Audio Engineering Society

Convention Paper 10185

Presented at the 146th Convention
2019 March 20 – 23, Dublin, Ireland

Reproducing bass guitar performances using descriptor driven synthesis

Dave Foster[1] and Joshua D Reiss[2]

[1]*Queen Mary University of London, London, UK*
[2]*Centre for Digital Music, Queen Mary University of London, London, UK*

Correspondence should be addressed to Dave Foster (`dave@swingcitymusic.co.uk`)

ABSTRACT

Sample-based synthesis is a widely used method of synthesising the sounds of live instrumental performances, but the control of such sampler instruments is made difficult by the number of parameters that control the output, the expertise required to set those parameters and by the constraints of the real-time system. In this paper, the principles of descriptor-driven synthesis were used to develop a pair of software tools that 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 employed build on existing work and commercially available products. The output of the system is compared to manipulation by expert users. 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.

1 INTRODUCTION

Orchestral sample-based Virtual Instruments (VIs) have been increasing in quality and range as the processing power and storage space that they require has become more affordable and available, to the point that they are acceptably used in media where the budget is not available for a live orchestra. However, the increase in quality is matched by an increase in complexity and required expertise, sometimes to the point where the demands on the user begin to approach those of learning to play the very instrument being synthesised.

Although there has been much work on the control of synthesisers which themselves generate the sounds, less has been done at the higher level of abstraction: that is, controlling the input and settings of an existing sampler VI. This is the level at which users interface with orchestral samplers and where the automation could be both useful and potentially aid the production of improved output.

The system described in this paper tackles a subset of the larger problem of automated synthesiser control, namely of recreating musical performances using sample-based VIs by the analysis of the input audio and generation of appropriate MIDI (Musical Instrument Digital Interface) messages. The use case is to take an input audio file of an instrumental performance, and to output a MIDI file which, when played through a specific VI, reproduces the nuances of that performance.

The problem has relevance in style transformation and performance evaluation or editing. The aim is produce output that is at least of comparable quality (if not better) than an expert user can produce, and in less time.

The results are evaluated by a listening test, where the participants were asked to listen to different synthesised reproductions of a musical extract, and to rate them according to how well they match the original recording.

The system was constrained by restricting it to a VI reproducing a single instrument, the fretless bass guitar (chosen for the predictable nature of plucked strings, and for its capacity for expressive performances using micro-tonal pitch variation); and to a subset of the range of MIDI controller message types, namely MIDI pitch, velocity, MIDI volume and expression.

2 RELATED WORK

The extraction of bass guitar parts from recordings, and their subsequent transcription are presented in the papers by Hainsworth and Macleod [1] and Ryynänen and Klapuri [2]. The more recent work by Abeßer and Schuller [3] expands on their techniques, and their system achieves an accuracy of 91% (in the note-based evaluation results for score-level evaluation), outperforming all other methods. If this is the state of the art, then up to 9% of notes will be incorrect, and hence there will be a need either for human generated input or human checking of output. We chose the former, so our system is provided with approximate note data to avoid errors. This means that the system is receiving the same input as a sight-reading musician would be, that is, a representation of the piece being played.

In Heise et al [4], a speech signal is reduced to its set of features, and then reconstructed 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, searching a multi-dimensional database, is similar to what is being done herein. The papers by Yang et al [5] and Lee [6] suggest techniques for the analysis of pitch, and the higher level features of portamento (pitch sliding up to, down to or between notes) and vibrato. Applying the latter analysis in our system did not reduce complexity or data size, but could be useful in future versions.

In Mitchell [7], the field of "evolutionary sound matching" is described, for application to a native FM synthesiser. Riionheimo and Välimäki [8] and Walker and Whalen [9] both use a Genetic Algorithm to match target sounds. Heise et al [10] match a generic sound to the settings of a generic synthesiser, allowing for up to 50 individual settings requiring a highly optimised search strategy for finding the best combination of settings. All of these papers concern the recreation of sounds using native, sound-generating synthesisers, not with the automation of controls at a higher level of abstraction as may be needed to control sampler VIs. Some of the approaches have analogies, particularly with the analysis of input sounds.

In another paper by Heise et al [11], 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. Of particular interest was their matching velocity values to output level. This allowed 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. Schuller et al [12] analyse the sound of a bass guitar and derive features from it such as plucking style, expression style and which string a note has been played on. These parameters were not considered in this paper, but could improve the performance if they were matched with corresponding settings in an advanced VI.

"Descriptor-driven transformation" is defined by Coleman et al [13] as being where "input samples are transformed with respect to target descriptors". They refer to Hoffman et al [14] for "descriptor-driven synthesis". 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 paper 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.

3 IMPLEMENTATION

The system we designed for the task consists of two parts: first, the exhaustive analysis of the output of a sample based VI; and second, the analysis of an input audio file and generation of a corresponding output MIDI file. Two software applications were developed to implement and automate these processes: "88x127"

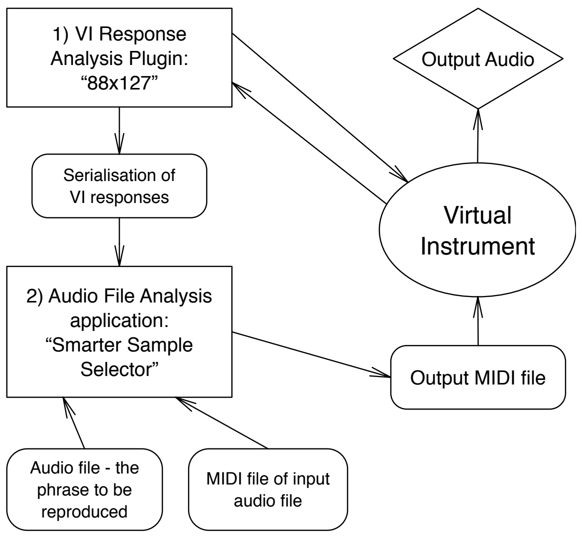
1) VI Response Analysis Plugin: "88x127"
Serialisation of VI responses
2) Audio File Analysis application: "Smarter Sample Selector"
Audio file - the phrase to be reproduced
MIDI file of input audio file
Output Audio
Virtual Instrument
Output MIDI file

Fig. 1: A flow chart of the segmented implementation

for generating the MIDI file used in testing the VI, and for analysing the resultant audio file; and "Smarter Sample Selector" for analysing and recreating an instrumental performance. 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. 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[1] EXS sampler plugin. Figure 1 shows a flow chart of how the system was segmented and implemented.

3.1 88x127

The system begins by exhaustively evaluating the output of a sample based VI, by triggering and recording a note corresponding to each combination of input parameters. Each of the notes is then analysed for features pertaining to the three categories of loudness, pitch and timbre.

3.1.1 Loudness

Loudness is evaluated discretely throughout the duration of the note using the RMS (Root Mean Square) formula. The values are stored as percentages of the maximum value.

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

3.1.2 Pitch

Pitch is analysed using the YIN estimator [15], and the knowledge of the expected pitch of the note was used in a novel variation to the algorithm which reduces computation time (often by 50%), 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 the window size. So formula (6) from [15] (where x is a window of a discrete signal, and ρ is the frequency period) is replaced by:

$$d_t(\rho) = \sum_{j=1}^{w'} (x_j - x_{j+\rho})^2, \tag{1}$$

where $w' = \text{floor}(F_s/F_0')$, F_s is the sample rate and F_0' is the upper frequency limit. Features are then extracted from the pitch curve:

- The mean value.
- The initial pitch.
- The time of first expected value (representing the portamento up / down to the note).
- The maximum absolute deviation from the mean.
- The first (if any) time of an absolute deviation from the mean that exceeds a set threshold value.

3.1.3 Timbre

Timbre is evaluated by two measures described in Klapuri [16]. The first is brightness, represented by spectral centroid:

$$C_t = \sum_{n=1}^{N} (M_t[n] \star n) \Big/ \sum_{n=1}^{N} M_t[n], \tag{2}$$

where M_t is a frame of an audio signal in the frequency domain, and n is the frequency bin number. The second is spectral dynamics, represented by spectral flux, the sum of the squared differences in the spectra of consecutive frames M_{t-1} and M_t:

$$F_t = \sum_{n=1}^{N} (M_t[n] - M_{t-1}[n])^2. \tag{3}$$

This process was automated by the development of the "88x127" application, which is presented as a single

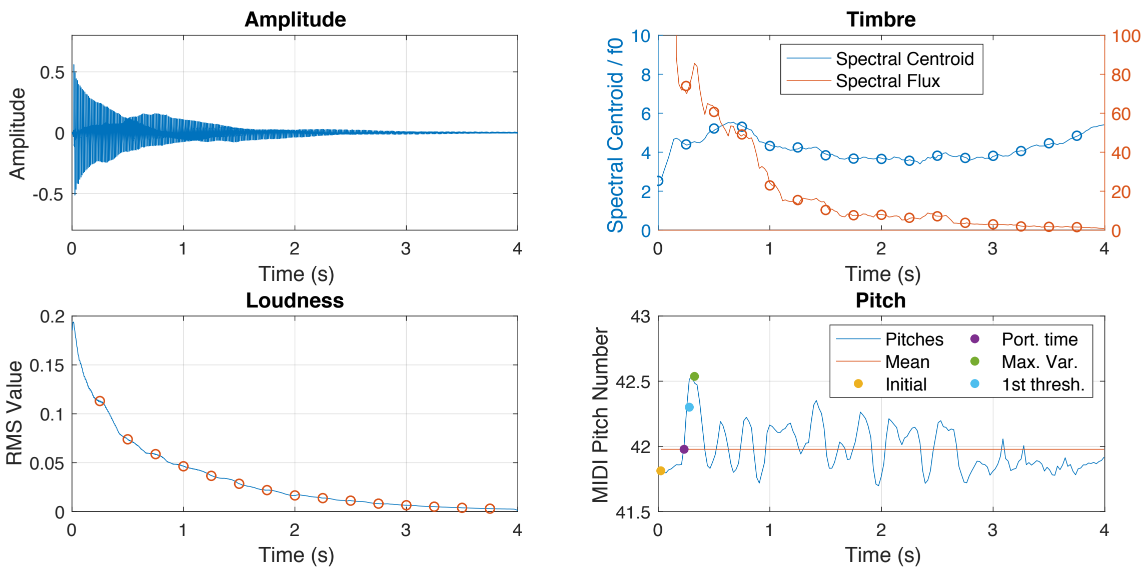

Fig. 2: 88x127 - Analysis results I

window with four tabs, worked through sequentially by the user to set the parameters, view the analysis results and output a data file.

The application takes as input the ranges and increments of the four variables listed in the introduction, and generates a MIDI file consisting of notes (separated by 1 second gaps) of every chosen pitch at every chosen velocity (at a 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. This MIDI file is then imported into a digital audio workstation (DAW) and put onto a channel strip with the VI under testing.

The audio file generated by the DAW is then imported back into 88x127 and the analysis begins by separating the audio file into separate notes. The values derived by the analysis of each note are saved in a data file, and the results displayed to the user. Figure 2 shows the analysis results of a single MIDI pitch / velocity combination, showing the derived amplitude, loudness, timbre and pitch values throughout the duration of the note. Figure 3 shows a graph of the maximum loudness given different MIDI volume / expression permutations. The visualisations of the data proved to be useful in choosing the most efficient range of values to use for the input.

3.2 Smarter Sample Selector

The second part of the system is to analyse a given audio performance, and to craft a MIDI file by finding the best matching parameter settings for each note by searching the corpus of data collected previously for the VI. The “Smarter Sample Selector” (SSS) tool was developed to automate this process, and requires three input files: the MIDI file containing the notes of the performance and their approximate timings; the audio file of the performance; and the data file saved in 88x127.

3.2.1 Note analysis

Similar to the 88x127 application, SSS 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. In this case, however, the exact start and end times of the notes is not known. So, to accurately reproduce 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 the onset detection easier than with an instrument with a softer onset, and the MIDI information supplied is used

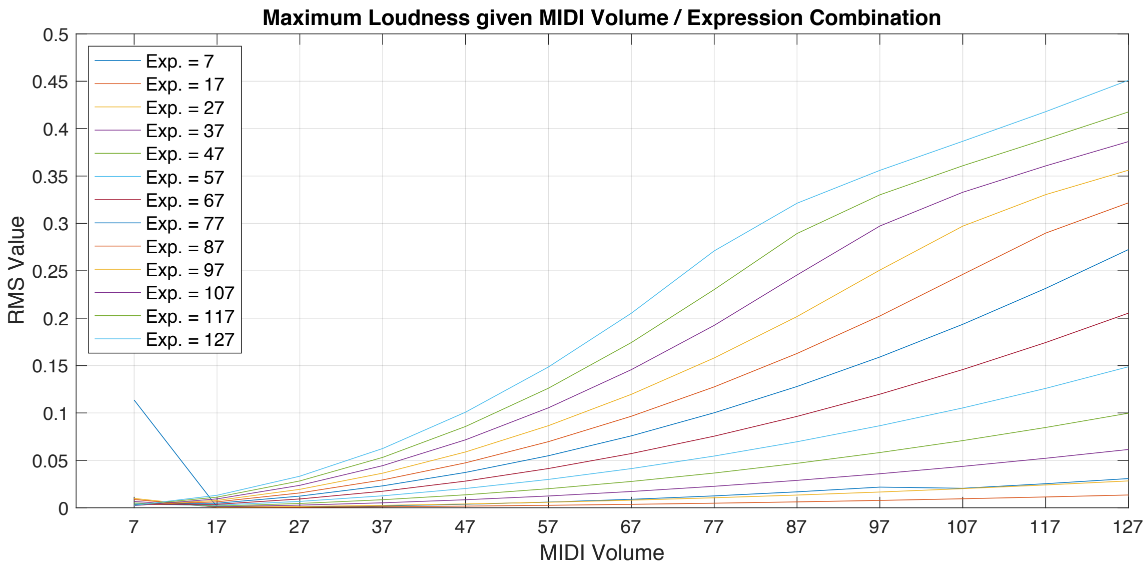

Fig. 3: 88x127 - Analysis results II

to provide a framework of likely start times. Spectral flux (equation 3) was found to be the most reliable onset detection technique for finding clear peaks. To improve accuracy further, values were windowed by a triangular window with the centre at the MIDI note position, thus weighting peaks closer to the likely start time higher.

These onset values are then used to segment the audio into separate notes, each ending either just prior to the subsequent note starting, or when the loudness of the note falls below a threshold. The loudness, pitch and timbre of the notes are then evaluated as before.

3.2.2 MIDI file construction

With the analysis complete, the next task is that of constructing the MIDI file. For each note, the following steps are taken:

1. The note on and off messages are set to the start and end times derived by onset detection.
2. The velocity value for the note is chosen by finding the option that best matches the timbre of the original.
3. The MIDI volume is set to match the initial loudness of the original.
4. A curve of expression values is constructed to match the loudness throughout the note.
5. A curve of pitch bend values is constructed to match the pitch variation of the original, by first compensating for variations in the chosen sample.

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 used in the loudness analysis. 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. This also gives a gain adjustment that will be required to match the loudness of the overall extract.

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. This gives the velocity value for the note: to match the initial loudness, the array of MIDI volume / expression combinations for this velocity value is searched for the closest value, and these values set at the start of the note.

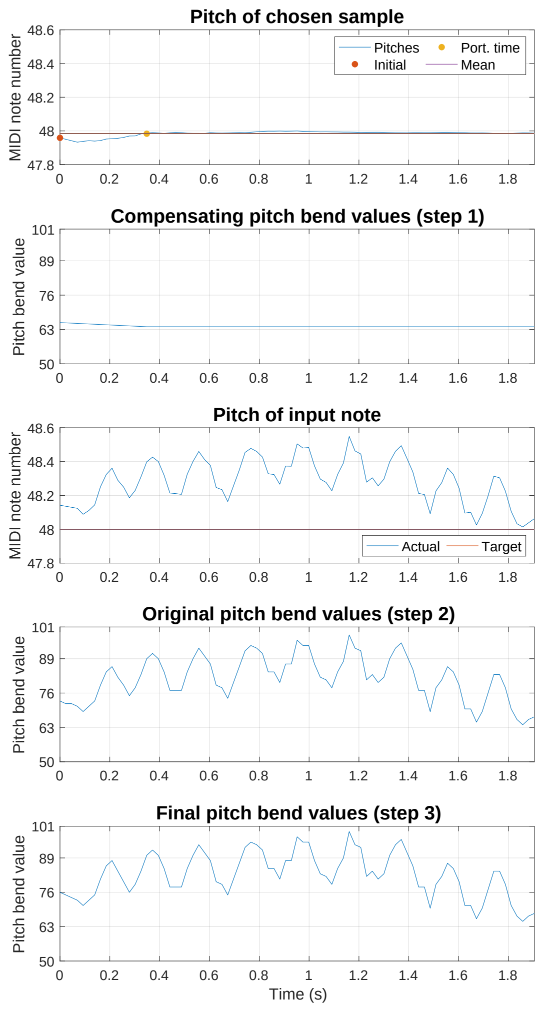

Fig. 4: Constructing the pitch bend curve

The loudness of the note throughout its duration is then crafted to match that of the original. This is achieved by comparing the loudness of the target note and the sampled note at each landmark point, and using the expression controller to compensate dynamically. When this has been performed for each note in the extract, the MIDI file now has MIDI volume values at the start of each note, and a discrete curve of expression values running throughout.

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 pitch variation of the output matches the input. This is done in three steps:

1. The main pitch variations of the chosen sample are compensated for by creating a curve array that cancels them out.
    - The mean pitch value of the sample is corrected by a constant pitch shift in the opposite direction.
    - 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.
2. The pitch variations in the input audio are converted into a pitch bend array.
3. The two arrays are added together, rounded to integer values, and applied at the correct time for the note.

Figure 4 shows how the curve is constructed, where the MIDI pitch bend values are in the range [0 127], which map to [-1 1] semitones, and 63 represents no pitch bend.

The constructed MIDI file can then imported into a DAW, using the same settings as were used for the analysis MIDI file, and the synthesised audio output can be compared to the original audio input.

4 EVALUATION

A listening test experiment was conducted to determine how well the synthesised output of the system reproduces its input audio.

4.1 Experimental set-up

Four musical extracts of fretless bass performances, 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 (simultaneously played notes). The extracts were transcribed and converted into MIDI files.

The sample library used for testing was the Kontakt[2] "Classic Bass" (fretted) bass guitar, included in the

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

standard library of the full version of the sampler VI. This sample library was chosen as it is the industry standard across the main platforms: the default library does not include a fretless bass guitar, so the fretted model was used with the knowledge that MIDI pitch bend could be used to mimic the characteristic pitch variation of the fretless instrument.

The 88x127 application was used to analyse the library's output. The Smarter Sample Selector application was run on each of the four extracts, where the trimmed (and processed) audio along with the corresponding MIDI file were loaded, analysed, and MIDI files extracted. These MIDI files were then opened in Logic Pro X, 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.

Two audio professionals, with expertise in MIDI manipulation, were recruited to create alternative performances with which to compare the output of the system. They were not using the system, but instead manually manipulating the MIDI controller values within a DAW to achieve the same outcome. 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 about the nature of the task, and given a set of test files (audio and MIDI) with which to experiment and practice with for 30 minutes. They were supplied with the audio and the MIDI file for each extract, and they were given 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 anchor for the test was chosen to be the playing of the transcribed MIDI file with no added velocity or controller alternations, as 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.

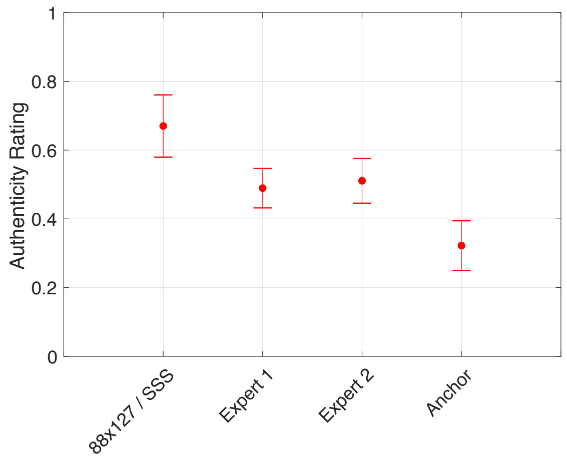

Fig. 5: The combined results of the listening test

4.2 Listening Test

A listening test, compiled using the Web Audio Evaluation Tool (WAET) [17] was carried out, where participants were asked to listen to the original extract, then the four synthesised versions, and to rate them on a slider by deciding how well each version matched the bass guitar performance of the original. 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.

4.3 Results

Figure 5 shows the combined results of all of the extracts, with the same mean and confidence intervals. The authenticity rating represents the overall rating of each version, where 0 = "worst" and 1 = "best". 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 1, where ** = p<0.0001, * = p<0.01, - = p> 0.05.

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

Table 1: Results of a Tukey post hoc test

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

5 DISCUSSION

The results show that the goal stated in the introduction 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. 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. They mentioned that factors such as the unrealistic nature of the pitch variation and how some isolated features which were strikingly different from the original, were "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 from, the overall effect.

The problem with the isolated errors 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 monophonic solo input.

6 SUMMARY AND FURTHER WORK

In this paper, we presented a novel approach to automating the reproduction of audio excerpts using sampler VIs. The findings of the listening test 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 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 learnt parameters to future assignments) would seem necessary for giving the best possible output, especially in the case where a perfect reproduction is not possible.

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 described by a set of parameters.

The specific problem tackled in this paper could be considered 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.

Theoretically, and with enough data and computational power, the "fingerprints" of every musician in an orchestra could be taken, and the recordings generated might accurately predict and be indistinguishable from a recording made by the orchestra themselves.

References

[1] Stephen W Hainsworth and Malcolm D Macleod. “Automatic Bass Line Transcription from Polyphonic Music.” In: *International Computer Music Conference*. 2001, pp. 431–434.

[2] Matti P Ryynänen and Anssi P Klapuri. “Automatic transcription of melody, bass line, and chords in polyphonic music”. In: *Computer Music Journal* 32.3 (2008), pp. 72–86.

[3] Jakob Abeßer and Gerald Schuller. “Instrument-Centered Music Transcription of Solo Bass Guitar Recordings”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.9 (Sept. 2017).

[4] Sebastian Heise, Michael Hlatky, and Jörn Loviscach. “Audio re-synthesis based on waveform lookup tables”. In: *129th Audio Engineering Society Convention*. Vol. 2. Nov. 2010, pp. 1186–1192.

[5] Luwei Yang, Khalid Rajab, and Elaine Chew. “AVA: An Interactive System for Visual and Quantitative Analyses of Vibrato and Portamento Performance Styles”. In: *17th Conference of the International Society for Music Information Retrieval*. 2016, pp. 108–114.

[6] Heejung Lee. “Violin portamento: An analysis of its use by master violinists in selected nineteenth-century concerti”. In: *9th International Conference on Music Perception and Cognition*. 2006.

[7] Thomas Mitchell. “Automated evolutionary synthesis matching”. In: *Soft Computing* 16.12 (2012), pp. 2057–2070.

[8] Janne Riionheimo and Vesa Välimäki. “Parameter Estimation of a Plucked String Synthesis Model Using a Genetic Algorithm with Perceptual Fitness Calculation”. In: *EURASIP J. Appl. Signal Process.* 2003 (Jan. 2003), pp. 791–805.

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

[10] Sebastian Heise, Michael Hlatky, and Jörn Loviscach. “Automatic cloning of recorded sounds by software synthesizers”. In: *127th Audio Engineering Society Convention*. 2009.

[11] Sebastian Heise, Michael Hlatky, and Jörn Loviscach. “Editing MIDI Data Based on the Acoustic Result”. In: *127th Audio Engineering Society Convention*. 2009.

[12] Gerald Schuller, Jakob Abeßer, and Christian Kehling. “Parameter extraction for bass guitar sound models including playing styles”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. 2015, pp. 404–408.

[13] Graham Coleman, Esteban Maestre, and Jordi Bonada. “Augmenting Sound Mosaicing with Descriptor-Driven Transformation”. In: *13th Int. Conference on Digital Audio Effects (DAFx-10)* (2010), pp. 494–497.

[14] Matthew Hoffman and Perry Cook. “The Featsynth framework for feature-based synthesis: Design and applications”. In: *International Computer Music Conference* (2007), pp. 184–187.

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

[16] Anssi Klapuri and Manuel Davy. *Signal processing methods for music transcription*. Springer Science & Business Media, 2007.

[17] Nicholas Jillings et al. “Web Audio Evaluation Tool: A browser-based listening test environment”. In: *12th Sound and Music Computing Conference*. July 2015.