O'Mard. 1997. A unitary model of pitch perception. *Journal of the Acoustical Society of America* 102 (3): 1811–20.

Miranda, E. R., and M. M. Wanderley. 2006. *New Digital Musical Instruments: Control and interaction beyond the keyboard*, Volume 21 of *The Computer Music and Digital Audio Series*, Chapter 2: Gestural Controllers. Middleton, WI: A-R Editions.

Miura, M., I. Hirota, N. Hama, and M. Yanagida. 2004. Constructing a system for finger-position determination and tablature generation for playing melodies on guitars. *Systems and Computers in Japan* 35 (6): 10–9.

Moorer, J. A. 1975. On the segmentation and analysis of continuous musical sound by digital computer. Ph. D. thesis, Department of Music, Stanford University, Stanford, CA.

Motokawa, Y., and H. Saito. 2006. Support system for guitar playing using augmented reality display. In *Proceedings of the IEEE and ACM International Symposium on Mixed and Augmented Reality*, Santa Barbara, CA, 243–4.

Nakano, M., K. Egashira, N. Ono, and S. Sagayama. 2009. Hamonic temporal structured clustering with unsupervised model learning for multipitch estimation. In *Music Information Retrieval Evaluation eXchange*, http://www.music-ir.org/mirex/abstracts/2009/NEOS2.pdf. Accessed April 8, 2013.

Newell, A. 1962. Some problems of the basic organization in problem-solving programs. In *Proceedings of the Conference on self-organizing systems*, Chicago, IL, 393–423.

Nichols, E., D. Morris, S. Basu, and C. Raphael. 2009. Relationships between lyrics and melody in popular music. In *Proceedings of the International Society for Music Information Retrieval Conference*, Kobe, Japan, 471–6.

O'Grady, P. D., and S. T. Rickard. 2009. Automatic hexaphonic guitar transcription using non-negative constraints. In *Proceedings of the IET Irish Signals and Systems Conference*, Dublin, Ireland, 1–6.

Oppenheim, A. V., A. S. Willsky, and S. H. Nawab. 1997. *Signals and systems,* 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Otley, M. 1600. *Cittern tablature: manuscript*, f. 22 (seq. 45). Cambridge, Massachusetts: Houghton Library, Harvard University. MS Mus 181.

Ozaslan, T. H., E. Guaus, E. Palacios, and J. L. Arcos. 2010. Attack based articulation analysis of nylon string guitar. In *Proceedings of the International Symposium on Computer Music Modeling and Retrieval*, Málaga, Spain, 285–97.

Paleari, M., B. Huet, A. Schutz, and D. Slock. 2008. A multimodal approach to music transcription. In *Proceedings of the IEEE International Conference on Image Processing*, San Diego, CA, 93–6.

Plumbley, M. D., S. A. Abdallah, J. P. Bello, M. E. Davies, G. Monti, and M. B. Sandler. 2002. Automatic music transcription and audio source separation. *Cybernetics and Systems* 33 (6): 603–27.

Poliner, G. E., and D. P. Ellis. 2005. A classification approach to melody transcription. In *Proceedings of the International Society for Music Information Retrieval Conference*, London, UK, 161–6.

Poliner, G. E., and D. P. Ellis. 2006. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*: 1–9.

Poliner, G. E., and D. P. W. Ellis. 2007. Improving generalization for polyphonic piano transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 86–9.

Quested, G., R. Boyle, and K. Ng. 2008. Polyphonic note tracking using multimodal retrieval of musical events. In *Proceedings of the International Computer Music Conference*, Belfast, Ireland.

Rabiner, L. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77 (2): 257–286.

Raczyński, S. A., and S. Sagayama. 2009. Multiple frequency estimation for piano recordings with concatenated regularized harmonic NMF. In the *Music In-*

formation Retrieval Evaluation eXchange, `http://www.music-ir.org/mirex/abstracts/2009/RS.pdf`. Last accessed April 8, 2013.

Radicioni, D., L. Anselma, and V. Lombardo. 2004. A segmentation-based prototype to compute string instruments fingering. In *Proceedings of the Conference on Interdisciplinary Musicology*, Graz, Austria.

Radicioni, D. P., and V. Lombardo. 2005a. Computational modeling of chord fingering for string instruments. In *Proceedings of the International Conference of the Cognitive Science Society*, Stresa, Italy, 1791–6.

Radicioni, D. P., and V. Lombardo. 2005b. Guitar fingering for music performance. In *Proceedings of the International Computer Music Conference*, Barcelona, Spain, 527–30.

Radisavljevic, A., and P. Driessen. 2004. Path difference learning for guitar fingering problem. In *Proceedings of the International Computer Music Conference*, Miami, FL.

Raphael, C. 2002. Automatic transcription of piano music. In *Proceedings of the International Society for Music Information Retrieval Conference*, Paris, France, 1–5.

Reboursière, L., C. Frisson, O. Lähdeoja, J. A. Mills, C. Picard, and T. Todoroff. 2010. Multimodal guitar: A toolbox for augmented guitar performances. In *Proceedings of the Conference on New Interfaces for Musical Expression*, Sydney, Australia, 415–8.

Rutherford, N. 2009. Fingar, a genetic algorithm approach to producing playable guitar tablature with fingering instructions. Undergraduate thesis, University of Sheffield.

Ryynänen, M., and A. Klapuri. 2005. Polyphonic music transcription using note event modeling. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 319–22.

Sawayama, K., N. Emura, M. Miura, and M. Yanagida. 2006. A system yielding the optimal chord-form sequence on the guitar. In *Proceedings of the International Conference on Music Perception and Cognition*, Bologna, Italy, 1545–50.

Sayegh, S. I. 1989. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal* 13 (3): 76–84.

Scarr, J., and R. Green. 2010. Retrieval of guitarist fingering information using computer vision. In *Proceedings of the International Conference of Image and Vision Computing New Zealand*, Queenstown, New Zealand, 1–7.

Slaney, M., and R. F. Lyon. 1990. A perceptual pitch detector. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, Albuquerque, NM, 357–60.

Smaragdis, P., and J. C. Brown. 2003. Non-negative matrix factorization for polyphonic music transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 177–80.

Stein, M., J. Abesser, C. Dittmar, and G. Schuller. 2000. Automatic detection of audio effects in guitar and bass recordings. In *Proceedings of the Audio Engineering Society*, London, UK.

Sutton, R. S., and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT.

Traube, C., and J. O. Smith. 2001. Extracting the fingering and the plucking points on a guitar string from a recording. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, New Paltz, NY, 7–10.

Tuohy, D. R., and W. D. Potter. 2005. A genetic algorithm for the automatic generation of playable guitar tablature. In *Proceedings of the International Computer Music Conference*, Barcelona, Spain, 499–502.

Tuohy, D. R., and W. D. Potter. 2006a. An evolved neural network/HC hybrid for tablature creation in GA-based guitar arranging. In *Proceedings of the International Computer Music Conference*, New Orleans, LA, 576–9.

Tuohy, D. R., and W. D. Potter. 2006b. GA-based music arranging for guitar. In *Proceedings of the IEEE Congress on Evolutionary Computation*, Vancouver, BC, 1065–70.

Tuohy, D. R., and W. D. Potter. 2006c. *Generating Guitar Tablature with LHF Notation Via DGA and ANN*, Volume 4031 of *Lecture Notes in Computer Science*, 244–53. Springer Berlin/Heidelberg.

Turnbull, H. 1991. *The Guitar from the Renaissance to the Present*, Chapter 2: The Sixteenth Century: The Vihuela and Four-Course Guitar. Westport, CT: The Bold Strummer, Ltd.

Verner, J. A. 1995. MIDI guitar synthesis: Yesterday, today and tomorrow. An overview of the whole fingerpicking thing. *Recording Magazine* 8 (9): 52–7.

Vincent, E., N. Bertin, and R. Badeau. 2007. Two nonnegative matrix factorization methods for polyphonic pitch transcription. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2007/F0_vincent.pdf`. Accessed April 8, 2013.

Wackerly, D., W. Mendenhall, and R. Scheaffer. 2007. *Mathematical Statistics with Applications,* 7th ed., Chapter 7: Sampling distributions and the central limit theorem, 370–7. Belmont, CA: Thomson.

Wang, Y., and Y. Zhang. 2012. Non-negative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering* 99 (preprint): 1–20.

Weeks, O. 2008. *Radiohead - In Rainbows (Guitar Tablature)*, 42. London, UK: Faber Music Ltd.

Yeh, C. 2008. Multiple fundamental frequency estimation of polyphonic recordings. Ph. D. thesis, Université Paris VI.

Yeh, C., and A. Roebel. 2010. Multiple-F0 estimation for MIREX 2010. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2010/AR1.pdf`. Accessed April 8, 2013.

Yeh, C., and A. Roebel. 2011. Multiple-F0 estimation for MIREX 2011. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2011/YR1.pdf`. Accessed April 8, 2013.

Yeh, C., A. Roebel, and X. Rodet. 2005. Multiple fundamental frequency estimation of polyphonic music signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Philadelphia, PA, 145–8.

Yeh, C., A. Roebel, and X. Rodet. 2010. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Transactions on Audio, Speech, and Language Processing* 18 (6): 1116–26.

Zhou, R., and M. Mattavelli. 2007. A new time-frequency representation for music signal analysis: resonator time-frequency image. In *Proceedings of the International Symposium on Signal Processing and its Applications*, Sharjah, United Arab Emirates, 1–4.

Zhou, R., and J. D. Reiss. 2008. A real-time polyphonic music transcription system. In the *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2008/F0_zhou.pdf`. Last accessed April 8, 2013.

Zhou, R., J. D. Reiss, M. Mattavelli, and G. Zoia. 2009. A computationally efficient method for polyphonic pitch estimation. *EURASIP Journal on Advances in Signal Processing* 2009 (729494): 1–11.