Queen Mary University of London

QUEEN MARY UNIVERSITY OF LONDON

MASTERS THESIS

Reproducing Bass Guitar Performances Using Descriptor Driven Synthesis

---

Author:

Dave FOSTER

Supervisor:

Prof. Joshua D REISS

A thesis submitted in fulfillment of the requirements

for the degree of MSc Sound and Music Computing

in the

School of Electrical Engineering and Computer Science

August 22, 2018

Declaration of Authorship

I, Dave FOSTER, declare that this thesis titled, 'Reproducing Bass Guitar Performances Using Descriptor Driven Synthesis' and the work presented in it are my own. I confirm that:

- This report, with any accompanying documentation and implementation, is submitted as part requirement for the degree of MSc in Sound and Music Computing at the University of London.
- It is the product of my own labour except where indicated in the text.
- The report may be freely copied and distributed provided the source is acknowledged.

Signed: 

Date: 22nd August 2018

QUEEN MARY UNIVERSITY OF LONDON

Abstract

School of Electrical Engineering and Computer Science

MSc Sound and Music Computing

Reproducing Bass Guitar Performances Using Descriptor Driven Synthesis

by Dave FOSTER

Sample-based synthesis is a widely used method of synthesising the sounds of live instrumental performances, but their control of such sampler instruments is made difficult by the amount of parameters that control the output, the expertise required to set those parameters and by the constraints of the real-time system. In this project, the principals of descriptor-driven synthesis were used to develop a pair of software tools which aid the user in the specific task of reproducing a live performance using a sampler instrument, by the automatic generation of MIDI controller messages derived from analysis of the input audio. The techniques build on existing work and that of commercially available products. The output of the system is compared to that of expert users hand manipulations, and the results show that the system outperforms the human version, despite the latter taking considerably more time. Future developments of the techniques are discussed, including the application to automatic performer replication.

Acknowledgements

Thank you to my supervisors, Dr. Bob Sturm for the first year, and Prof. Josh Reiss for the second. Thank you also to my SMC cohort at Queen Mary, particularly those from the first year and my fellow part (old?)-timers John Flynn and Nash for the camaraderie and inspiration. To Rob Sneddon and Mike Holt for providing the expert user versions of the music extracts for comparison, and to the participants in the listening tests. To Phil Knights for proof reading and encouragement. Last, but not least, thank you to my wife Rachel, and children Holly & James, for patiently sharing me with my books for these last 2 years.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

Synthesisers have been used in all fields of music since the middle of the 20th Century, and have gradually increased in flexibility, capacity and complexity to the present day. The early uses of synthesisers to replicate or replace orchestral instruments were with the string synthesisers of the 1970's, when a paradigm shift occurred: the cost had reduced and the quality improved to the point where they could be used on a pop record for less than hiring a string section, with no discernible reduction in production quality. It is only, however, since the turn of the century that the technology has become both affordable to the home musician and of sufficient quality to be able to approach the reproduction of the full symphonic palette. This has been aided by two things; the improvement in processing power, which means that complex rendering can be performed in real time; and the cheapness of hard drive space, vast amounts of which are needed to store the comprehensive library of individual sample recordings required by many applications.

Various methods for orchestral instrument synthesis have been employed by such synthesisers, often referred to as "virtual instruments" (VI), including unit generators, wavetable synthesis and additive synthesis, but the most ubiquitous method of recent times has been sampling synthesis. It is on this technique and these technologies that this project will concentrate on.

1.1.1 Sampling synthesis

Sample based synthesisers use short excerpts of recordings of live instruments (or sound effects) which are triggered by the depression of a keyboard key, or remotely by electronic message. Ideally, a dedicated sample recording is available for every key, and with various articulations and at various dynamics. Often, if this is not the case, then neighbouring values (either in pitch or dynamic) are used after some manipulation to get them to the desired values. The more advanced products will combine sounds, automatically loop them (to accommodate notes which are longer than the original recording), and provide alternative recordings for repeated notes, all of which increase the realism and playability.

The earliest sample based synthesisers, such as the Mellotron (first developed in 1963), had the samples recorded on magnetic tape and the depression of the keys on the keyboard would move the tape across a playhead. By the seventies, digital recordings were being used for the sample recordings, and various methods were used to compensate for the relative lack of computing power and storage space, such as using a sample for the first portion of a note, and synthesising the rest using other methods.

Until the 1990s, all sampler were hardware based, after which point they started to become available as software based, and became available as plugin instruments which could be opened in and operated from a digital audio workstation (DAW). This paradigm has continued to the present day, where there is a thriving market for professional as well as home users for sample libraries. These libraries cover the whole range of instruments, orchestral, ethnic, pitched and unpitched, and the recording of the sample components is a source of income for musicians and recording studios.

1.2 Motivations

Synthesised orchestral music technology is used in two fields (among others): for providing orchestral music for media where there is insufficient budget for hire a

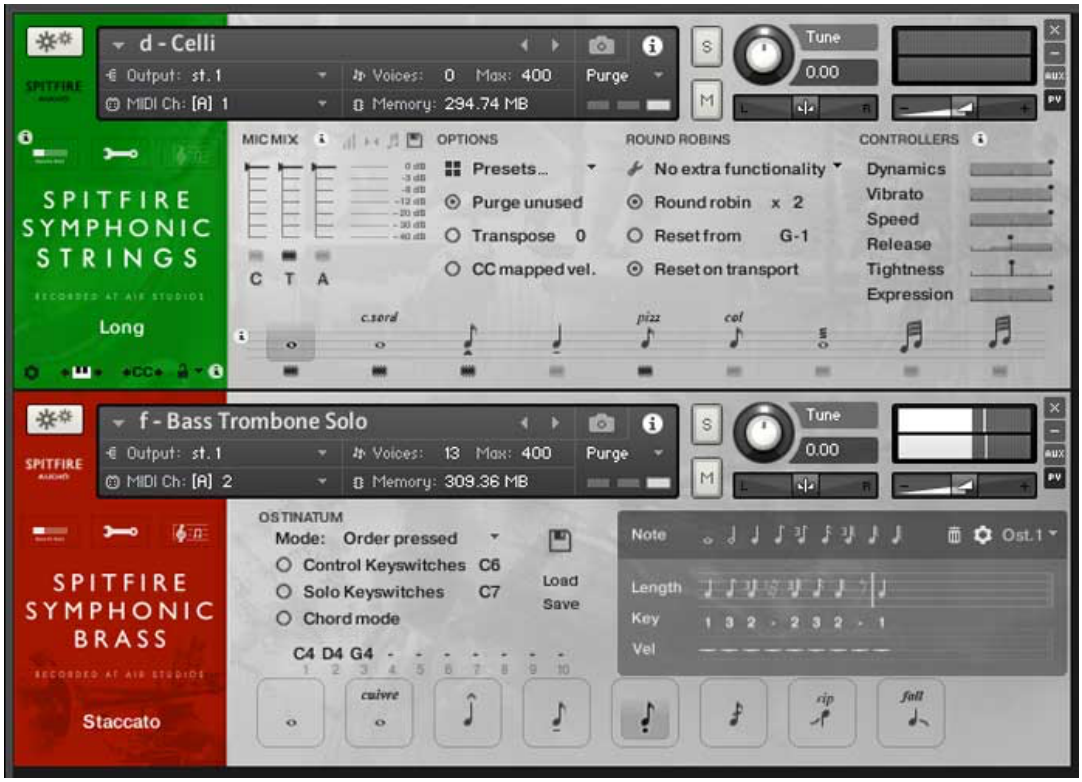

FIGURE 1.1: The UI for the Spitfire Audio sampler instruments

live orchestra; and for making "mock-ups", that is a demonstration of an orchestral composition or arrangement that can be used for feedback or approval without having the expense of using real musicians. In the latter case (which often happens with films, higher end television and pop music), the final version will then be either completely re-recorded by an orchestra, or elements of it replaced by live musicians. This lays testament to the fact that the output of the technology is not currently of sufficient quality to be indistinguishable or a replacement for a genuine orchestral performance.

1.2.1 VI Complexity

As samplers have increased in flexibility, being able to reproduce many different playing techniques and idiomatic gestures within a single instrument's palette, they have also increased in complexity. In the more advanced implementations, every nuance of the synthesised performance can be adjusted across a spectrum of possible

values, allowing for a vast variety in possible outputs. Figure 1.1 shows the UI of two sample based VIs developed by Spitfire Audio[1], a British company who make sample libraries for use in Kontakt[2], the industry standard sample library host plugin. The upper portion shows a "long" string patch (often there are separate "instruments" for long and short notes), which has 9 different playing styles (shown along the bottom with music notation representation) and 6 different continuous controller settings (shown along the right as sliders). These afford the user tremendous flexibility to craft and hone a virtual performance, but there are two problems that arise with these possibilities.

Firstly, there is a level of orchestration knowledge that is needed to successfully operate the interface. Both the knowledge of how the instrument is physically played and of the standard terms for the different playing techniques are required to properly operate it. These may be things that the vast majority of potential users will know: but when such instruments allow for very specific direction, such as the string on which a note is to be played, it is really only those who have some experience with playing or players that would be able to properly exploit the control. Also, decisions about what will sound best when presented with nuanced options (such as between the marcato and staccato options in the Bass Trombone instrument at the bottom), are very difficult to make as they are not only instrument or idiom based: but context specific, depending on tempo, surrounding notes and the surrounding ensemble.

Secondly, the time that needs to be spent honing each musical phrase increases with the amount of control that the user has. If every note has the possibility of a different articulation and dynamic variation, then the user can feel compelled to edit each one: and without reasonably expert knowledge they will only find the most realistic option by chance. The time it can take and the expertise required means that it could have the potential to be comparable with learning to play the actual instrument.

[1] https://www.spitfireaudio.com

[2] https://www.native-instruments.com/en/products/komplete/samplers/kontakt-5/

There are, of course, already processes in place and solutions available to bridge the gap. All sample instruments will have algorithms for selecting the best combination of samples and for blending the transitions between notes, as well as dedicated arrpegiators for generating life-like repeated passages. All music notation software has systems for interpreting the notation for playback: NotePerformer[3] is an application that works within music notation software to manipulate its own sounds to match the symbolic information in the score, without the need for further user interaction. But the output of notation software is no match for a well honed sampler output, which itself is still often not of a quality to reproduce or replace a live performance.

1.2.2 MIDI shortcomings

All samplers use the MIDI protocol for the live control of both note events and settings. MIDI (an acronym of Musical Instrument Digital Interface) is a protocol for communicating between electronic music devices, and was first proposed in 1982. It became the industry standard soon afterwards and, although some additions have been made incrementally, the fundamental modus operadi remains the same. A note on message triggers the start of a note, a note off message stops it; continuous controller (CC) messages change the value of a given parameter; instrument sound changes are triggered by program changes (the latter are rarely used in the modern paradigm, where a single VI represents a single instrument). It operates exclusively in real time, as although sequences can be saved and edited, the messages are always processed as new when the sequence is played. There is no capacity to look ahead or over an entire sequence. This is also the case with alternative protocols, such as the Zeta Instrument Processor Interface and Open Sound Control, which have a richer palette of message types but are still constricted by their real time nature.

The MIDI protocol is excellent at controlling instruments where the notes are triggered by a striking action, such as keyboard instruments, tuned and untuned percussion, and plucked strings (although variation in attack and other factors make the

[3] https://noteperformer.com

latter more problematic). Instruments whose playing technique involves a constant attack (be that by bowing or blowing by one means or another) are much harder to model using conventional MIDI messages. Single notes are easy to model, because they are often just an actual recording of a real instrument playing the note, but as soon as two or more notes are played in a melodic sequence, the VI immediately sounds mechanical.

The crucial difference in the techniques of keyboard and non-keyboard instruments is the ability that the second group have to impart *intent* into their notes. For example, a trumpet player playing a single crochet C will play that C differently to how he/she plays the same C followed by a D, even though the note is of the same pitch and of the same duration. And the C will be different again if it was followed by a different note, or if the articulation on the second note was different. The starts of the notes may be very similar but, during the course of the note, the central pitch, velocity and vibrato will become different to prepare for the next note. Also, there is a transitional period before the next note starts - the larger the interval between the two notes, the more pronounced this will be. Finally, the second note will begin differently depending on the note that precedes it.

The trumpet player makes these changes (automatically and unconsciously) because he/she knows, whether reading music or improvising, what note (if any) is following the current one. He/she has the knowledge of what is to follow, and hence the note is given intent. Conventional MIDI messages, with their simplistic note-on/off messages, are unable to reproduce and hence model this behaviour. Once a note is struck, there is no more information about what (if anything) follows it: just the promise of a note-off at some point. Therefore, all notes will be played the same, regardless of the context. Even if a VI has provision for note transitions, by definition they cannot be adequate as the portion of the transition that happens before the second note cannot be applied, at least at the correct time. The first thing the VI knows of the second note is when it is hit, so cannot anticipate the transition.

Some current systems trigger note transitions by using overlapping notes or the

sustain pedal to signify a slurred passage. Another approach is to add a lag into the playback: NotePerformer adds a one second gap as does the Synful Orchestra, described in the paper by Lindemann, 2007. But unless the system is able to look over an entire musical phrase and make performance decisions based on what it finds, it will always require user intervention to add the missing intent: which, as described earlier, is a time consuming and expert task.

1.3 Proposed system

In order to overcome the complexity of the sampler controls and shortcomings of the MIDI protocol, a radical approach would be for the sampler to use either: an off-line processing model, where the whole piece is considered before samples chosen and manipulated; or an on-line message protocol where information about the length of the note and its context within a phrase given at the note-on event. In both of these cases, the paradigm of the sampler being able to be controlled by a live musician playing the keyboard would have to be abandoned. But it is also possible for improvements to be made within the present operating parameters, by imposing a higher level of analysis and interpretation onto the MIDI protocol, and using its features to produce a humanised performance with an autonomous system. MusicXML[4], a universal standard for describing music notation could be used for the off-line model, or a new format that is not as much tied into the constraints of standard Western notation.

Such a system would require knowledge of how a musician (either specific or generic) would vary their output given any input, either by consulting a corpus of pre-compiled data, or by referring to a similar passage of audio at every node. The development of a system like this is beyond the scope of this project (although some proposals for implementation are presented in the Conclusions (chapter 5), and so this project investigates and implements an intermediary step: reproducing a musical

[4] https://www.musicxml.com

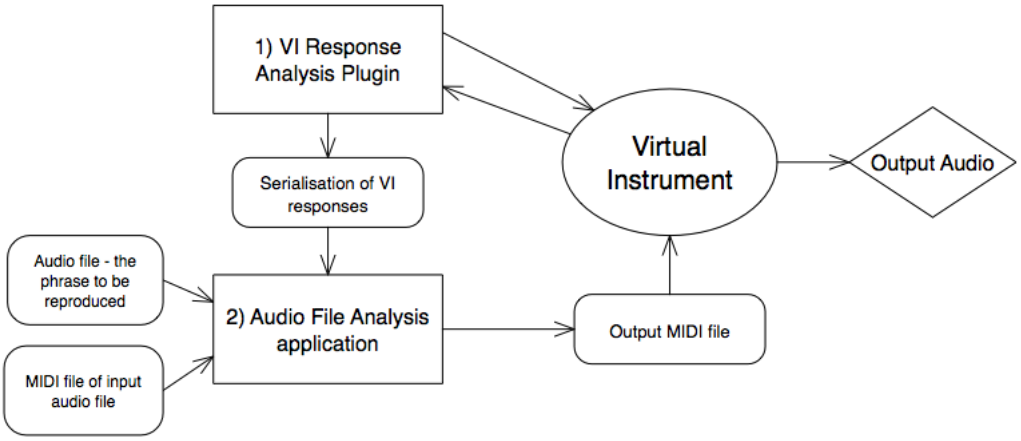

FIGURE 1.2: A flow diagram of the proposed system

phrase of a live performance using a sample based VI. This is to be achieved by working within the constraints of the current paradigm, and ideally will be entirely autonomous, without user intervention (although subsequent editing, re-purposing and manipulation of the output is also desirable).

There are two stages to the proposed system: first, to thoroughly analyse the output of a sample based VI given the full range of input values; secondly, to analyse a pre-recorded musical phrase, and to use the derived information to choose the messages and settings to use, so that the output matches the input as closely as possible. The two stages will be developed as separate applications, so that the initial (and presumably time consuming) task of the full analysis of the VI is separated as it is a single operation (for each VI analysed): the analysis of phrases and subsequent parameter matching is something that will be run repeatedly, and so a dedicated application can be stream-lined for this process. The process is shown as a flow diagram in Figure 1.2.

The inputs to the first application will be user settings defining the range of parameters to investigate, and the response of the VI to each permutation of these parameters as a temporally annotated audio file: the output will be a data structure containing the results of the analysis of each permutation, reduced to a set of values that sufficiently describe each feature. The inputs to the second application will be an

audio file (to be matched), a corresponding MIDI file (giving the notes and timings of the musical content) and the data file outputted from the first system: the output will be a MIDI file to be passed through the VI. The decision to require the user to provide the constituent notes of the audio (rather than the system identify them automatically) was made for several reasons: to aid the autonomy of the system, as the problem of automatic music transcription is difficult and inconsistent, as will be discussed in the Background research (Chapter 2); the system is intended to be a precursor to a system which takes just the note data (along with all the contextual information described above), so introducing this as an input at this stage provides a good platform for future development. The file containing the notes will either be generated by a manual transcription of the recording (in the case of using a pre-existing recording) or from the source music (in the case of using specifically commissioned recordings, or where the sheet music of the pre-existing recording is available). The exact timings of the notes is not a pre-requisite: in fact, one of the features that the system should be able to identify is the deviation of timing from the written notation. The pitches will have to be correct, and in the correct order (although additional notes or gestures between written notes should be tolerated), hence an accurate transcription or note perfect performance is required, which will possibly mean that careful checking and/or pre-processing of the inputs are necessary.

A constraint that was imposed on the system for this project was to restrict it to a single instrument, namely the fretless bass guitar. By limiting the initial development to one particular instrument, the fundamental processes can be identified and iteratively developed, without the requirement to ensure universal compatibility. The specific instrument was chosen for several reasons: the author's familiarity with the instrument, both in its performance and mechanics; the percussive nature of the plucked strings give a somewhat predictable output; the continuous nature of the range of potential output pitches give it an expressive dimension unavailable to keyboard instruments (without extra controller inputs); it is monophonic in nature (the case of double stopping will not be considered), making pitch analysis considerably

more consistent; and the availability of free and relatively easily controllable sample based VIs of the instrument.

Another constraint that was put on the project was the number of MIDI controls that were going to be used. In order to keep the number of permutations to a reasonable level, whilst still having the flexibility to make noticeable variations of the sound, the controls were restricted to note-on, note-off, velocity, MIDI volume, expression and pitch bend. These six values are also the core group which are implemented in most VIs, and will generally have the same output variable mapped to them. In the case of the test VIs used, MIDI volume and expression both mapped to the overall level of the output, but in such a non-linear co-dependant manner that it was decided not to eliminate one or other, but to use them as two dependant variables.

1.3.1 Descriptor-driven synthesis

"Descriptor-driven transformation" is defined in a related paper by Coleman, Maestre, and Bonada, 2010 as being where "input samples are transformed with respect to target descriptors", and they go on to refer to "descriptor-driven synthesis" when referring to the MSc thesis of Mintz, 2007. In both papers, audio features pertaining to pitch, loudness and timbre are extracted and it is the set of these features, rather than the entirety of the input audio data, that are used when constructing a new signal (in their cases, using "mosaicing", where a corpus of samples are directly manipulated in the former or by constructing a new signal entirely in the latter). As this project takes a similar approach of extracting features from audio (both in the collation of data from the sample library and from the audio to be replicated) and using those features to select and manipulate samples (in this case obliquely, via MIDI messages), the term seemed to be a valid way of categorising the methodology employed.

1.4 Use cases

The actual process being automated here, namely exactly reproducing a live performance using a sample based VI, is a slightly contrived one, and not something that professionals would often be required to do when generating synthesised music. However, several use cases were identified where the system could be of great assistance to users.

1.4.1 Use case 1

Alan has some recordings of bass phrases that he wants to change slightly to fit his song: he uses the system to produce a MIDI file that, when played through the VI, accurately replicates the recorded phrase, which he can then alter to fit his song. Notes can be changed in pitch, positioning and duration to fit the new requirements, by moving the corresponding MIDI note in a DAW while retaining the characteristic performance of the original.

1.4.2 Use case 2

Barbara wants to produce some unique sounds from the VI - she records herself playing phrases on her bass guitar, and uses the system to perform a "style transformation". The output MIDI file is played through a bassoon VI and the output has the timbre of a bassoon with the pitch and loudness envelopes of a bass guitar.

1.4.3 Use case 3

Carl wants to emulate the performing style and timbre quality of his favourite bass guitar player. He uses the system to generate MIDI files representing the performances, which he uses both: to evaluate the exact portamento, vibrato and relative velocity of the notes, in order to better emulate them; and to see what dynamic

changes to the loudness are required for his instrument and peripherals to match that of the target.

1.5 Overview

The remainder of the thesis is organised as follows:

- In chapter 2, the background research done for the project is summarised, and the state of the art identified.
- Chapter 3 presents the applications developed for the project, shows how they are used, and discusses how the decisions regarding their design were made.
- In chapter 4, an experiment to test the quality of the results is described, and the results presented.
- Finally, chapter 5 summarises the project, presents conclusions, as well as offering suggestions on how the work can be developed in the future.

Chapter 2

Background research

A wealth of research and literature has been done into subjects that both pre-date and otherwise inform the current work, some of which has been studied and is summarised here. First, the problem of automatic transcription is examined because, even though it is not being attempted in this project, the techniques are still relevant. Then work on audio feature alignment and on aligning MIDI and audio is discussed, then the research specifically on automatic synthesiser control and bass guitar synthesis.

2.1 Automatic Transcription

The problem of automatic music transcription has been considered for a long time, and was referred to by James Moorer, 1977 in his classic paper. He speculates whether a computer can transcribe music as well as a human, concluding that "we can make a program which does this to some extent, given certain restrictions on the music used." To some extent, despite the many advances in both algorithms and in computing power, the same is true over forty years later. In the most recent MIDEX conference (*MIREX 2017 Evaluation Results* 2017), where an annual competition is held to test the state of the art algorithms and techniques for computer transcription, the record of 74.3% accuracy in audio beat tracking was not broken, neither was the record of 72% accuracy in the multi-f0 estimation category. So the best systems are still a long way from being reliable, and performance has not improved greatly over the last decade, causing speculation that a "glass ceiling" has been reached and that

no further improvement can be made. According to Dr. Simon Dixon, 2017, who has worked on different aspects of the problem since the turn of the century, trained humans can score over 80% in both of these tasks. He concludes that:

> "Automatic transcription systems will never be perfect, but a semi-automatic system where the user directs the transcription process and/or corrects transcription errors using a graphical interface is feasible."

The transcription required for this project does have the in-built restrictions that Moorer alludes to (albeit not the exact same ones): the source is a clean recording of a single, known instrument, performed monophonically. A recent paper by Abeßer and Schuller, 2017 describes a system specifically designed for transcribing solo bass guitar music. The report has a good overview of different techniques for the constituent phases, namely onset detection, fundamental frequency tracking and note offset detection, all of which informed the implementation of these elements in this project. Their system is evaluated against the state of the art transcription systems (all of which are generic, and not instrument specific), and it is successful in outperforming them all. They conclude that the "glass ceiling" for automatic music transcription could potentially be broken by the design of instrument specific approaches.

The best score achieved in the paper is 91% (in the note-based evaluation results for score-level evaluation), which still means that one out of every ten notes is mis-identified in at least one aspect. This level was not considered adequate for the system being developed in this project, which led to the decision discussed in the Introduction (Chapter 1) to require the exact notes and approximate timings as an input, which will result in far fewer errors.

2.2 Audio Feature Analysis

An important part of this project is the identification of audio features, and there are many options for what features to extract, how to extract them, and how often

to extract them to be able to fully describe the music. In his seminal book, "Psychology of Music" (Seashore, 1938), Carl Seashore identifies the four classes of musical features by which music can be characterised, classified or described: tempo and timing; velocity, loudness or intensity; pitch; and timbre. In his doctoral thesis, Alexander Lerch (Lerch, 2008) takes each of these features and identifies the main techniques and algorithms for approaching their analysis and detection, with the goal of furthering the field of music performance research by extracting data from various recordings of the same string quartet piece.

In Ramirez, Maestre, and Serra, 2010, the authors seek to develop a system that automatically identifies the performer in a jazz recording. Their analysis is based on the same four classes, and they use selected features to make up a vector by which each note can be described as well as a separate vector describing each transition between notes. They also record a set of descriptors which show the surrounding musical information (the notes preceding and following this one and the structure to which it belongs), which is crucial for the task they are tackling.

Timbre is notoriously difficult to define (except by defining what it is not - the seminal definition from the Acoustical Society of America defines it as anything except pitch and loudness that distinguishes two sounds) and hence to know what to analyse. In Mintz, 2007, the author uses the MPEG-7 standard of five descriptors for harmonic sounds (log-attack time, spectral centroid, spectral spread, spectral variation and spectral deviation) which he uses to develop a synthesis technique. In Sturm, Morvidone, and Daudet, 2010, Mel-frequency cepstral coefficients (MFCCs) are used to derive features related to timbre which are used to distinguish and identify musical instruments. Heise, Hlatky, and Jörn Loviscach, 2010 also uses MFCCs as the defining features of an audio signal, in this case to synthesis speech with the aim of reducing the bandwidth required to transmit it. They analyse a speech signal, reduce it to its set of features, and then reconstruct it by searching a corpus of audio snippets for the closest match for each feature vector and putting them together. The last part of the process is similar to what is being done in this project, and their

method for searching the multi-dimensional database was influential on how it was performed in this project.

In his article describing the technology behind the Synful Orchestra synthesiser, Lindemann, 2007 shows how a slightly different set of parameters are obtained and stored: namely residual pitch (the pitch of the note once noise has been removed), loudness, harmonics and noise. The concept of modelling a musical note as a combination of sinusoids and random noise (which can be characterised by a measure of its stochastic nature) was first introduced in the article by Serra and Smith, 1990, and has proved to be a popular and efficient method of both analysis and synthesis. It was not included in the characterisation of sounds in this project, as it does not form such a crucial aspect of the sound of a plucked as it does with bowed or blown instruments, but it could be included in future versions. The methods in this paper for the acquirement of instantaneous values by the means of windowing, and for peak detection / following are now considered fundamental tools for audio analysis and versions of them were used for this project.

Regarding the analysis of pitch, and the higher level features of portamento and vibrato, the work by Yang, Rajab, and Chew, 2016 was particularly relevant for the analysis, the identification of salient points, and the presentation of the data. The MATLAB implementation of their algorithms and methods, and the associated user interface was influential on the design and architectural decisions for the design in this project. The paper by Lee, 2006 examines the portamento style of master violinists, and identifies three different types of portamento which can be employed by instrumentalists using fretless string instruments. The automatic detection and reproduction of these variants is beyond the scope of this project, but again could be included in future versions.

2.3 Audio / MIDI Alignment

The requirements for the synchronisation of the audio and MIDI were relatively simple in this project, but several pieces of related work in the field proved relevant. Schwarz, 2004 discusses the challenges involved, the requirement for robust onset detection and fundamental frequency identification. It identifies the main fields where the technique may be used, and estimates the accuracy required of the system in each one. The field of musical synthesis was supposed to have the highest accuracy requirement of all. The paper identifies the different paradigms of on-line (hence real time) and off-line: the on-line model, required for such uses as automatic accompaniment, is more difficult because the decisions regarding whether a detected event is the correct match has to be done instantaneously, and without the knowledge of what is to follow; the off-line model, as is being implemented in this project, has more flexibility as it can look backwards and forwards, and do several passes over the entire data before coming to conclusions about matches. This insight was particularly precedent in aiding the onset detection used, and described in Chapter 3.

In Dannenberg, 2006, the author describes a score matching system that uses the results to generate audio output using concatenate synthesis. The system has a similar methodology to this project, in that it analyses a corpus of data, creates a database, and uses the best matches within that to generate output, but differs in that its requirement for matching is not so rigorous as it works under much less prescribed parameters.

The PhD thesis of Colin Raffel, 2016, regards the large-scale matching of MIDI to audio, involving matching entire songs to MIDI files, which again has a lower standard for exact note timings than required in this system. There are aspects of his system and questions brought up in the paper that were considered for use in this system, or example: when a note deviates from the target temporal position, could that be inferred to be the first indication of a slightly different performance tempo?;

and whether machine learning techniques could be used both for evaluation and improving performance. In the case of the former, it was decided to only allow MIDI files that corresponded exactly with the performance in terms of tempo (although that could be retroactively added): and in the latter, the development time allocated to the project was deemed to be better spent on the MIR requirements, and that machine learning techniques could be added later, as discussed later in this chapter and in Chapter 5.

2.4 Synthesiser Control

The automatic control of synthesiser parameters is an established area of research, and established practices and conventions were revealed upon researching the literature. In Mitchell, 2012, he describes the field of evolutionary sound matching, which is an iterative approach to finding matching parameters: a target sound is analysed, and then the difference between the features derived and that of the synthesiser are minimised by repeatedly making incremental changes to the settings. The output in this case are the settings for a native FM synthesiser, and not for settings on a sample based VI, but he does attempt to replicate the sounds of some orchestral instruments which has limited success due to the limitations of the synthesiser.

In Riionheimo and Välimäki, 2003, the best-known evolutionary computation algorithm, the Genetic Algorithm, is used to match target sounds using a plucked strings synthesiser. This is similar to the requirements of this project, but here they are using the derived parameters to control a physical modelling algorithm which generates the string sound without a pre-recorded sample. Walker and Whalen, 2013 also uses a Genetic Algorithm to choose the best fit synthesiser settings, but here the input is real time data from a live musician playing a guitar.

The paper by Heise, Hlatky, and Jörn Loviscach, 2009b is very relevant, as they are also tackling the automatic reproduction of recorded sound by setting synthesiser settings. They are matching a generic sound to the settings of a generic synthesiser,

not specifically a sample based VI with the unique complications associated. As in the Sturm, Morvidone, and Daudet, 2010 paper, they use MFCCs to categorise the sounds, and use several iterations of the same VST synthesiser running off-line (and hence running without the time constraint of one opened in a DAW). They allow for large amounts of individual settings, in the region of 50 (so far more than are being used in this project), which requires a highly optimised search strategy for finding the best combination of settings. They test their system on drum, synth and piano sounds, and show the comparisons between the system's output and the original sound, but do not perform any listening tests to verify how successful the method is.

In another paper by Heise, Hlatky, and Jörn Loviscach, 2009a, the authors again analyse the output of a synthesiser given iterative parameter changes, but this time with the goal of finding the meaningful start time of each sample, so that extremely precise temporal placement can be achieved. Of particular interest to the development of this project was the section on matching velocity values to output level, to allow the precise altering of levels across a range of notes where the level differences between neighbouring velocity values is not necessarily the same for each note. A similar matching of levels given velocities is part of the system in this project, and they also implement a dynamic alteration of the MIDI volume setting to match the level throughout the note, not just at the start. Again, they present no data regarding how well their system performs, either in listening tests or graphically.

2.5 Bass Guitar Synthesis

Although the bass guitar is not directly being synthesised in this project, recent work on current techniques was consulted in case there was any relevancy. In the paper by Kramer et al., 2012, the bass guitar is synthesised by using digital waveguide synthesis. They model five different plucking styles (right hand) and 6 different expression styles (left hand). In listening tests, their model performed better than highly compressed versions of a live musician. The 11 different playing techniques

(and permutations thereof) would require some expert setting, and manual alteration throughout a musical passage to make the same variations that a live (uncompressed) musician would make: it would benefit from some automatic parameter settings as being developed for this project.

Some of the same authors collaborate again on the paper Schuller, Abeßer, and Kehling, 2015, where they analyse the sound of a bass guitar and derive features from it: noticeably they estimate the plucking style and expression style (with the same values as before) as well as which string a note has been played on. These parameters were not considered in this project, but could certainly improve the performance if they were matched with corresponding settings in an advance VI, such as the MODO Bass[1].

Finally, in the system described in Tremblay and Schwarz, 2010, the bass guitar is used as a live controller for a synthesiser: in real-time, the audio signal is deconstructed into audio fragments which are then dynamically mapped to appropriate values within the corpus of the synthesiser, and the output reconstructed. The authors state that this enables "recycling the virtuosity acquired on the electric instrument", which is close in intention to the second of the use cases described in Chapter 1.

[1] http://www.ikmultimedia.com/products/modobass/

Chapter 3

Implementation

3.1 Introduction and Working Environment

The implementation of the project was split into two separate applications: “88x127” for generating the MIDI file used in testing the VI, and for analysing the resultant audio file; and “Smarter Sample Selector” for analysing an input audio file and generating an corresponding output MIDI file.

MATLAB was used for prototyping and early proof of concept implementations, and the App Designer application within MATLAB used to bring together the various elements and to provide a GUI. All of the algorithms and processes were coded “from scratch”, to ensure that they were all optimised for the specific requirements, had the necessary inputs and outputs, and for the author’s deeper understanding. The exception is the MATLAB code for reading and writing MIDI files, which was based on the open source library written by Ken Schutte[1]. The function for writing a MIDI file, which was originally restricted to simple note-on and note-offs, was developed to allow it to encode arrays of pitch bend and continuous controller messages. Existing MATLAB packages, such as the MIR Toolbox, developed by members of the Finnish Centre of Excellence in Interdisciplinary Music Research[2] were investigated for extracting the audio features, but felt to be both too general (and not catering to

[1] http://kenschutte.com/midi

[2] https://www.jyu.fi/hytk/fi/laitokset/mutku/en/research/materials/mirtoolbox

the specific requirements of this project) and to be bypassing some of the project's challenges.

A simple VI was chosen for prototyping and experimentation, namely the "Fretless Bass" instrument, which is a downloadable additional instrument for the Logic Pro X[3] EXS sampler plugin. The author recorded some short extracts of simple musical packages (including extremes of portamento and variations in attack) and created corresponding MIDI files of the notes and approximate timings which were used to develop and test the system.

3.2 88x127

The "88x127" application gives the user a tool for comprehensively evaluating the output of a sample based VI. The application is presented as a single window with four tabs, which the user works through sequentially to set the parameters and move through the workflow, ending with viewing the analysis results.

3.2.1 User Interface

The first tab is shown in figure 3.1, where the user first chooses the range of notes and velocities to be considered. The title of the application refers to the number of permutations of the full possible range of a piano, with 88 keys and 127 velocities (a velocity of 0 is an alternative to a MIDI note-off command). A compromise has to be made between the complete-ness of the data and the processing time / data size that is required. The values shown in figure 3.1 were found experimentally to be optimal in the case of the fretless bass VI used for experimentation: the range of MIDI note numbers from 28-67 cover the range from the low E string up to the 22nd fret of the G string, which is the complete range of a standard 4-string instrument; the range of velocities from 67-127 in steps of 5 covers the usable range of the VI, where velocities lower than 67 produce un-pitched percussive effects, and where neighbouring

[3] https://www.apple.com/uk/logic-pro/

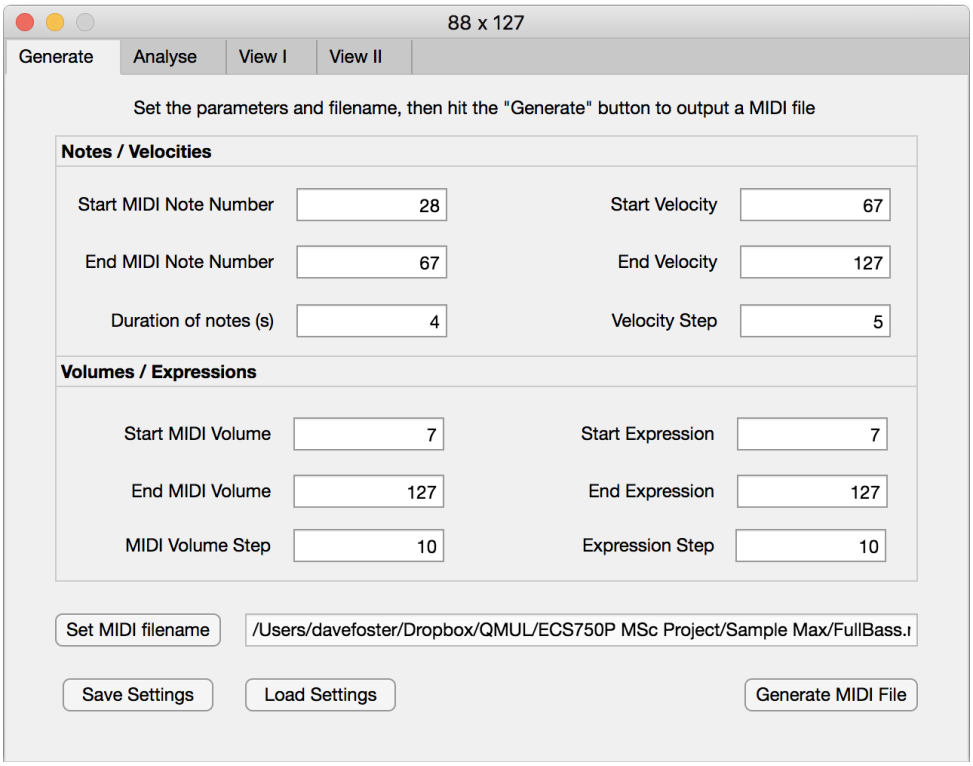

FIGURE 3.1: 88x127 Tab 1 - Generate

velocities were found to duplicate the same audio playback. The duration of the note to be tested is also defined here.

The user then has to make a similar decision regarding the range of MIDI volume and expression settings. These two settings were found to control the amplitude of the output, and in such a non-linear way that made interpolation unreliable. Hence, it was found that a nearly full range of note number / velocity / volume / expression permutations was necessary to adequately predict and control the output. Experimentally, very short notes of 0.03 seconds were found to be all that was needed to ascertain the amplitude of the output given a set of values, due to the percussive nature of the plucked strings recordings being tested. In figure 3.1, both volume and expression are set to ranges of 7-127 in steps of 10, which was found to cover the usable range of values (between which interpolation could be used). These settings generate $13 \times 13 = 169$ possible values for each note / velocity permutation.

The user then generates a MIDI file which then has to be opened in a DAW and

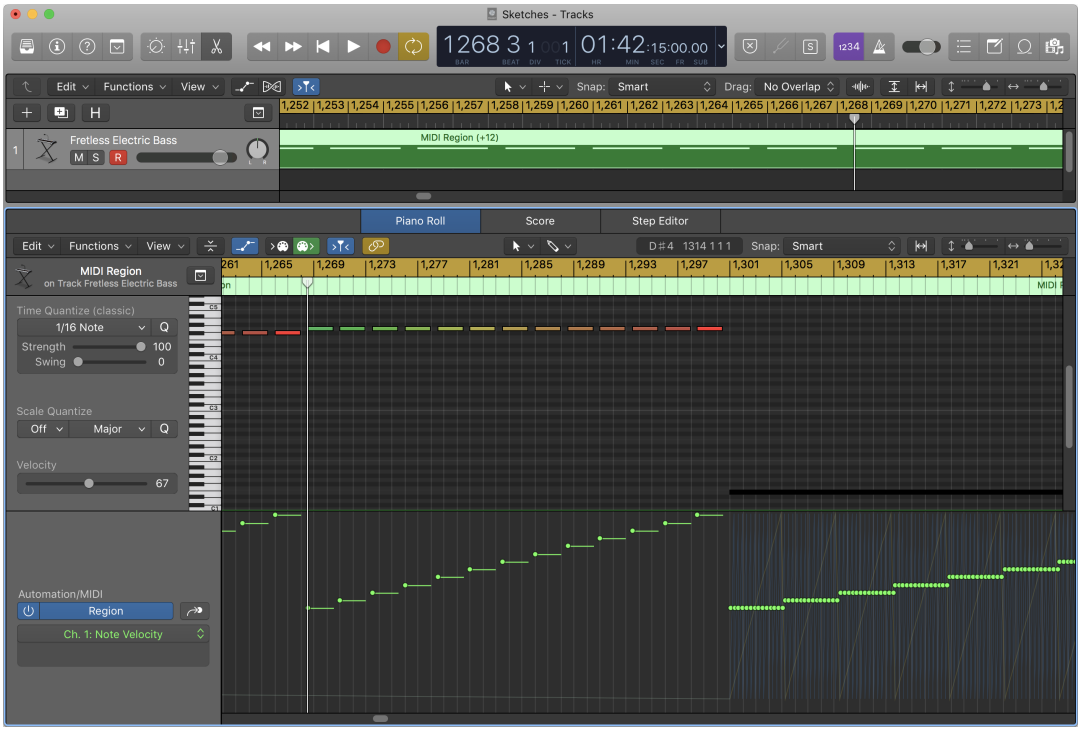

FIGURE 3.2: 88x127 output MIDI file opened in Logic X

put onto a channel strip with the VI under testing. The MIDI file generated using the settings in figure 3.1 is shown opened in Logic X in figure 3.2. Note that an octave transposition has been applied to compensate for that imposed by the VI (which matches the usual written transposition of the Bass Guitar). The MIDI file consists of 4 second notes (separated by 1 second gaps) of every chosen note at every chosen volume (at set central value of 95 for volume and expression), followed by the 0.03 second notes at every one of the note / velocity / volume / expression permutations described above. The total length of the file is 1 hour 56 minutes 34 seconds, which takes 3 minutes 05 seconds to export using the offline bounce feature of Logic X.

The user can then return to 88x127, move to the second tab (shown in figure 3.3), select the bounced file and press “Analyse” to begin the analysis process. An additional setting of the number of analysis points per second is available, which sets the number of landmarks are recorded in the output data. This affects the size of the output, but not the running time of analysis, as all points in between are still calculated for visualisation.

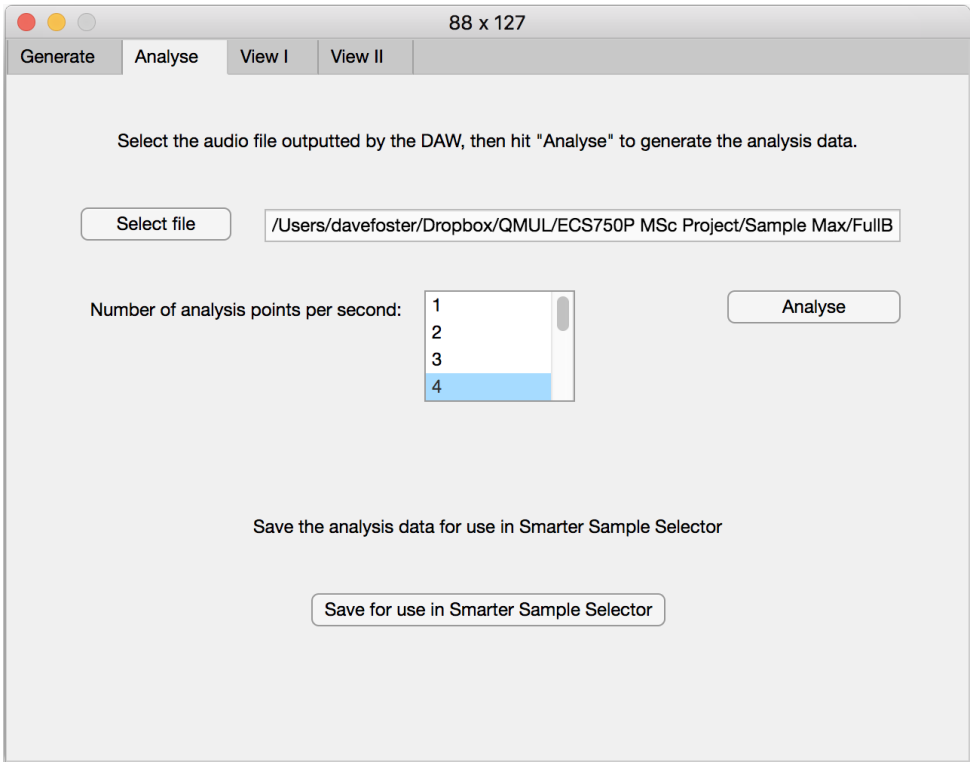

FIGURE 3.3: 88x127 Tab 2 - Analyse

As an aside, alternative methods for collecting this data were considered, and there were two main other options. The first was to use the internal data file used by the sampler instrument, which sets which audio samples are used for any given note / velocity input. The sample file accessed by any permutation could be found on the hard drive and analysed, which would have eliminated the need to open the sample library in a DAW. This was discounted as it would have required reverse-engineering the data format, and was thought not to guarantee the exact output from the VI, given other possible user settings and application specific variables. The other option was to open the VI in the background of the analysis application, as described in the paper by Heise, Hlatky, and Jorn Loviscach, 2009. This would also eliminate the need to separately open the DAW and gives the possibility of a more efficient implementation because of allowing the analysis to be done only when needed on a case to case basis instead of all at once. This was discounted due to the difficulty of passing MIDI messages to a VST instrument opened in MATLAB, and because

the audio would be created in real time (and not offline), meaning slower overall performance. Hence the "brute force" approach of "playing" all notes as described above was settled on as, although cumbersome (requiring the user to go through a multi-step and multi-application workflow) it guarantees the genuine output of the VI, as well as front-loading the processing time to a time-consuming one-off event which generates a data file that can be quickly and cheaply accessed ad infinitum.

3.2.2 Analysis

The analysis begins by separating the audio file into separate notes, using the values which were used to create the original MIDI file. An error is thrown if an unexpected pitch or duration is found, which would mean that the incorrect audio file had been loaded. Each of the notes this then analysed for features pertaining to the three categories of loudness, pitch and timbre. Data arrays for the general information and the loudness / timbre landmarks are initialised.

Loudness

The audio data for each note is first searched for the greatest absolute value, for which the time and value is saved into the general data array. The audio is then parsed for loudness using the RMS (root mean square) formula:

$$x_{rms} = \sqrt{\frac{1}{n}(x_1^2 + x_2^2 + \cdots + x_n^2)}, \tag{3.1}$$

where $\{x_1, x_2, \ldots, x_n\}$ are the samples within the supplied 1024 sample window, and using a 512 sample hop size. The maximum value and time are saved in the general data array. The array is then smoothed by passing through a moving average filter, and the hops corresponding to the pre-set landmark times are normalised by dividing by the maximum value before saving in a separate array.

Pitch

Various pitch detection approaches were experimented with, in both the frequency domain (phase vocoder, interpolated peaks) and the time domain (autocorrelation, YAAPT), but the most efficient and accurate was the YIN estimator proposed by Cheveigne and Kawahara, 2002. The fact that the target pitch is known means that the search range can be greatly reduced and octave errors can be eliminated. More advanced variations of the approach, such as PYIN (Mauch and Dixon, 2014) which tracks various pitch candidates before deciding on the most likely, showed no improvement within the confines of the problem. A window of 2048 and hop size of 512 were used except in the case of the notes below bottom G, for which that window size is insufficient to capture a full period of the oscillation. Here, the window size was doubled. A novel variation to the algorithm was used to reduce computation time, by restricting the range of the difference function to use the upper frequency limit (chosen to be 1 tone above the target pitch), instead of half of the window size. So formula (6) from the YIN paper is replaced by:

$$d_t(\rho) = \sum_{j=1}^{w'} (x_j - x_{j+\rho}), \tag{3.2}$$

where w' = floor(fs / f0'), fs is the sample rate and f0' is the upper frequency limit. This variation decreased the processing time by approximately 50%. Features are then extracted from the pitch curve: the mean value, the initial pitch and time to expected value (representing the portamento up / down to the note), the maximum absolute deviation from the mean, and the first (if any) time of an absolute deviation from the mean that exceeds a set threshold value.

Timbre

Defining, capturing and qualifying timbre features of an audio signal is a notoriously difficult task, but various approaches are available and widely used. The MDS studies of timbre perception, such as those by Krumhansl, 1989 and McAdams et al.,

1995 suggest that log attack time (already measured with the loudness evaluation), brightness (represented by spectral centroid) and spectral dynamics (represented by spectral flux) are the main features. The spectral centroid of a window of audio data is the weighted average of its frequency spectrum, and the spectral flux is a measure of the amount that the frequency spectrum changes from the previous window. These two values are used to measure the timbre of the note, and are calculated with the same window size and hop as for the pitch detection. The spectral centroid values are normalised by dividing by the target frequency, so that they are relevant regardless of the exact pitch. Again, the hop values corresponding to the landmark timings are saved into results arrays.

Another measure of timbre that was experimented with was that of inharmonicity, which is a measure of the degree to which the overtones of the fundamental frequency deviate from the exact multiples expected in the harmonic series. The equation from Fletcher, 1964 (which evaluates piano strings) was used, in a similar fashion to that described in Dixon, Mauch, and Tidhar, 2012. The values derived were, however, inconsistant and found not to be suitable for either aiding to characterise the timbre or for selecting a similar note. The values are displayed to the user, but not used in any further capacity.

Once all of the longer notes are evaluated for each of these qualities, the shorter values are each evaluated for their maximum loudness and these values saved in a 4D array. The user can then save the resulting data into a .mat file, which can be used in the other application.

3.2.3 Display Results

The final two tabs of the application show the results of the analysis. Figure 3.4 shows the third tab, which displays graphs for the currently selected note / velocity permutation of the amplitude, loudness, timbre and pitch. The user can change the current pitch and velocity being viewed, as well as listening to the selection. Figure

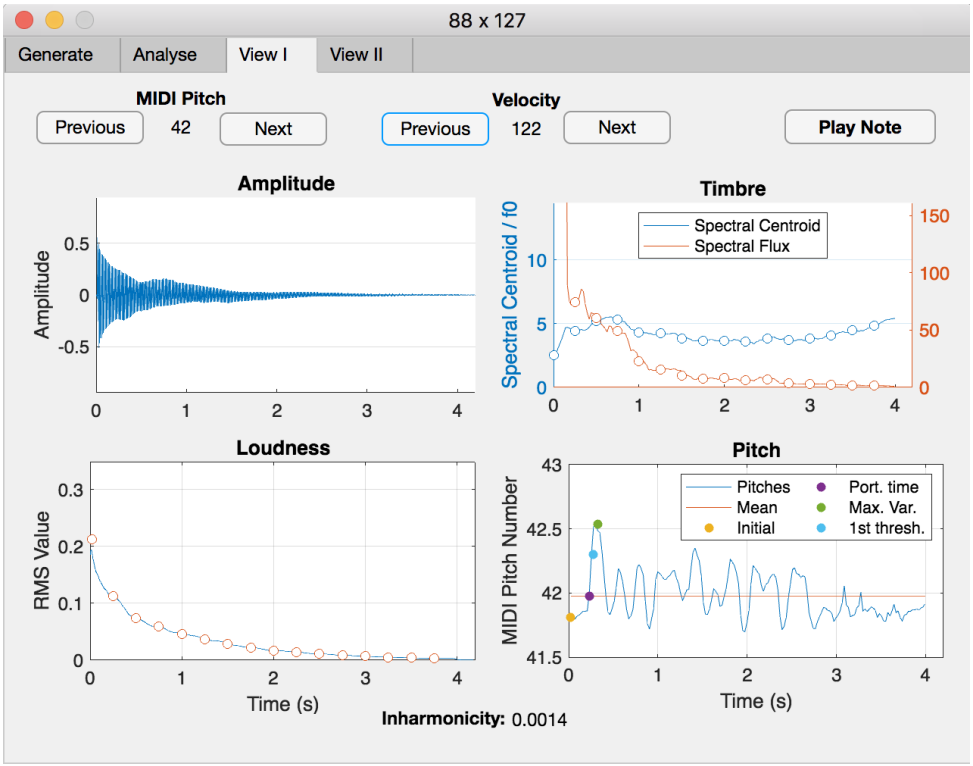

FIGURE 3.4: 88x127 Tab 3 - View I

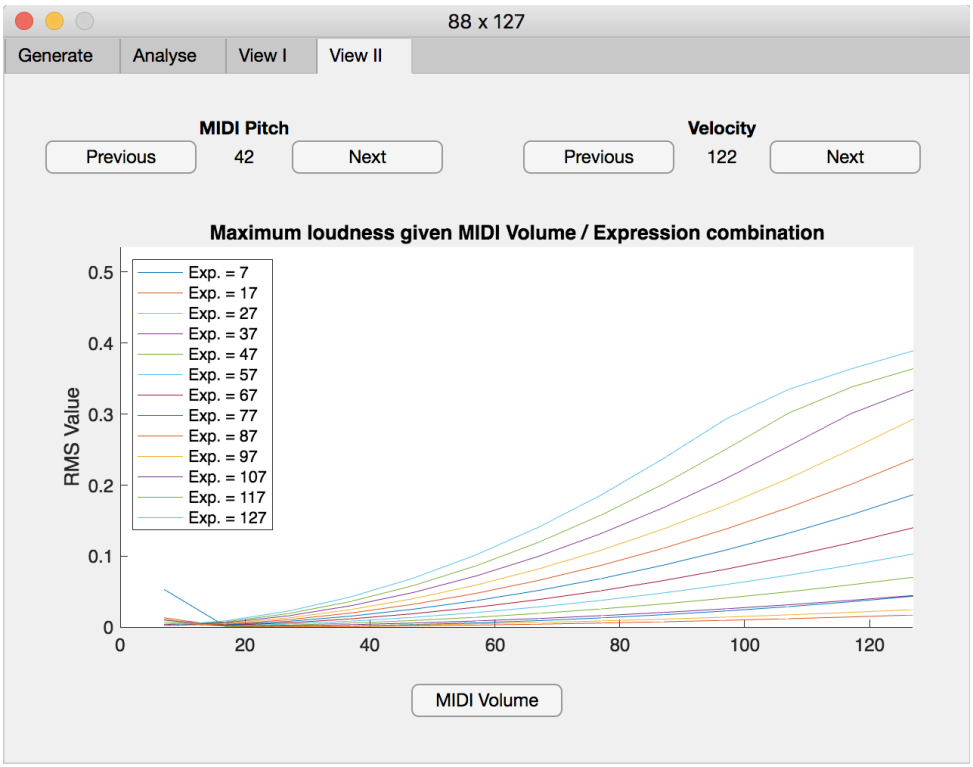

FIGURE 3.5: 88x127 Tab 4 - View II

3.5 shows the final tab, where the results of the MIDI volume / expression permutations is displayed on a graph. The x axis can be toggled to represent either the volume or expression, and the other of the values shown as a coloured series of lines.

The visualisations of the data prove to be useful in choosing the ultimate best (and most efficient) range of values, and were used to eliminate values and settle on the values seen in figure 3.1. Further refinement could be done on the basis of the values seen in figure 3.5, where combinations of volume and expression where values are lower than 20 yield such low output that they could assumed to be useless.

3.3 Smarter Sample Selector

The Smarter Sample Selector tool uses the results of the 88x127 analysis to shape a MIDI performance to best match a given audio performance. The application also has a single window UI presentation, with two tabs. The first tab, shown in figure 3.6 requires that the user provides the location of three files: the MIDI file containing the notes and approximate locations of the performance; the audio file of the performance; and the data file saved in 88x127. When the application confirms that the contents of these files are appropriate, the "Process" button becomes active and the user can trigger the analysis process.

3.3.1 Analysis

Similar to the 88x127 application, Smarter Sample Selector analyses each note and extracts corresponding data points from it, before using that information to pick and place the most appropriate MIDI events, and further sculpt the synthetic performance using MIDI controllers.

Onsets

The main difference between the analysis here and before is that in the previous stage, the exact start and end times of the notes was known. So, to accurately reproduce

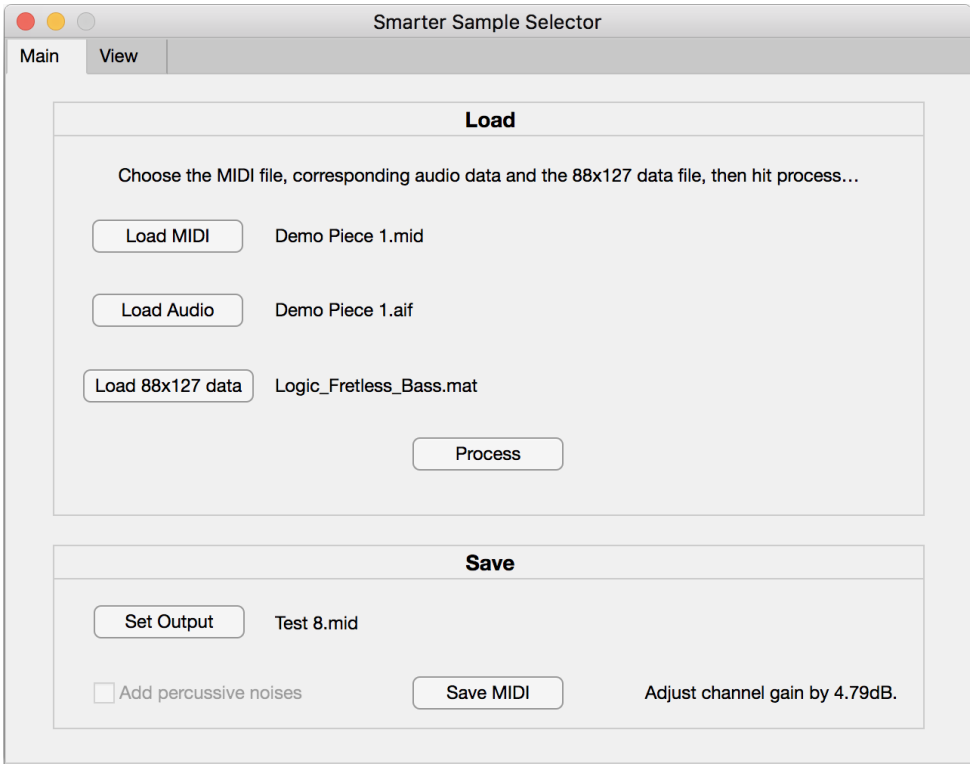

FIGURE 3.6: Smarter Sample Selector Tab 1 - Main

the start time and duration of the notes (and not rely on the MIDI data, which either will have been gathered by aural transcription or from the sheet music from which the musician played), the onset of each note must be found. The percussive nature of the plucked string makes this easier than with an instrument with a softer onset. Various methods were experimented with, both in the time domain (simple envelope follower, RMS energy) and frequency domain (high frequency content, spectral flux, phase deviation), which all require some fixed threshold values to be set. The system requires autonomy in finding these values, and any lack of robustness would cause errors in the subsequent analysis, all of which rely on accurate start times for comparison. Some difficulty was had in finding values that were reliable on all of the test recordings, but a solution was found by using the MIDI information to provide a framework of likely start times. The spectral flux was the best performer for finding clear peaks, and so these values were windowed by a triangular window, with the centre at the MIDI note position, thus weighting peaks closer to the likely start time

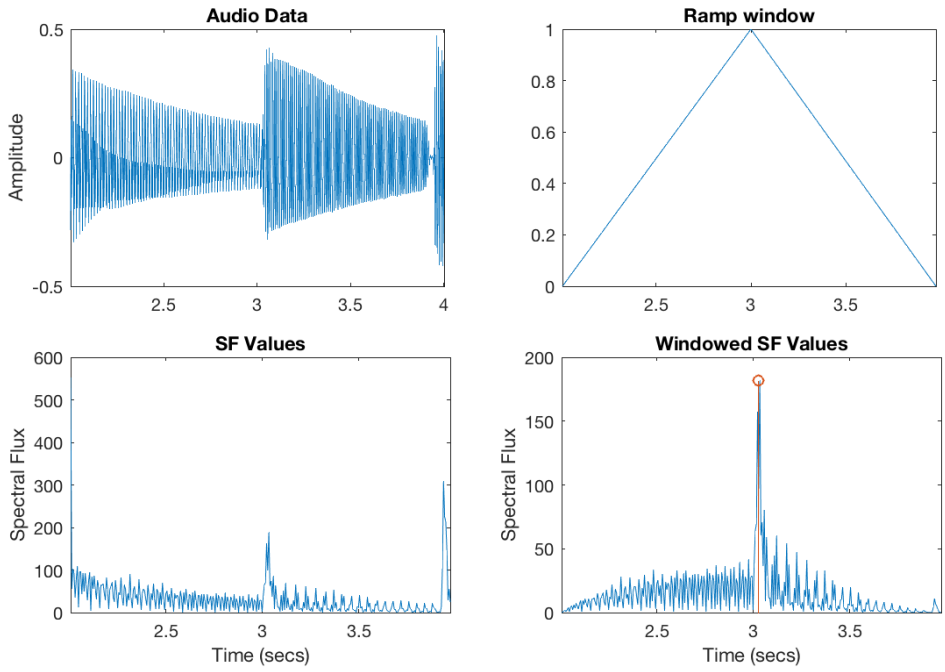

FIGURE 3.7: Windowing the spectral flux values

higher. This proved to be reliable in the majority of cases, and the exceptions were when the notes were either very faint or very short. Figure 3.7 shows the original spectral flux values and the windowed values, which clearly show the peak corresponding to the actual note onset. These onset values are then used to segment the audio into separate notes, each ending just prior to the subsequent note starting.

Another pass is done over the input audio to identify any other noticeable onsets that may correspond to rhythmical percussive additions. The spectral flux data is used again, and the peaks already identified as being note starts removed. A reasonably generous peak threshold is used, along with a proximity threshold that eliminates events that are too close to each other, or the "real" notes, to construct an array of candidate times that could be used as part of the final MIDI output.

Loudness

The segmented notes are then subjected to the same RMS loudness evaluation as before, and the same regularity of landmark time values used to collate the array of values describing the envelope. A minimum threshold is set, below which a note

is deemed to have ended: if this threshold is not breeched before the next note is sounded, then the end of the note remains just prior to the next start. A further nuance which was investigated but not fully implemented, was to evaluate the derivative of the RMS curve. If this is seen to move sharply downwards, this could be evidence that the note was muted (either by the right hand or by the release of pressure by the left hand), instead of being allowed to naturally decay. As many sample libraries play a muting sound when a note-off is registered, this could be exploited to make the ends of the notes more realistic.

Pitch and timbre

With the full information of the note start and duration, the analysis of the pitch and timbre can be performed as before. No features are extracted from the pitch curve in this instance, as all of the values will be represented in the final output. Experiments were done during the prototyping phase to extract a series of descriptors of the mean pitch and depth / rate of the vibrato at regular intervals during the note. This was inspired by the AVA system developed by Yang, Rajab, and Chew, 2016. These values were then used to control a low frequency oscillator in the sample instrument, controlling the pitch deviations. Although the results were virtually indistinguishable from those using the exact pitch curve, this approach was abandoned when it became clear that the amount of data points needed to recreate the curve in this manner was not substantially lower than the number of points it was replacing. Also, the pitch bend controller was needed (to set the centre pitch in each interval), so there was no reduction in the complexity of controls used.

The curves of both the spectral centroid and spectral flux are obtained as before, and the landmark points at the pre-determined regularity collated in an array.

3.3.2 Constructing the performance

With the analysis complete, the application can now move onto the task of constructing the MIDI file. For each note, the start and end time of the corresponding MIDI representation is set to the values derived in the onset and loudness analysis. To allow the greatest flexibility in choosing appropriate MIDI volume and expression values, the input audio values are scaled to correspond to the lowest velocity values in the 4D loudness array. This is done by finding the note with the greatest loudness value, and calculating the ratio between that and the loudness of the lowest velocity for that note value. That ratio converted to a dB adjustment and is displayed to the user in the initial tab after analysis, so that they can adjust the gain value of the channel strip to exactly match the loudness of the original. The results pertaining to timbre, loudness and pitch are then used to hone the performance.

Timbre: picking the best fit velocity

The velocity of the note is chosen by comparing the timbre values at the landmark points of the input audio with values derived at all available velocities for the pitch of the note in question. A cluster analysis algorithm (k-means nearest neighbour search) is performed, using the landmark values of the input as the search vector, and the array of values for appropriate velocities as the candidate array. The pitch values of the members of the candidate array are also examined: any member with a pitch outside the pre-determined threshold (after any initial portamento) is excluded from the search process. The value with the smallest closeness value is identified, and the corresponding velocity chosen for the given note.

Loudness: shaping the RMS curve

As the range of the loudness values has been scaled to the range of the 4D loudness array, the system can pick the volume and expression values that will exactly match the initial loudness of the input note. Initially setting the volume to maximum (127),

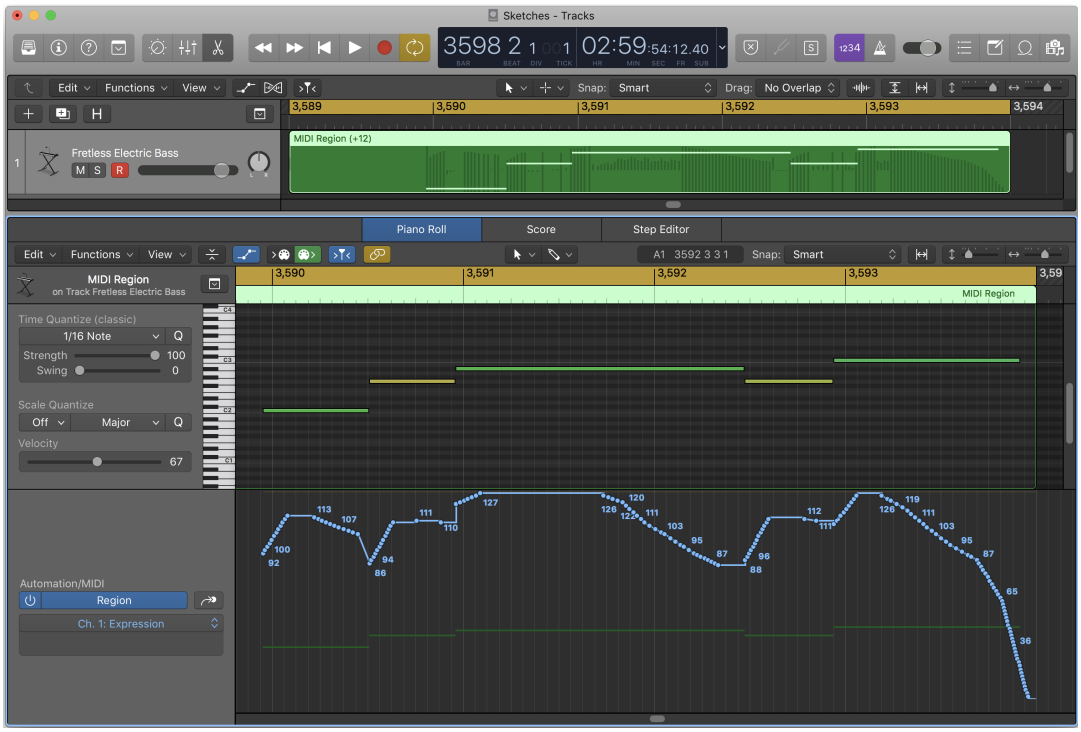

FIGURE 3.8: Expression curve in output MIDI file

the expression value that yields the closest match is found, and then interpolation used to get the exact value. The corresponding RMS landmark values of the input note and the chosen velocity are then compared for the adjustments that are needed while the note is playing for the latter to best match the former. It is assumed that scaling the initial expression value of a note will affect the loudness of the rest of the note comparatively, for which further experimentation would be needed to confirm. Based on this assumption, the expression value needed to correct any discrepancies between the values at each landmark point is calculated, and a smooth step generated between each value. Figure 3.8 shows how this curve manifests in an outputed MIDI file, and figure 3.9 compares how closely the output matches the input when the expression is fixed compared to using the dynamic curve matching.

Pitch: compensating and shaping

The pitch values of the input audio and of the chosen pitch / velocity combination are used to create a curve of MIDI pitch bend values so that the output matches the input.

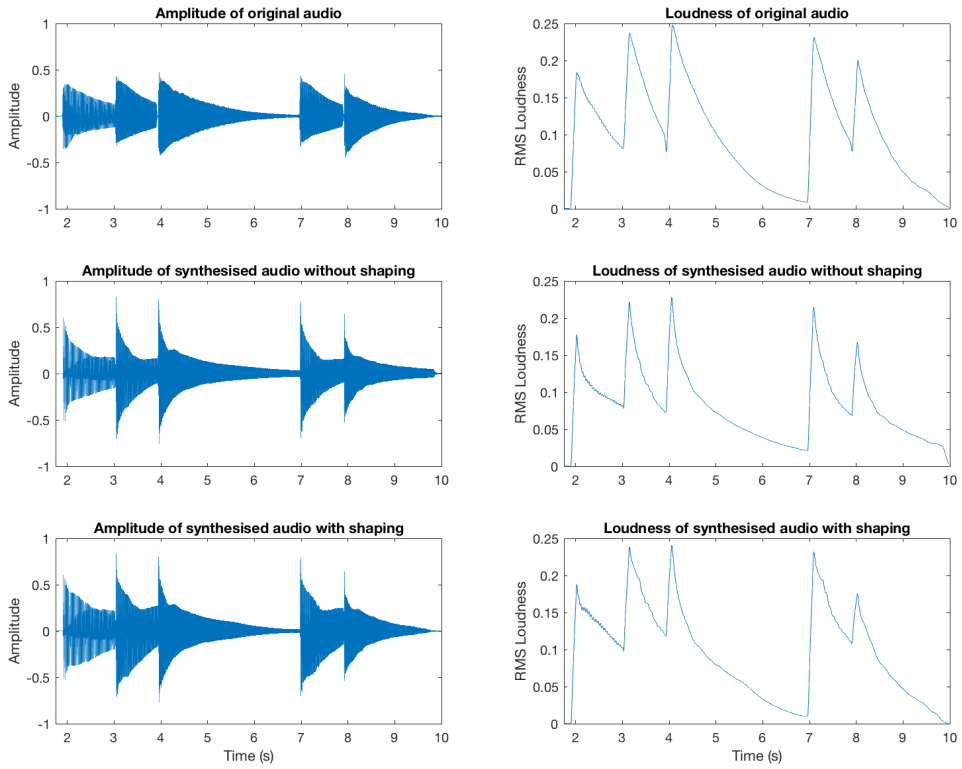

FIGURE 3.9: Effect of loudness matching

First, the main pitch variations of the chosen sample are compensated for by creating a curve that cancels them out: the mean pitch value of the sample is corrected by a constant pitch shift in the opposite direction; then, any initial portamento is corrected by a ramp starting in the opposite direction and ending at the mean pitch at the same temporal point as the input does. Finally, the pitch variations in the input audio are converted into a pitch bend array, which is added to the compensation array to yield the output. Figure 3.10 shows such an curve in an outputted MIDI file.

3.3.3 Display results and output MIDI file

When the analysis is complete, the user can also move to the second tab, shown in figure 3.11, to see the results of the analysis. The same four graphs as on the third tab of 88x127 are shown with the results of the amplitude, loudness, timbre and pitch analysis for the chosen note, as well as a similar button to play the note in isolation. Below these, the parameters of the chosen velocity value are shown, with a graph of

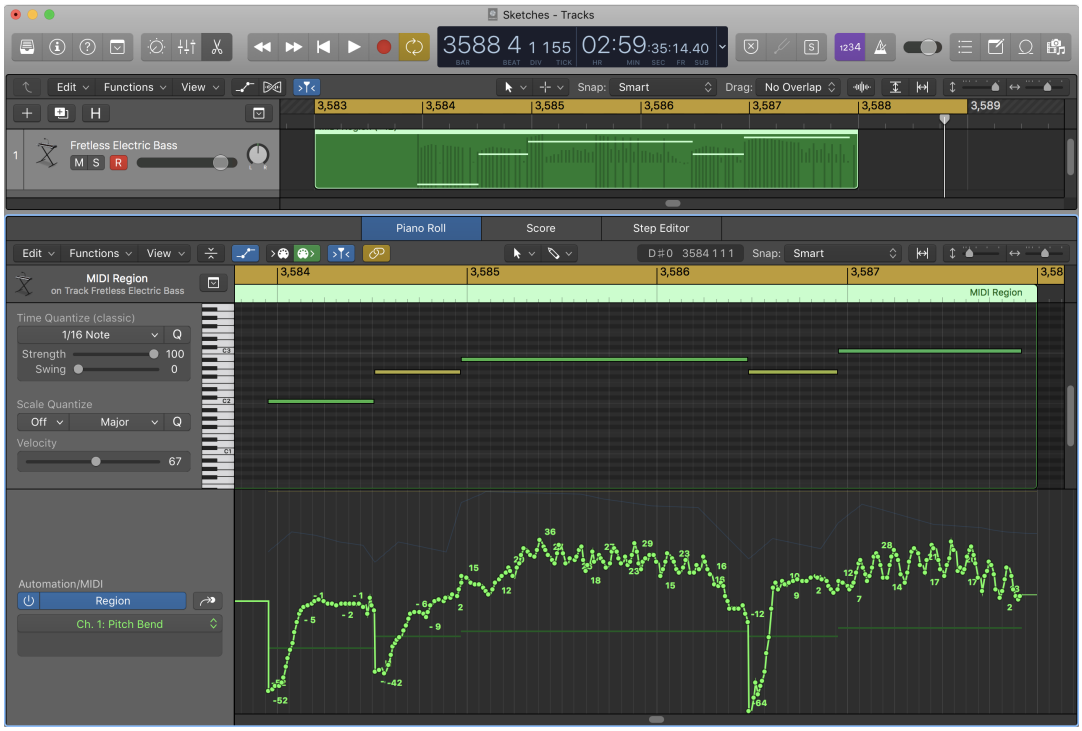

FIGURE 3.10: Pitch curve in output MIDI file

the loudness of this combination overlaid with the MIDI expression curve needed to compensate for the discrepancies in the loudness curves.

The user can select where to output the resultant MIDI file to, and if to add percussive noises (using pre-determined MIDI notes and velocities) at the points determined by the additional onset detection. The MIDI file can then be imported into a DAW, using the same settings as were used for the analysis MIDI file and the gain adjusted to the suggested setting. Finally, the synthesised audio output can be compared to the original audio input.

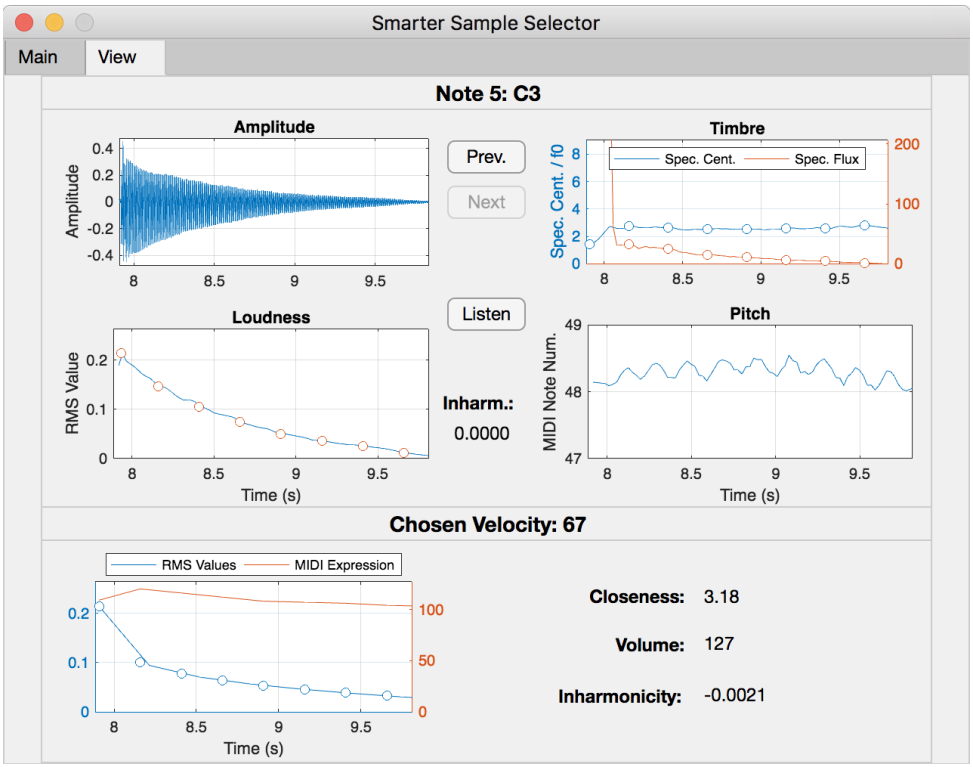

FIGURE 3.11: Smarter Sample Selector Tab 2 - View

Chapter 4

Evaluation

4.1 Introduction

The choice of methods with which to evaluate the effectiveness of the system is non-trivial. The goal is to match the original audio with the output of the VI, but the criteria by which to measure this is not necessarily to compare the audio. Ideally, the output should exactly match the input but, as the authors say in Heise, Hlatky, and Jörn Loviscach, 2009a (when referring to the error metric of their search process, but it applies to the overall assessment of the system output):

> "...obviously, a comparison of the original and the simulated waveform on a sample-by-sample basis would be by far too picky as the waveforms need only sound similar, which can be achieved even by waveforms that differ from the original."

An alternative would be to do the same analysis on the audio signals as was used in the process itself, and to compare the same derived audio feature values to calculate a closeness value. The problem with this approach is that, because the system is designed to match these qualities, the output will inherently be close to these values (within the confines of the range of outputs of the VI itself), but this is no guarantee that it is these qualities which are the important ones when a listener compares sounds, or whether the weighting placed on different measures is the most successful.

This leads to the conclusion that listening tests should be conducted to appraise the quality of the output. Another implicit aspect of the project is that it has developed as a tool to aid users by automatically setting MIDI values. Hence, an equally important measure of the effectiveness of the system is whether it can produce output that is at least of comparable quality (if not better) than an expert user can do, and in less time. So the listening test should not only compare the output of the system to the input audio, but assess the quality of user created audio, and to rank the different versions in a way that makes it clear which are more or less successful.

Given listening tests will form the basis of the evaluation, then the next thing to consider is what question to ask the subjects: should it be to determine how close any given version sounds to the original, or how life-like it sounds? The two questions are possibly interchangeable in some cases, but here there is a distinction: both user and the system are constrained by the sonic quality of the sample library being used, and the most life-like output may be to not to alter any values at all. But, in this case, as the main consideration of the project is reproducing a performance, the question should be the former, even if it transpires that this results in less life-like output. Another consideration is that a human user may be able to bestow more life-like qualities to their output using their expertise and experience, but as identifying the nature of those qualities is beyond the scope of this project, it would be of limited use to test for it.

4.2 Comparison Experiment

An experiment was conducted to determine how well the synthesised output of the system reproduces its input audio, using the parameters as discussed above.

4.2.1 Experiment set up

Four musical extracts, with durations of approximately 30 seconds each, were chosen with the following criteria: musically interesting, with an expressive performance;

good quality audio recording, with opportunity to isolate the solo instrument using panning / filtering (if not completely solo); monophonic, so no double stops. The extracts were transcribed by the author and converted into MIDI files using the music notation application Sibelius[1]. The accurate tempo for each extract was also transcribed and added to the MIDI files on a per bar basis. The audio for each extract was trimmed to include a bar of silence at the start and end, and (where appropriate) panning, editing and filtering was applied to isolated the bass guitar.

It is noted that, by using the extracted bass guitar from commercial recordings, the intended scope of the project is being extended, as the software tools were designed to be used on monophonic recordings of the isolated instrument. This was done partly because time constraints meant that commissioning a specifically recorded corpus of extracts was not possible: but also both to see how the system would perform with live performances in a realistic context, and to present them to the listening participants in that manner.

The sample library used for testing was the Kontakt[2] "Classic Bass" (fretted) bass guitar, included in the standard library of the full version of the sampler VI. A fretted instrument was chosen out of necessity, as the standard "factory" library included with Kontakt does not include a fretless instrument, and the Kontakt sampler was the only plugin guaranteed to be installed on the users computers, where different DAWs were being used. The 88x127 application was used to analyse the library's output, using the settings as in the image 3.1. These settings were saved, then loaded into the Smarter Sample Selector application, and for each of the five extracts, the trimmed (and processed) audio along with the corresponding MIDI file were loaded, analysed, and MIDI files extracted. These files were then opened in Logic Pro X[3], onto the same track as the sample library, the gain adjusted by the amount advised by the application, and the results exported as a WAV file.

[1] http://www.avid.com/sibelius

[2] https://www.native-instruments.com/en/products/komplete/samplers/kontakt-5/

[3] https://www.apple.com/uk/logic-pro/

Two audio professionals, with expertise in MIDI manipulation, were recruited to create alternative performances with which to compare the output of the system. They worked remotely, with their own equipment, but each had the Kontakt library on their computers: both used Logic Pro X. They were given a set of instructions (see Appendix A.1) about the nature of the task, and given a set of test files (audio and MIDI) with which to experiment for 30 minutes, in which they could acquaint themselves with the necessary controls of the VI and form a strategy for manipulation. They were supplied with the audio (both original and panned/filtered) and the MIDI file for each extract.

On discussion with various professionals in research for the experiment, it was suggested that spending 1.5 hours manipulating MIDI data for each instrument on a 3 minute song would be a appropriate amount of time, which would imply that 30 minutes per minute of music was a reasonable amount. So, it was decided that they would have 15 minutes to work on each of the extracts, for a total of 1 hour. After the hour was ended, they sent back the resultant MIDI files.

The MIDI files generated by the system (which took approximately 4 seconds each to produce with the SSS application) and the audio professionals were exported as wav files, and mixed with the extracted backing of the original (or, in one case, the synthesised piano accompaniment), to present them in a similar fashion to the original extracts.

The question of what to use as the "anchor" (or lower benchmark) for the listening test was not trivial. One possibility was using the output of NotePerformer, an automatic interpreter of music notation discussed in Chapter 1, but because it would not be using the same sample sounds, it would immediately sound different: also, in experiments the output was not of sufficient quality to match the sample based sounds. Another alternative was to use a live musician playing the extracts, matching the original as best as they were able. This was discounted because it may be obvious to the participants that it was a live musician (as they would not be hearing the same palette of sampled sounds), and that it might have life-like qualities

that distracted from the pure comparison of sampled versions. The medium that was decided upon was the playing of the transcribed MIDI file through the sample library in a DAW. The MIDI file was exported from Sibelius, which adds some slight "humanisation", in that it varies the velocity slightly based on in-built algorithms: otherwise the pitch, volume and expression are constant. While there is a possibility of this sounding somewhat robotic, and could be considered as a "straw man", it was deemed to be a valid comparison: this is the base by which the system could feasibly be compared, and being provably better than this would be an improvement on simple input. Also, as mentioned before, it may be possible that the best result is achieved by using "flat" settings and that the sampler performs best when not encumbered by excessive user manipulation. So using this method for an "anchor" is a valid one, and each of the four extracts was prepared and extracted in this manner.

4.2.2 Listening Test

A listening test, compiled using the Web Audio Evaluation Tool (WAET)[4] (Jillings et al., 2015) was carried out, where participants were asked to listen to the original extract, then the four synthesised versions, and to rate them in terms of which bast matched the bass guitar performance of the original. They were asked their age, and to identify whether they have experience with any of the following: playing a musical instrument; recording or mixing audio; making music using synthesisers or virtual instruments. A total of 15 participants took part, aged from 25 to 53, with a mean age of 37.9 years and standard deviation of 8.7. Of the participants, 13 identified themselves as "playing an instrument", 14 as "recording or mixing audio" and 13 as "synth user". The listening tests were performed remotely by the participants, using their own equipment, and they were asked to report the make and model of the headphones that they used.

Figure 4.1 shows the interface that was presented to the participants, where each of the four versions of the extract were to be placed on a line between "bad" and

[4] https://code.soundsoftware.ac.uk/projects/webaudioevaluationtool

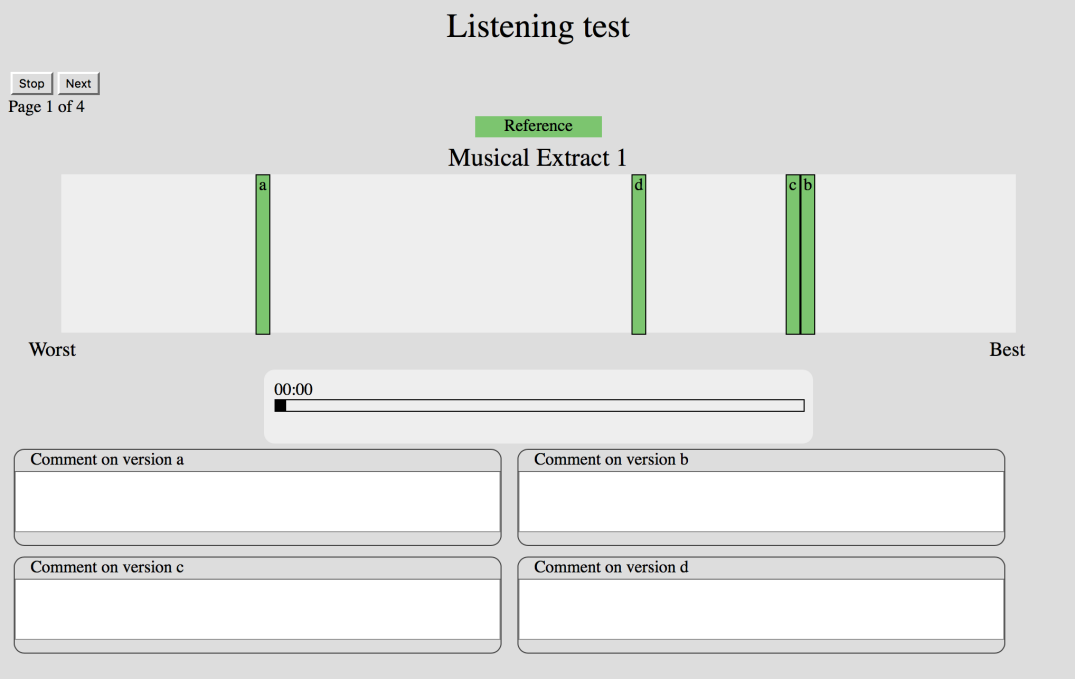

FIGURE 4.1: The WAET listening test interface

"good", represent how well the participant believed that they reproduced the original performance. The WAET tool was set to require that the participant listened to each extract and moved each slider, and that the full range of the slider was used. The latter of these settings was decided upon because, although by lifting this restriction, data could be derived about the relative success of each extract's reproduction, the test was really of the comparative success in each instance, and by demanding full scale usage allows for comparison between extracts.

4.2.3 Results

Figure 4.2 shows the results of the listening test for each extract, where the box plot for each version shows the mean authenticity rating plus 95% confidence intervals (approximately two standard deviations). The raw data from the test is shown in Appendix A.3. In the first extract, where the bass part is a simple accompaniment playing the root and fifths of the chords, the means are very similar as there was comparatively little to distinguish the versions. The 88x127/SSS version captured more

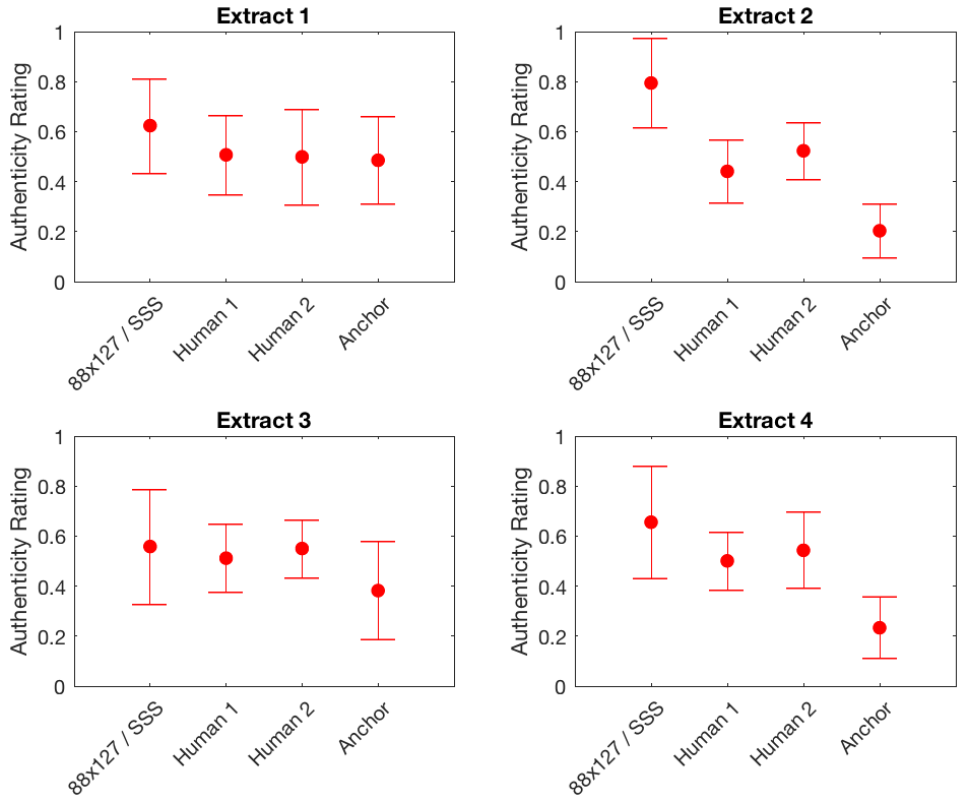

FIGURE 4.2: The results of the listening test for each extract

of the pitch variation than the human versions, and this leads to a greater variance in the results, as some listeners identified this as improving the sound and others cited a single instance where it emphasises a portamento more than on the original as being unrealistic enough to mark it lowest.

The second extract is of a melodic line played in a rubato fashion by the bass guitar, and here the 88x127/SSS version is marked higher than both of the human versions, which in turn mark much higher than the anchor. Several comments made suggest that the 88x127/SSS version often sounds slightly out of tune (as it matches more of the pitch variations of the original) which, although sometimes distracting, make it sound less mechanical than the other versions.

In the third extract, which is a jazz solo on the bass guitar, the results are similar as for the first extract, except that the 88x127/SSS version and anchor versions have greater variance due to a larger difference of opinion within the participants. Again, the more pronounced pitch variation is given as a reason both for higher and lower

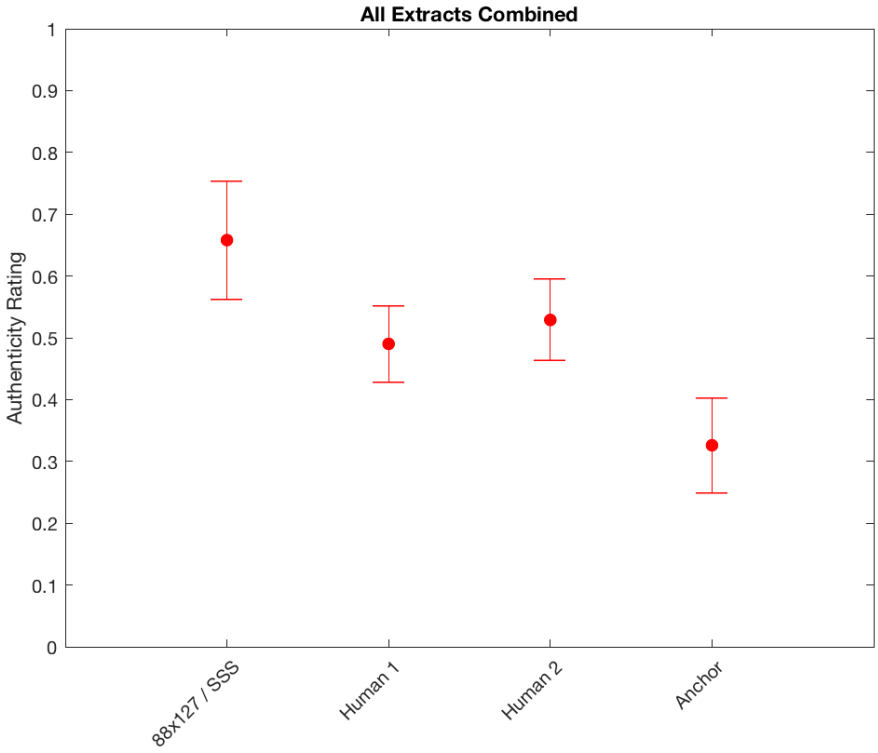

FIGURE 4.3: The combined results of the listening test

ratings, with comments both praising the vibrato and suggesting that it is "drunk sounding".

The forth extract is another jazz solo, this time with a chorus effect on the sound, which the 88x127/SSS version applications capture some elements of, with varying success. Again, this is reflected in the highest overall score with a higher variance representing the disparity of opinions. Several listeners commented that it was clearly the most accurate reproduction, although it still sounds somewhat "un-natural" and that "some artefacts are distracting".

Figure 4.3 shows the combined results of all of the extracts, with the same mean and confidence intervals. A one-way ANOVA test was performed to find out the effect of the type of MIDI manipulation on the authenticity ratings, and a significant effect was found ($F(3, 236) = 15.52$, $p<0.0000001$). A Tukey post-hoc test was also performed on the data to find the individual comparisons between each type, and the results can be seen in Table 4.1, where ** = $p<0.0001$, * = $p<0.01$, - = $p> 0.05$.

The combined results show that the 88x127/SSS system scores higher than both

	88x127	Human 1	Human 2	Anchor
88x127	.	*	*	**
Human 1	*	.	-	*
Human 2	*	-	.	*
Anchor	**	*	*	.

TABLE 4.1: Results of a Tukey post hoc test

of the expert versions which, in turn, score higher than the general MIDI anchor. The post hoc test shows a significant difference between the 88x127/SSS system and the others, and only the two expert versions are considered as indistinguishable.

4.2.4 Analysis

The results show that the goal stated at the start of the chapter, that of being able to "produce output that is at least of comparable quality (if not better) than an expert user can do, and in less time", has been met and, to some degree, exceeded. A few caveats, however, have to be added.

Firstly, when questioned about their experience, the expert users both reported that they felt that the time constraint of 15 minutes per extract was too short, and that another 10-15 minutes on each would have yielded better results. Certainly the level of detail achieved by the 88x127/SSS system was not met by the experts, as the time required to transcribe both the pitch and loudness values to that level of accuracy would far exceed the time allocated. Another version of the listening test could include an expert version where there was no time constraint applied, to give the "gold standard" version. But the time restriction is an inherent part of the problem that the software attempts to tackle - that is, to reduce the time needed by a user to craft a performance, and the fact that they needed more time to match the detail achieved automatically by the system shows that it has a potential to be useful.

Secondly, the ratings for the 88x127/SSS system were not universally positive, despite the mean ratings being the highest. Several participants placed the system's version at the bottom of the spectrum for some of the extracts, with some of the comments mentioned in section 4.2.3 about the unrealistic nature of the portamento,

and how some isolated features which were strikingly different from the original "spoiling" the overall effect for them. The complaint about the pitch variation shows that exact mapping of pitch from the original to a somewhat unrelated sample library does not necessarily produce the most life-like performance. The expert users were able to exploit this by including just the variations that they could tell added to, and did not prove a distraction to, the overall effect. This is shown in the instances where their versions were chosen above the automated system, which has no inherent way of discerning which pitch gestures fall into each category. The study of how this distinction could be implemented into the system is beyond the scope of this project, but the use of a more sophisticated VI and the addition of controls within the system to adjust the added parameters would likely alleviate much of this issue. The problem with the isolated errors (which were identified when bouncing the audio from the outputted MIDI files but left in for authenticity) are largely down to background elements in the audio causing false readings in the analysis (where onsets, loudness variation and pitch are misidentified). This could be improved by more rigorous error checking in the code, by having a user check the values before output, or simply by restricting the audio input to the monophonic solo input for which it were actually designed.

Finally, as discussed in section 4.2.2, the choice made to have the different versions for each extract rated from "worst" to "best", rather than from "bad" to "good" meant that no data was collected regarding how successful (or otherwise) the participants thought that any of the versions were, either for each extract or overall. It is possible, that for any given extract, they thought that all of the versions were almost indistinguishable to the original, or that they were completed unrelated: all that the test has shown is the relative ratings of the different versions. Another version of the test would ask for "bad" to "good" ratings, which would make overall comparisons more difficult, but would give this information. This, however, would only really show an estimation of the limitations of the sample library being used, for which no amount of MIDI manipulation could improve.

Chapter 5

Conclusions

5.1 Reflection

5.1.1 The development process

The motivation for this project evolved from the author's desire to automate a tedious process, and to provide an essential step towards a better musical computer performance. Each step towards the goal has been hard fought, but always satisfying. The fact that the project represents the culmination of two years study on the Sound and Music Computing MSc course, and that almost every module of the course (from academic studies to practical coding via DSP theory) has directly contributed to some aspect of it, is also extremely satisfying. The inclusion of a listening test into the evaluation stage has provided much valuable insight, particularly as repeated listening to the finer details of short audio extracts renders an individual's ability to discern good from bad all but useless. It has been enlightening to find that the research into related literature has shown vast amounts of excellent work in a similar field, and a relief to find none (so far) that covers exactly the same goals and approach.

5.1.2 The final system

The findings of the listening test described in Chapter 4 show that the system is effective in performing its task, and does so in a much shorter time than is possible by human manipulation. The use cases described in Chapter 1 could be achieved

with the applications in their current states. In the cases where the output MIDI files have slight errors in, or that it is deemed that the loudness or pitch variations seem too exaggerated, the author finds that it is much easier and quicker to edit those files to correct or compensate for these, than to start with a basic file and add all of the desired variation "by hand". It is possible that, similar to the conclusions reached by Dr. Simon Dixon cited in section 2.1, a user editing stage is ultimately required to yield the best results. However, before that conclusion is reached, there are several areas that have been identified where improvement can be made.

5.2 Shortcomings and additional features

5.2.1 Areas to improve

The system, as it stands, is lacking an effective system of feedback and self evaluation, with which many facets of its performance and output could be improved. The criteria for what would define a successful output, or how it would be evaluated is not clear: would it be done by a human listening, or by another computer system applying the same (or different) MIR techniques to the output? Either way, having a feedback loop to train the system towards iteratively improving the output (and being able to apply the parameters learnt to future assignments) would seem necessary for giving the best possible output, especially in the case where a perfect reproduction is not possible.

For example, the system currently chooses the most appropriate velocity by using a somewhat arbitrary set of timbral features, all of which are equally weighted. It is very likely that using these parameters, even if they are able to pick a velocity with identical values for them all, that the most appropriate candidate is still not chosen. By expanding the set of features used, with further timbre values such as MFCCs, including a measure of the noise in the signal or by including loudness and pitch information as well, and having an annotated set of output data, a neural network

could be applied to find the weighting for all of the input variables which yields the best output. Such a system is described in the paper by Heise, Hlatky, and Jorn Loviscach, 2009, where the off-line VST instruments are iteratively passed parameter settings and the output evaluated in an ongoing feedback loops that ends either when a minima is found or when the user decides that enough time has passed. It is not clear whether this approach could be applied to this system, or whether different overall approach to the process of evaluating the sample library would be required.

Improvement could also be made in the amount of data and processing resources used. Optimisation could be applied to the data structures produced to find duplicated values or to find linear associations that could be more efficiently be described by a set of parameters. For example, the volume / expression curves seen in figure 3.5 could possibly be described with sufficient accuracy by a polynomial. This could allow for greater fidelity to be recorded in other areas, where the relationships are found be be non-linear. Certainly, if the system is going to be expanded in the direction of predicting or continuing a performance, as discussed in section 2, then the data collection, evaluation and storage will need to be significantly streamlined because of the huge amount of data that would be required by such a system.

5.2.2 Expanding the scope

The system is currently optimised for the synthesis of a single instrument, and for a basic set of controls. It could be developed so that it is able to be applied it to other instruments (similar or otherwise), and to take advantage of the much wider breadth of controls available on more sophisticated software instruments.

More bass guitar control

Figure 5.1 shows the interface of the MODO Bass virtual instrument developed by IK Multimedia[1], which is a state of the art product based on physical modelling (and hence not sample based). Here, there are 5 MIDI controller settings on top of the ones

[1] http://www.ikmultimedia.com/products/modobass/

FIGURE 5.1: The interface of the MODO bass VI

currently engaged with by this project, as well as 14 key switches, which correspond to keys at the bottom of the keyboard which toggle between different settings. This, or other such advanced interfaces could be controlled and automated by the 88x127 / SSS applications, although some fundamental architectural modification would be required. Any attempts to do a similar brute force analysis of all combinations of settings would be thwarted by extremely long processing time and large amounts of derived data. A better approach would be to start at the centre of all of the settings, and work outwards incrementally, looking for any linearity at each step that could be used to interpolate areas of values.

Other ways that such a software instrument could be exploited would include adding a higher level of technical knowledge to the system's analysis phase. Things such as string muting, fret noise, inadvertently brushed strings could be identified and reproduced. The paper by Schuller, Abeßer, and Kehling, 2015 shows how some such technical playing techniques can be identified. They also describe an engine that recognises which string a note has been played on based on timbre, which could

be expanded to use an intelligent model of hand position, which would guide the most likely option based on previous notes. This could automatically trigger the appropriate variation, adding a further nuance of realism.

As discussed in the Introduction, the complexity of the MODO instrument and others makes them very difficult to master, and rely on a lot of physical knowledge of the instrument being reproduced. When the user has the capacity to specify the virtual "string" on which a note should be played, they need to have an intimate knowledge of fingering and hand position to be able to do so. In which case, any automation that this system or another could provide would be very useful to the user.

More instruments

As stated in the Introduction (Chapter 1), one of the reasons that the bass guitar was chosen was for the relative simplicity and predictability of the output. There is no reason why the system could not be applied to synthesisers (sample based or otherwise) of any instrument, but further challenges would be faced and autonomy made more difficult. In the case of instruments that are capable of continuous note variation, such as bowed strings, woodwind or brass, being able to reliably identify the note starts and ends, along with the specific articulation type would be much more difficult to achieve than with the bass guitar, with its relatively small palette of ADSR values. Perhaps, in these cases, a hybrid system where a human annotates and corrects the output of the automated system would be a good compromise between consistency and automation.

Other applications

The very accurate and reliable onset detection implemented in the system (based on the expected time from the corresponding MIDI file) means that it would be trivial to calculate the deviation from the "on the beat" time for every note. Putting together a corpus of such deviation timings would allow for analysis of a musician's temporal

approach or "groove". Several studies have been done to examine such timings, on the drums (Dittmar, Pfleiderer, and Müller, 2015, Räsänen et al., 2015) and the saxophone (Wesolowski, 2016), and a similar approach could be applied to the bass line: the ratio of durations of neighbouring notes, or "swing" (and whether consistent or not); whether ahead or behind the beat; how it matches or plays against the drums.

5.3 Further Work

The specific problem tackled in this project (reproducing a performance with a synthesiser) was acknowledged in the Introduction (Chapter 1) to be rather contrived, as it is not a task that would regularly be performed by somebody working in the field. But it is hoped that the methods and technologies developed here will be a step towards the ultimate goal of improved automated synthesiser performance, given just the input that a human musician would have: a piece of sheet music. This is already a task performed to some degree by DAWs and notation software, but there is vast room for improvement as the output is always distinguishable from live performance, unless it has been significantly honed by a human editor.

By applying the analysis techniques shown here on a large corpus of annotated recordings (audio synced to corresponding MIDI / MusicXML files), it is possible that the unique "fingerprint" of a musician could be captured. Possibly by modelling the temporal dimension as a hidden Markov model, a system could learn the most likely trajectory in all parameters at any given instance. When given the symbolic input of a musical extract is given the system would work through and apply appropriate controller messages, as shown in this project, could be sent to the synthesiser: hence "playing" the piece in the same way that the musician "would have" played it. A recent paper by a team from Google (Oord et al., 2016) showed a similar concept where piano performances were created using very small audio grains. Another paper, Shih et al., 2017, shows some success in extracting the unique playing style of different violin soloists, and uses that data to create new performances by editing

existing ones. But a system which could use existing synthesiser technology would be much less computationally demanding and more accessible to working musicians. Theoretically, and with enough data and computational power, the "fingerprints" of every musician in an orchestra could be taken, and the correctly mixed recordings generated could accurately predict and be indistinguishable from a recording made by the orchestra themselves.

Appendix A

Listening Test Resources

A.1 List of musical extracts

1. "5 O'Clock Lullaby" - Barbra Lica (Marc Rogers playing bass guitar) (Lica, 2016)[1]. See A.1.

2. "About Now" - Mark Egan (Mark Egan playing bass guitar) (Egan, 2014)[2]. See A.2.

3. "Bright Size Life" - Pat Metheny (Jaco Pastorius playing bass guitar) (Metheny, 1976)[3]. See A.3.

4. "James" - Pat Metheny (Christian McBride playing bass guitar) (Metheny, 2004)[4]. See A.4.

[1] https://www.youtube.com/watch?v=FZSxyqsfonQ
[2] https://www.youtube.com/watch?v=5DWHGtdZ5rI
[3] https://www.youtube.com/watch?v=RQKcRBJcQYk
[4] https://www.youtube.com/watch?v=5_C7QQFBiH0

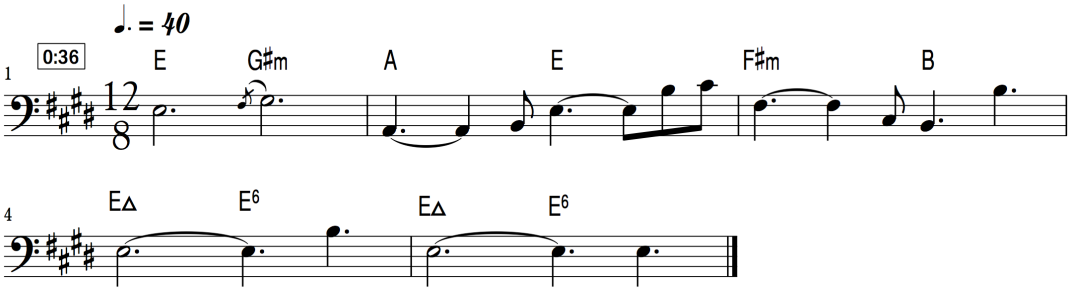

FIGURE A.1: Musical Extract 1 - "5 O'Clock Lullaby"

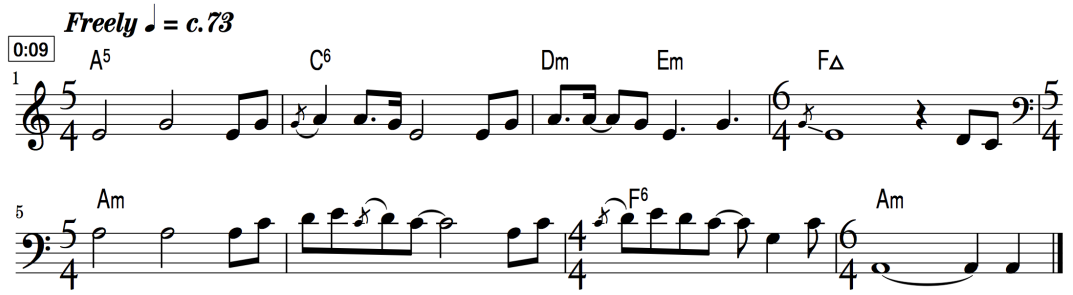

FIGURE A.2: Musical Extract 2 - "About Now"

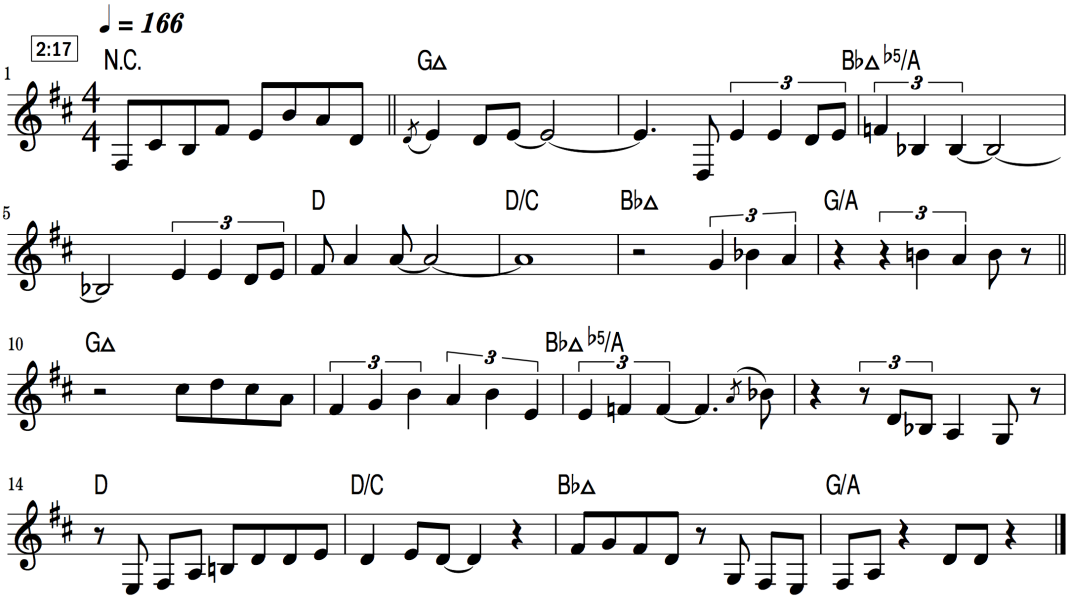

FIGURE A.3: Musical Extract 3 - "Bright Size Life"

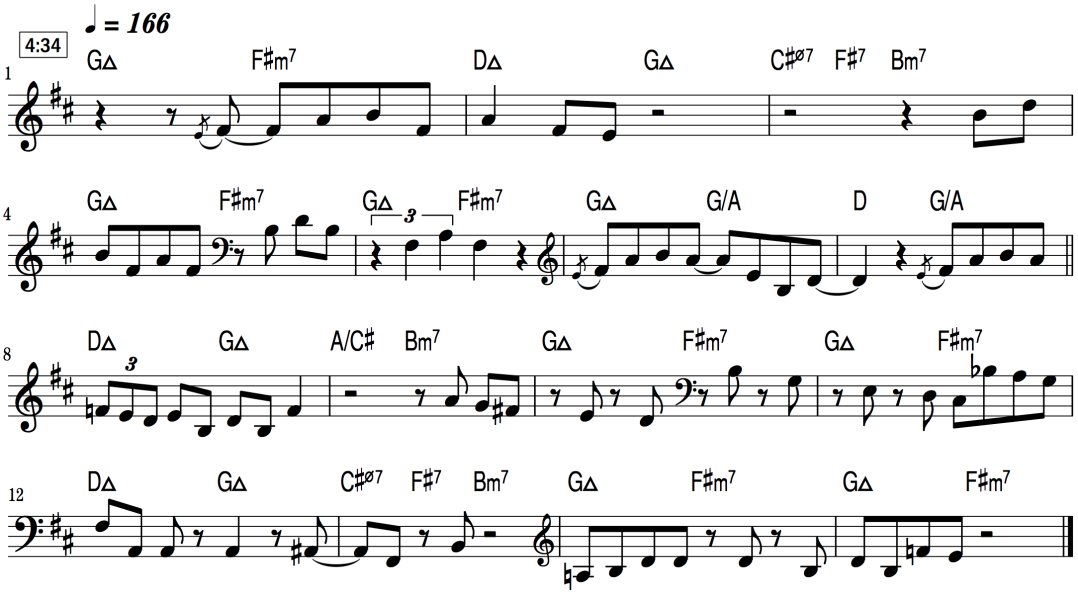

FIGURE A.4: Musical Extract 4 - "James"

A.2 Instructions given to expert users

Thank you for agreeing to help me evaluate my MSc project. I have designed and built an application that automatically reproduces musical performances by generating MIDI files to be played through a sampler virtual instrument (VI). It does this by first exhaustively analysing the output of the VI, before analysing the original audio, and picking the best values to match the audio.

Your challenge is to provide an expert human alternative, given the same inputs. This will then be compared with the output of my system in a listening test, to evaluate which sounds closest to the original. The control will be an unedited MIDI file performance. The participants in the listening test will be asked to rate each version for closeness to the original file, not to rate how human or life-like it sounds.

You have been given files which contain 4 musical extracts (chosen to cover expressive performances in various styles, tempos and complexities). For each extract,

you have been supplied with 4 files: the sheet music of the transcription; the original audio; a filtered version of the audio (where the bass is isolated or enhanced); and a MIDI file of the transcription. The latter two elements will be used as inputs to my system. A Logic X file is supplied with all of the files lined up correctly: if you are using another DAW, then use the files with the suffix "- COMPLETE" (2 audio files and 1 MIDI file) to create the project with.

The VI we will be using is the "Classic Bass" from the Kontakt 5 Factory Library (in the "Band" folder), with the controls and FX settings at the default values. My system will use exactly this library and settings as well.

Your task is to manipulate the lower MIDI line (using the upper as a reference) to make the output of the VI match the original audio file as closely as possible. You can manipulate the following to try and match the audio:

1) Note on / off position (lining up with the audio on screen may aid this)

2) Velocity

3) Pitch Bend

4) MIDI volume

5) MIDI expression (volume and expression combine to alter the output volume)

6) Add / delete notes (for ghost notes or to aid glisses)

In order to make the test uniform, you are asked to spend a set amount of time manipulating each extract. You have also been supplied a set of "Test" files (sheet music and audio / MIDI files at the end of the main project) that you can use to experiment with, and prepare your strategy: please spend no more than 30 minutes with this file. You should then spend up to 15 minutes on each of the 4 extracts, and please do not exceed that on any one (even if less time is spent on another).

When you have finished, please send the resultant MIDI files to me, and I will bounce the audio into the correct format for the listening test.

#	Age	Musician	Sound Engineer	Synth User	Headphones	Extract 1 88x127	Extract 1 Human 1	Extract 1 Human 2	Extract 1 Anchor
1	32	☑	☑	☑	AKG K550	0.91	0.37	0.42	0.07
2	25	☑	☑	☑	Adam A7 (Monitors)	0.00	1.00	0.64	0.33
3	50	☑	☑	☑	Bose QC35 II	0.83	0.08	0.33	0.61
4	41	☑	☑	☑	AKG K240	0.59	0.16	0.37	0.89
5	35	☑	☐	☐	None	0.87	0.61	0.17	0.40
6	53	☑	☑	☑	Oppo PM-1	0.70	0.86	0.78	0.12
7	40	☑	☑	☑	Oppo PM-1	0.68	0.19	0.40	0.93
8	41	☑	☑	☑	AKG K240	0.84	0.80	0.18	0.08
9	38	☐	☑	☑	AKG K240	0.96	0.51	0.15	0.41
10	33	☐	☑	☑	AKG K550	0.92	0.38	0.60	0.08
11	45	☑	☑	☑	AKG K450	0.83	0.17	0.38	0.52
12	48	☑	☑	☑	Sennheiser HD215	0.16	0.66	0.80	0.72
13	28	☑	☑	☐	Sennheiser HD215	0.85	0.75	0.45	0.16
14	35	☑	☑	☑	Grado SR 80	0.08	0.81	0.85	0.77
15	25	☑	☑	☑	AKG K450	0.42	0.00	1.00	0.78
MEAN	**37.9**	**13**	**14**	**13**		**0.64**	**0.49**	**0.50**	**0.46**

#	Extract 2 88x127	Extract 2 Human 1	Extract 2 Human 2	Extract 2 Anchor	Extract 3 88x127	Extract 3 Human 1	Extract 3 Human 2	Extract 3 Anchor	Extract 4 88x127	Extract 4 Human 1	Extract 4 Human 2	Extract 4 Anchor
1	0.80	0.10	0.61	0.41	0.83	0.41	0.56	0.17	0.82	0.16	0.32	0.42
2	1.00	0.60	0.35	0.01	0.00	0.65	0.94	0.34	1.00	0.49	0.69	0.00
3	0.06	0.31	0.84	0.63	0.06	0.59	0.33	0.93	0.08	0.89	0.53	0.29
4	0.90	0.36	0.60	0.16	0.60	0.41	0.89	0.16	0.08	0.88	0.57	0.34
5	0.87	0.63	0.39	0.15	0.88	0.35	0.62	0.10	0.88	0.12	0.39	0.63
6	0.95	0.71	0.42	0.14	0.77	0.61	0.14	0.34	0.89	0.54	0.43	0.10
7	0.92	0.51	0.46	0.15	0.80	0.86	0.24	0.75	0.82	0.69	0.53	0.15
8	0.70	0.89	0.36	0.08	0.85	0.48	0.55	0.16	0.85	0.52	0.23	0.17
9	0.80	0.60	0.41	0.14	0.96	0.61	0.43	0.08	0.92	0.34	0.42	0.11
10	0.98	0.63	0.40	0.09	0.93	0.31	0.45	0.09	0.86	0.54	0.43	0.08
11	0.98	0.38	0.55	0.12	0.90	0.37	0.65	0.05	0.97	0.43	0.32	0.10
12	0.14	0.35	0.79	0.60	0.17	0.43	0.64	0.83	0.10	0.81	0.30	0.70
13	0.97	0.57	0.44	0.19	0.84	0.36	0.52	0.09	0.93	0.32	0.45	0.14
14	0.87	0.13	0.16	0.10	0.04	0.98	0.52	0.49	0.79	0.13	0.74	0.08
15	1.00	0.66	0.00	0.29	0.01	0.69	0.24	0.94	0.00	0.89	1.00	0.38
MEAN	**0.79**	**0.50**	**0.45**	**0.22**	**0.58**	**0.54**	**0.52**	**0.37**	**0.67**	**0.52**	**0.49**	**0.25**

FIGURE A.5: Listening Tests - Raw Data

A.3 Exact results of listening tests

The data collected from the listening tests is shown in its entirety in A.5.

Bibliography

Abeßer, Jakob and Gerald Schuller (Sept. 2017). 'Instrument-Centered Music Transcription of Solo Bass Guitar Recordings'. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.9, pp. 1741–1750.

Cheveigne, Alain de and Hideki Kawahara (2002). 'YIN, a fundamental frequency estimator for speech and music'. In: *The Journal of the Acoustical Society of America* 111(4), pp. 1917–1930.

Coleman, Graham, Esteban Maestre, and Jordi Bonada (2010). 'Augmenting Sound Mosaicing with Descriptor-Driven Transformation'. In: *Proceedings of the Digital Audio Effects (DAFx-10)*, pp. 1–4.

Dannenberg, Roger B (2006). 'Concatenative Synthesis Using Score-Aligned Transcriptions Music Analysis and Segmentation'. In: *International Computer Music Conference*, pp. 352–355.

Dittmar, Christian, Martin Pfleiderer, and Meinard Müller (2015). 'Automated Estimation of Ride Cymbal Swing Ratios in Jazz Recordings'. In: *The Journal of the Acoustical Society of America*.

Dixon, Simon (2017). 'Queen Mary Sound and Music Computing: ECS731P Music Analysis and Synthesis Lecture Notes'.

Dixon, Simon, Matthias Mauch, and Dan Tidhar (2012). 'Estimation of harpsichord inharmonicity and temperament from musical recordings'. In: *The Journal of the Acoustical Society of America* 131.1, pp. 878–887.

Egan, Mark (2014). *About Now*. Wavetone Records.

Fletcher, Harvey (1964). 'Normal Vibration Frequencies of a Stiff Piano String'. In: *The Journal of the Acoustical Society of America* 36.1.

Heise, Sebastian, Michael Hlatky, and Jorn Loviscach (2009). 'Editing MIDI Data Based on the Acoustic Result'. In: *AES 127th Convention*. Convention Paper 7863. Audio Engineering Society.

Heise, Sebastian, Michael Hlatky, and Jörn Loviscach (2009a). 'Automatic cloning of recorded sounds by software synthesizers'. In: *Audio Engineering Society Convention 127*. Audio Engineering Society.

— (2009b). 'Editing MIDI Data Based on the Acoustic Result'. In: *127th Audio Engineering Society Convention*.

— (Jan. 2010). 'Audio re-synthesis based on waveform lookup tables'. In: *129th Audio Engineering Society Convention*. Vol. 2, pp. 1186–1192.

Jillings, Nicholas et al. (July 2015). 'Web Audio Evaluation Tool: A browser-based listening test environment'. In: *12th Sound and Music Computing Conference*.

Kramer, Patrick et al. (2012). 'A digital waveguide model of the electric bass guitar including different playing techniques'. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings* Vi, pp. 353–356. ISSN: 15206149.

Krumhansl, Carol (1989). *Structure and perception of electroacoustic sound and music*. Ed. by S. Nielzén and O. Olsson. Excerpta Medica. Chap. Why is musical timbre so hard to understand?, pp. 43–53.

Lee, Heejung (2006). 'Violin portamento: An analysis of its use by master violinists in selected nineteenth-century con- certi'. In: *9th International Conference on Music Perception and Cognition*. ICMPC.

Lerch, Alexander (2008). 'Software-Based Extraction of Objective Parameters from Music Performances'. PhD thesis. ISBN: 978-3640294961.

Lica, Barbra (2016). *I'm Still Learning*. ADA / Justin Time Records.

Lindemann, Eric (2007). 'Music synthesis with reconstructive phrase modeling'. In: *IEEE Signal Processing Magazine* 24.2, pp. 80–91. ISSN: 10535888.

Mauch, Matthias and Simon Dixon (2014). 'pYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions'. In: *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*.

McAdams, Stephen et al. (1995). 'Perceptual scaling of synthesized musical timbres: Common dimensions, specificities, and latent subject classes'. In: *Psychological research* 58.3, pp. 177–192.

Metheny, Pat (1976). *Bright Size Life*. ECM.

— (2004). *Estival Jazz Lugano*. Jazz on the Screen.

Mintz, Daniel (2007). 'Toward Timbral Synthesis: a new method for synthesizing sound based on timbre description schemes'. MA thesis. University of California (Santa Barbara).

MIREX 2017 Evaluation Results (2017). URL: http://www.music-ir.org/mirex/results/2017/mirex_2017_poster.pdf.

Mitchell, Thomas (2012). 'Automated evolutionary synthesis matching: Advanced evolutionary algorithms for difficult sound matching problems'. In: *Soft Computing* 16.12, pp. 2057–2070. ISSN: 14327643.

Moorer, James A. (1977). 'On the Transcription of Musical Sound by Computer'. In: *Computer Music Journal* 1.4, pp. 32–38. URL: http://www.jstor.org/stable/40731298.

Oord, Aäron van den et al. (2016). 'WaveNet: A Generative Model for Raw Audio'. In: *CoRR* abs/1609.03499. arXiv: 1609.03499. URL: http://arxiv.org/abs/1609.03499.

Raffel, Colin (2016). 'Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching'. PhD thesis. Columbia University.

Ramirez, Rafael, Esteban Maestre, and Xavier Serra (2010). 'Automatic performer identification in commercial monophonic Jazz performances'. In: *Pattern Recognition Letters* 31.12, pp. 1514–1523. ISSN: 01678655. URL: http://dx.doi.org/10.1016/j.patrec.2009.12.032.

Räsänen, Esa et al. (2015). 'Fluctuations of hi-hat timing and dynamics in a virtuoso drum track of a popular music recording'. In: *PLoS ONE* 10.6, pp. 1–16. ISSN: 19326203.

Riionheimo, Janne and Vesa Välimäki (July 2003). 'Parameter Estimation of a Plucked String Synthesis Model Using a Genetic Algorithm with Perceptual Fitness Calculation'. In: *EURASIP Journal on Advances in Signal Processing* 8, p. 758284. ISSN: 1687-6180. URL: https://doi.org/10.1155/S1110865703302100.

Schuller, Gerald, Jakob Abeßer, and Christian Kehling (2015). 'Parameter extraction for bass guitar sound models including playing styles'. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 404–408. ISBN: 9781467369978.

Schwarz, Diemo (2004). 'Data-driven concatenative sound synthesis'. PhD thesis. Ircam Centre Pompidou.

Seashore, Carl E (1938). *Psychology of Music*. McGraw-Hill.

Serra, Xavier and Julius Smith (1990). 'Spectral Modeling Synthesis: A Sound Analysis/Synthesis System Based on a Deterministic Plus Stochastic Decomposition'. In: *Computer Music Journal* 14.4, pp. 12–24. ISSN: 01489267, 15315169. URL: http://www.jstor.org/stable/3680788.

Shih, Chi Ching et al. (2017). 'Analysis and synthesis of the violin playing style of Heifetz and Oistrakh'. In: *20th International Conference on Digital Audio Effects*. DAFx, pp. 466–473.

Sturm, Bob L, Marcela Morvidone, and Laurent Daudet (2010). 'Musical instrument identification using multiscale mel-frequency cepstral coefficients'. In: *European Signal Processing Conference* 1, pp. 477–481. ISSN: 22195491.

Tremblay, Pierre Alexandre and Diemo Schwarz (2010). 'Surfing the Waves : Live Audio Mosaicing of an Electric Bass Performance as a Corpus Browsing Interface'. In: *New Interfaces for Musical Expression* June, pp. 15–18. URL: http://eprints.hud.ac.uk/7421/.

Walker, Timothy M. and Sean P. Whalen (2013). 'Dynamic recombination of evolving guitar sounds (DREGS): A genetic algorithm approach to guitar synthesizer control'. In: *Proceedings - 2013 IEEE International Symposium on Multimedia, ISM 2013*, pp. 248–254.

Wesolowski, Brian C (2016). 'Timing deviations in jazz performance: The relationships of selected musical variables on horizontal and vertical timing relations: A case study'. In: *Psychology of Music* 44.1, pp. 75–94. ISSN: 0305-7356. URL: http://pom.sagepub.com/cgi/doi/10.1177/0305735614555790.

Yang, Luwei, Khalid Rajab, and Elaine Chew (2016). 'AVA: A Graphical User Interface for Automatic Vibrato and Portamento Detection and Analysis'. In: *Proceedings of the 42nd International Computer Music Conference*.