The University of Oxford

Engineering Science

4YP Project Report

- Using Machine Learning to Control Software Synthesisers -

Candidate Number: 355136

April 5, 2018

FINAL HONOUR SCHOOL OF ENG

DECLARATION OF AUTHORSHIP

You should complete this certificate. It should be bound into your fourth year project report, immediately after your title page. Three copies of the report should be submitted to the Chairman of examiners for your Honour School, c/o Clerk of the Schools, examination Schools, High Street, Oxford.

Name (in capitals): ……………………………………………………………………………………………

College (in capitals): …………………………………… **Supervisor:** ……………………………………………

Title of project (in capitals): …………………………………..…………………………………………………

Page count (excluding risk and COSHH assessments): ………………………………………………………

Please tick to confirm the following:

I have read and understood the University's disciplinary regulations concerning conduct in examinations and, in particular, the regulations on plagiarism (*The University Student Handbook. The Proctors' and Assessors' Memorandum, Section 8.8*; available at https://www.ox.ac.uk/students/academic/student-handbook)	☐
I have read and understood the Education Committee's information and guidance on academic good practice and plagiarism at https://www.ox.ac.uk/students/academic/guidance/skills.	☐
The project report I am submitting is entirely my own work except where otherwise indicated.	☐
It has not been submitted, either partially or in full, for another Honour School or qualification of this University (except where the Special Regulations for the subject permit this), or for a qualification at any other institution.	☐
I have clearly indicated the presence of all material I have quoted from other sources, including any diagrams, charts, tables or graphs.	☐
I have clearly indicated the presence of all paraphrased material with appropriate references.	☐
I have acknowledged appropriately any assistance I have received in addition to that provided by my supervisor.	☐
I have not copied from the work of any other candidate.	☐
I have not used the services of any agency providing specimen, model or ghostwritten work in the preparation of this project report. (See also section 2.4 of Statute XI on University Discipline under which members of the University are prohibited from providing material of this nature for candidates in examinations at this University or elsewhere: http://www.admin.ox.ac.uk/statutes/352-051a.shtml.)	☐
The project report does not exceed 50 pages (including all diagrams, photographs, references and appendices).	☐
I agree to retain an electronic copy of this work until the publication of my final examination result, except where submission in hand-written format is permitted.	☐
I agree to make any such electronic copy available to the examiners should it be necessary to confirm my word count or to check for plagiarism.	☐

Candidate's signature: …………………………………………….. **Date:** ………………………..

Abstract

Software Synthesisers (Soft-Synths) are computer applications which create sounds in response to musical input typically in the form of MIDI messages. They are widely used in a variety of musical contexts. Typical soft-synths have hundreds or thousands of parameters which control the sound generation algorithm, allowing the user to create sounds that suit their musical needs. This results in a high dimensional, non-linear search space which the user must navigate in order. Research indicates that humans are bad at navigating such interfaces with serial controls, and that there is a strong link between interface design and the level of creativity that the user with experience when using the interface.
This work describes a novel interface designed to help users control synthesisers in a quicker and more intuitive manner. It combines 3 interfaces together: a traditional ’knob and slider’ interface, a search space visualisation interface, and an iterative blending interface.

THIS ABSTRACT NEEDS A LOT MORE WORK, WILL WORK ON AFTER WRITING MORE OF THE REPORT

Contents

Chapter 1

Introduction

1.1 Background Information

Software Synthesisers (Soft-Synths) are computer applications which create sounds in response to musical input typically in the form of MIDI messages. They are widely used in a variety of musical contexts. Typical soft-synths have hundreds or thousands of parameters which control the sound generation algorithm, allowing the user to create sounds that suit their musical needs. This results in a high dimensional, non-linear search space which the user must navigate in order. Research indicates that humans are bad at navigating such interfaces with serial controls, and that there is a strong link between interface design and the level of creativity that the user with experience when using the interface.

This work describes a novel interface designed to help users control synthesisers in a quicker and more intuitive manner. It combines 3 interfaces together: a traditional 'knob and slider' interface, a search space visualisation interface, and an iterative blending interface.

1.2 Aims of the project

Aim to ...

1.3 Methodology

dtsadfdsafdsaf

Chapter 2

Literature Review

2.1 Previous studies of synthesiser parameter spaces

Several previous...

2.2 Previous attempts of using machine learning to control synthesisers

There have been several previous attempts...

2.3 Description of HCI design principles for creative musical interfaces

A number of guidelines for creative musical interfaces

Chapter 3

Description of Synthesiser and Image Processing Algorithms

3.1 Synthesiser Algorithm

3.1.1 Description

In order to develop the synthesiser controller, it was necessary to choose a synthesiser to work with. To remove some of the complexities of interfacing with many commercial synthesisers, a purpose built synth was made to make it as easy as possible to work with the interface, whilst being representative of typical soft-synths available. The programming language Supercollider was used to create a 6 operator Phase Modulation synthesiser, loosely based on the Yamaha DX7, a very famous synthesiser from the 1980s (CHECK DATE).

The synthesis algorithm is based around the FM7 UGen for Supercolldier [5], which implements 6 operators with independent amplitudes and frequencies, whose outputs can be used to modulate the phase of any of the operators. The implementation can be simply described by the following discrete time equation: FIND REFERENCE

$$y_i[t+1] = a_i * sin(2\pi T f_i + \sum_{j=1}^{6} y_j[t] * m_{ij}) \tag{3.1}$$

T is the sampling interval, $y_i[t]$ is the output of operator i at time t, a_i and f_i are the amplitude and frequency of operator$i$ and m_{ij} is a scalar parameter determining the level of phase modulation from operator j to operator i. This is a very flexible sound generation algorithm which can create a large number of different sounds.

The full synthesiser architecture can be described as follows.

A MIDI Note ON message is received with a musical note number N and a velocity V. This triggers the sound generations process for this particular note. the note number N is converted into a base frequency F. The frequency of each operator is set using the equation:

$$f_i = F * f_i^{coarse} * (1 + f_i^{fine}) \tag{3.2}$$

The coarse frequency parameter f_i^{coarse} for operator i can take discrete values $\{$0.5, 1, 2, 3, ...$\}$, allowing the operators to set to different harmonics of the base frequency. The fine frequency parameter f_i^{coarse} for operator i can take any value in the range [0,1], and is used to detune operators away from the perfect harmonic ratios. This is the approach usually taken in typical synthesisers when setting frequencies, as usually the fine frequency values will be very small. REFINE THIS BIT

The base frequency F can be continually modulated by the MIDI PitchBend control, and by a vibrato envelope (an exponential ramp from a start frequency and amplitude to an end frequency and amplitude, over a time period in the range [0,20] seconds).

The operators' amplitudes a_i are then continuously modulated by two low frequency oscillators (LFOs), and a modulation envelope. LFO A is a zero mean triangle wave oscillator with a frequency in the range [0, 20Hz], amplitude in the range [0, 1], and phase spread in the range [0, 1] (a parameter which allows the LFO's initial phase to be randomly varied). LFO B is a zero mean square wave oscillator with a frequency in the range [0, 20Hz], amplitude in the range [0, 1], and pulse width in the range [0, 1]. The modulation envelope is an ADSR (Attack-Decay-Sustain-Release) style envelope generator, which is triggered by the MIDI Note ON message. This can be described by the equation:

$$a_i[t] = (1 + LFO^a[t] * lfo_i^a) * (1 + LFO^b[t] * lfo_i^b) * (ENV^{mod}[t] * env_i^{mod} + (1 - env_i^{mod})) \tag{3.3}$$

where $LFO^a[t]$ and $LFO^B[t]$ are the LFOs' output values at time t, ENV[t] is the modulation envelope's output value at time t, and lfo_i^a, lfo_i^b, and env_i^{mod} are parameters which determine how much operator i is affected by the respective signals.

After amplitude modulation with the LFOs and the modulation envelope, the operators are fed into the previously described phase modulation algorithm. The 6 outputs from this algorithm are then mixed together, and multiplied by the Amplitude Envelope to form the output signal: $Y[t] = ENV^{amp}[t] * \sum_{i=1}^{6} y_i[t] * a_i^{output}$, where a_i^{output} is the Output Level parameter for operator i, and the Amplitude Envelope is another ADSR Envelope generator, similar to the Modulation Envelope.

All of the synths parameters can be easily adjusted in real time, by sending messages through the Open Sound Control (OSC) protocol, a UDP based protocol for sending messages between applications. CITE UDP

A selection of 35 presets were created for this synth, to give a selection of all the different kinds of 'nice' sounds the synth can make, and to provide data to design the interface with.

3.1.2 Design choices and justification

Typical commercial synthesisers have a large number of parameters (Yamaha DX7 - 126 parameters, Omnisphere 2 - several thousand parameters), which can be discrete or continuous. Some synthesisers have Modular or Semi-Modular architectures, which allow for a lot of flexibility, but due to the combinational explosion of number of parameter combinations, and difficulty blending between presets, this type of architecture is not addressed in the work.

The type of synth that this work is aimed at is 'analogue-style' synths, in which there is a medium number of parameters, with a fixed internal routing. The synthesiser features a combination of continuous and discrete parameters, both of which the interface must be able to cope with.

The synthesiser was designed to have access to as wide a range of sounds as possible with as few parameters as possible. For example instead of having a separate envelope generator for each oscillator (as in the Yamaha DX7), only 2 envelopes were included, with a blendable amount per operator. The synth has 96 total parameters, compared to the 126 of the DX7, but it can replicate most of the useful sounds of a DX7.

In Yee King's thesis CITE PROPERLY, a similar FM7 Ugen based Supercollider synthesiser was used, but no envelopes or LFOs were included, so that it was limited to producing sounds with a static timbre. Whilst this had advantages for the timbre mapping approach of the work, it resulted in a synthesiser that was not very representative of ones typically used, as envelopes are such an important part of designing sounds.

An interesting property of this synthesiser design is that due to the 6 identical operators which can be configured in any combination, there is lots of possible permutation ambiguity, as if two operators have all of their parameters switched, the sound will remain identical. This presents a challenge as two presets can have very different parameters despite having the exact same sound.

3.2 Image Processing Algorithm

3.2.1 Description

asdasdasda

3.2.2 Design choices and justification

asdasdas

Chapter 4

Description of Full Interface

4.1 Traditional Interface

asdasdasdasd

4.2 PCA Interface

asdasdasdsa

4.3 Blending Interface

asdasdasdsa

4.4 How the interfaces are combined

asdasdasdadas

Chapter 5

Design and Evaluation of Interfaces

5.1 Traditional Interface

5.1.1 Detailed Description

asdasdasd

5.1.2 Strengths

asdasdad

5.1.3 Weaknesses

asdasdsa

5.2 PCA interface

5.2.1 Detailed Description

asdasdassda

5.2.2 Global PCA vs Time/Timbre PCA

asdasdasd

5.2.3 PCA + Histogram Equalisation Description

asdsadsa

5.2.4 Demonstrations of preset group clustering

asdasdsad

5.2.5 Investigate how the PCA mapping scales with number of presets

asdasdasdsad

5.2.6 Quantify the extra variance the macro controls give

asdasdasdsa

5.2.7 Investigate Permutation Ambiguity

asdasdsa

5.3 Blending Interface

5.3.1 Detailed Description

asdasdasd

5.3.2 Strengths

asdsadsa

5.3.3 Weaknesses

asdasdsad

5.3.4 Development / Verification using Image Editing Interface

asdasdsadsa

Tests of Image Comparison Metric

asdasdasd

Convergence Tests for different preset generation algorithms

asdasdasd

5.3.5 Investigate Permutation Ambiguity

asdasdsad

5.4 Comparisons of interfaces

asdasdsad

5.4.1 Perfect /Imperfect User Model

asdasdsadad

5.4.2 Comparison of isolated interfaces

asdasdsad

5.4.3 Comparison of combined interfaces

asdasdsadsa

Chapter 6

User tests and Interviews

asdasdasdsadasd

Chapter 7

Conclusion

The interface proposed in this project has many benefits over a traditional synthesiser interface, as has been designed following design heuristics from the fields of Human Computer Interaction and Creative Cognition. Based on simulated user studies, the interface has at least as good performance as a traditional interface when carrying out search based tasks, and based on real user feedback it has many advantages in terms of creativity.

7.1 Further Work

The evaluation of this interface was only carried out on a single synthesiser, due to the projects time constraints. The interface has been designed to be as general purpose as possible, allowing it to control arbitrary synths over the OSC protocol, but due to a lack of standardisation between soft synths, and lack of implementation time, more work needs to be done to create a truly general purpose soft synth controller. As many soft-synths are in the VST format, making a version of the interface with acts as a VST host, and uses the parameter retrieval and preset storage functionality of VSTs is a good next goal if this project is continued in the future.

Bibliography

[1] Martin, R. (2009). Clean Code. 1st ed. Upper Saddle River, NJ: Prentice Hall, pp.36-52.

[2] En.wikipedia.org. (2017). Levenberg-Marquardt algorithm. `https://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm`

[3] En.wikipedia.org. (2017). Optimisation Algorithms & Methods. <`https://en.wikipedia.org/wiki/Category:Optimization_algorithms_and_methods`>

[4] Introduction to Object Oriented Programming in Matlab `https://uk.mathworks.com/company/newsletters/articles/introduction-to-object-oriented-programming-in-matlab.html`

[5] Stefan Kersten. skUG SuperCollider UGen library., 2008. `http://space.k-hornz.de/software/skug/`.