The University of Oxford

Engineering Science

4YP Project Report

- Using Machine Learning to Control Software Synthesisers -

Candidate Number: 355136

April 9, 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.

A video of the full interface being used can be viewed at (LINK TO VIDEO).

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...

Talk about the preset blending study, including the parameter freezing.

Talk about the google sample mapping with TSNE (infinite drum machine).

Talk about how taking a sound sample from each preset could be a good way to do the dimensionality reduction interface, but how a parameter based approach is preferable in many ways. Mention which sound metrics would be good to use, and the difficulties involved.

2.3 Description of HCI design principles for creative musical interfaces

A number of guidelines for creative musical interfaces

2.4 Principal Component Analysis

Describe the PCA algorithm, and dicuss the possible extention to using Sparse PCA.

2.5 Bayesian Optimisation

Discuss how this is a promising technique in this field, for example preference gallery interface, but due to large ammount of training data neccesary, and bad performance in high dimensions it wont be used.

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

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

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.

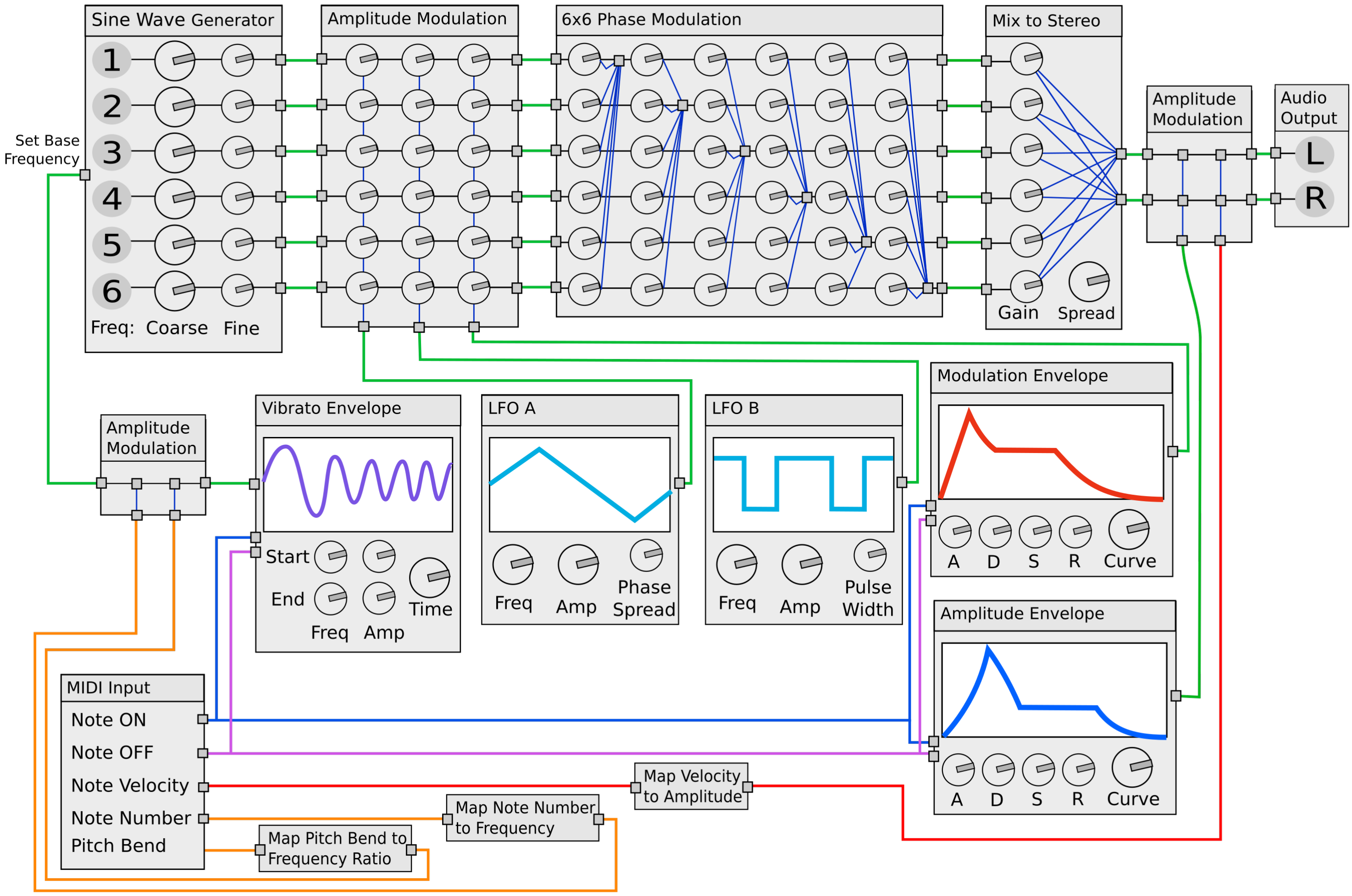


Figure 3.1: Synthesiser Schematic

3.1.2 Synthesiser Parameters

This synthesiser has 96 parameters, as described in Table 3.1.

USE TABLE TO CLEAN UP PREVIOUS SECTION

Table 3.1: Synthesiser Parameters

Parameter Group	Parameter	Domain	Time/Timbre	Weighting
Phase Modulation	36 x m_{ij}	$[0, 10]$	Timbre	3
Coarse Frequency	6 x f_i^{coarse}	$\{0.5, 1, 2, 3, 4, ...\}$	Timbre	10
Fine Frequency	6 x f_i^{fine}	$[0, 1]$	Timbre	10
Output Levels	6 x a_i^{output}	$[0, 1]$	Timbre	3
Modulation Envelope Amount	6 x ENV_i^{mod}	$[0, 1]$	Timbre	1
LFO A Depth	6 x LFO_i^a	$[0, 1]$	Timbre	1
LFO B Depth	6 x LFO_i^b	$[0, 1]$	Timbre	1
LFO A	Rate Amplitude Phase Spread	$[0, 20]$ Hz $[0, 1]$ $[0, 1]$	Time	3
LFO B	Rate Amplitude Pulse Width	$[0, 20]$ Hz $[0, 1]$ $[0, 1]$	Time	3
Amplitude Envelope	Attack (A) Decay (D) Sustain (S) Release (R) Curve	$[0, 10]$ seconds $[0, 10]$ seconds $[0, 1]$ $[0, 10]$ seconds $[-5, 5]$	Time	10
Modulation Envelope	Same as previous		Time	10
Miscellaneous	Vibrato Start Amt. Vibrato End Amt. Vibrato Start Freq. Vibrato End Freq. Vibrato Time Stereo Spread	$[0, 1]$ $[0, 1]$ $[0, 20]$ Hz $[0, 20]$ Hz $[0, 20]$ seconds $[0, 1]$	Time	1

3.1.3 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, and are usually contrained within a range. 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. Some synthesisers use short audio recordings, known as samples, as part of their sound creation algorithm. This creates difficulties blending between presets, so this type of architecture will not be considered in this work. (However both of these type of algorithms would be interesting to try as a follow up 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, all of which are bounded. Common bounds are: $[0, 1]$, $[0, k]$ and $[-k, k]$, where $k \in \mathbb{R}^+$.

The DX7 was chosen to base the synth off because it is a very famous and powerful synth, but is notoriously difficult to use due to the phase modulation algorithm being very un-intuitive. This means that an improved interface has the ability to open up DX7 programming to a less expert user base. WORK ON THIS

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. Despite aims to reduce parameter count, there are still many redundant parmaeters in most presets for this synthesiser. For example, if the amplitude of one of the LFOs is set to zero, all of the other parameters to do with the LFO will have no effect on the sound. This is a common feature with synthesisers, and presets a challenge when trying to learn useful information from the parameters.

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

The full interface is a combination of 3 interfaces. Each interface has the ability to vary the entire set of parameters, and produce a wide variety of sounds, but they are designed to complement each other as well as possible. The user can easily move back and forth between the different intefaces, and there are consistent parameter visualtions in all interfaces.

4.1 Traditional Interface

The Traditional Interface is designed to be representative of a typical software and hardware synthesisers. It has a knob or slider for each parameter of the synthesiser, and has interactive visualisations for some parts of the architecture (namely the envelopes and LFOs). The interface is arranged in three seperate pages, two of which are shown in Figure 4.1. If the full interface were to be extended to work with any VST, the traditional interface would be replaced by the VST's user interface.

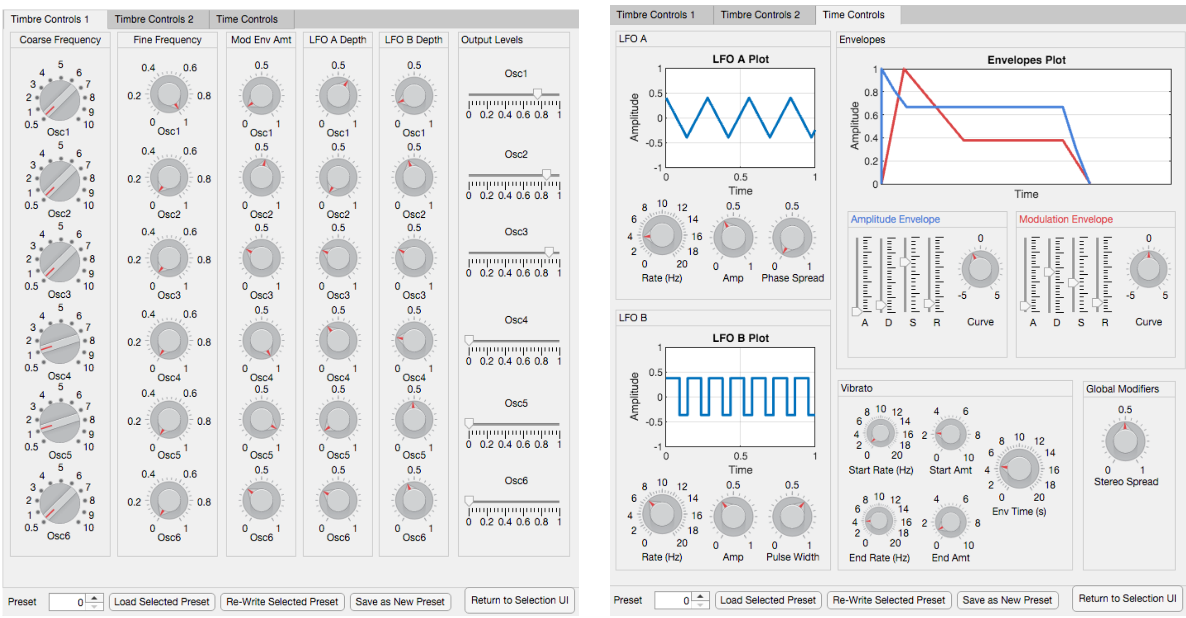


Figure 4.1: Traditional Interface - Screen 1 and 3

4.2 Selection Interface

Typical Soft-Synths are supplied with a large number of presets, which are usually named and arranged into categories. They are usually displayed in a list which can be searched by name, or split into category. In many soft-synths, each preset is given a set (usually 8) of Macro Controls, which each vary single parameter or combinations of parameters, and aim to give the user a quick way to tweak each preset.

The Selection Interface is a new approach to displaying presets, arranging the presets as cells of a 2D voronoi diagram, such that similar presets are close to each other and coloured similarly. The presets can quickly be compared by moving the mouse around the diagram, and presets are selected by clicking on the cells. Once a preset is selected it can be edited by macro controls. The presets have been divided into a number of categories. Clicking the category buttons at the top of the interface highlights the selected category, as shown in Figure 4.2b. By clicking the 'Display Mode' button, it is also possible to limit the presets displayed to just the selected category(s), displayed in a recalculated voronoi diagram.

Both the 2D spacing of the presets, and the creating of the macro controls is calculated automatically from the set of presets' parameter values using Principal Component Analysis (PCA).

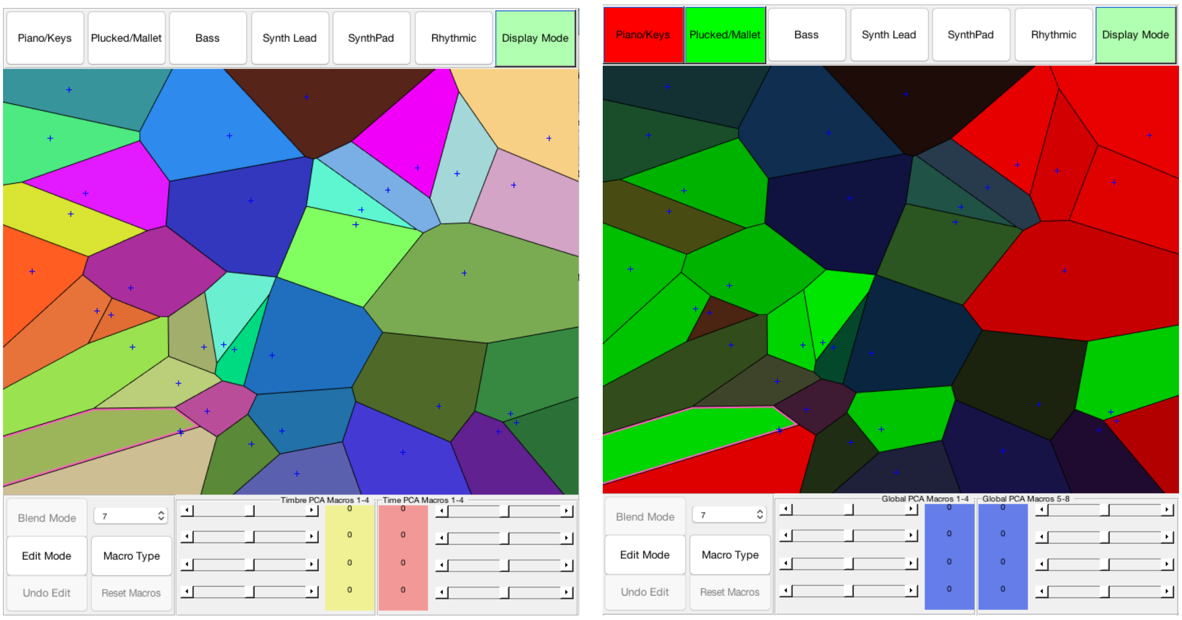


Figure 4.2: PCA Interface - Normal View, and Category Highlight View

4.3 Blending Interface

Preset Blending is a somewhat popular approach in synth interface design, where new presets are created by a linear combination of presets. (EXAMPLES) The Blending Interface is an extension of this approach.

Three initial presets (A, B, C) are selected, and are placed on the coreners of the triangle in the centre of the interface. As the cursor moves over the interface, a new preset is created as a weighted sum of the three presets, based on promixity to the corners. Once the user finds the optimal preset in the space, the user clicks. The preset clicked on then becomes the preset A (I.e is placed on the bottom corner of the triangle), and a new preset B and C are generated. This process is repeated as many times as desired, allowing the user to keep searching through the parameter space.

If the user clicks on the 'Pause on Selected Preset Button', they are shown an interactive display of their past preset choices, which they can use to go back to previously selected settings and resume searching. It is also possible to select three of the previous presets, and assign them to preset A, B and C. The user has the option to freeze sections of the parameter space, which helps the search be more fine tuned to their needs. The colours of the Blending Interface are calculated to be consistent with the Selection Interface.

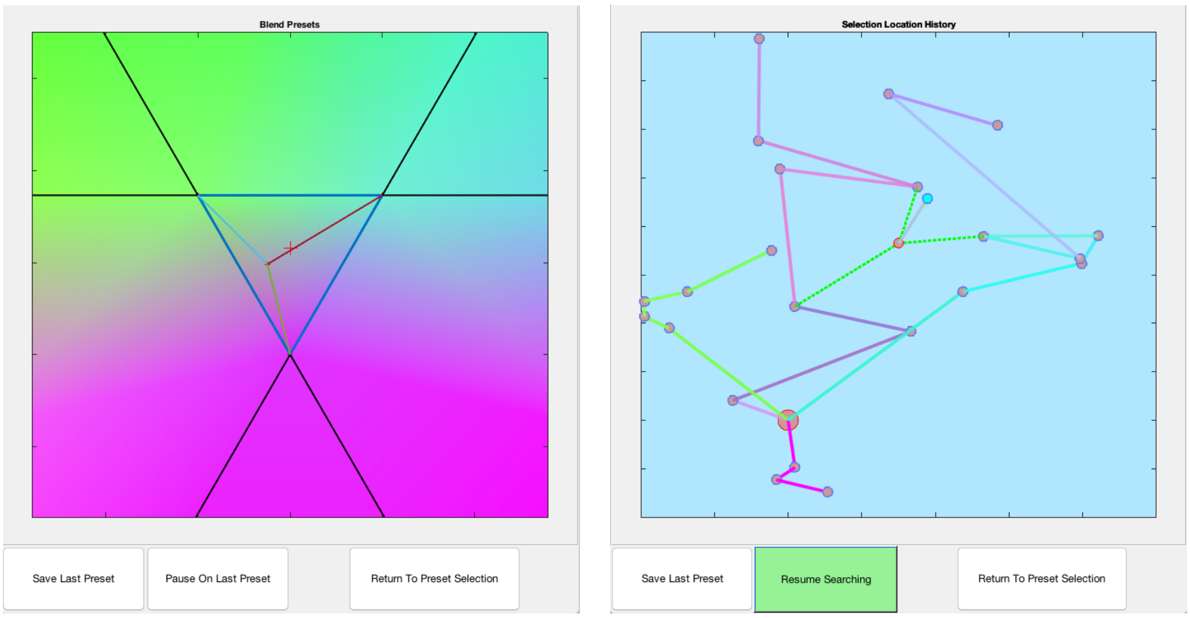


Figure 4.3: Blending Interface - Normal View, and Selection History View

4.4 How the Interfaces are Combined

When the application is started, the Selection interface is displayed first. Once a preset has been selected and possibly modified with the Macro Controls, the user has the option to click the 'Edit Mode' button, which opens the Traditional Interface, allowing the currently selected preset to be further modified. The user can move back to the Selection Interface by clicking the 'Return to Selection UI' button. The varied preset is displayed on the Selection Interface with a marker, attatched to the original preset, which is positioned and coloured to be consistent with the PCA mapping. If the user wants to undo the edits made to the preset they can click the 'Undo Edit' button. In this way the user can move back and forth between the Blending and PCA interfaces as often as desired.

Once three presets have been selected, the user has the option of clicking the 'Blend Mode' button which opens the blending interface, and assigns the three seceted presets to A, B and C. The user can then use the blending interface for as long as necessary, and then click the 'Return to Preset Selection' button. This time another marker is created, with dotted lines to the 3 initially selected presets.

This marker can then be used in the same way as the cells of the voronoi diagram, I.e it can be previewed by clicking, selected by double clicking, can be edited with the Macro Controls or the Traditional Interface. There are a set of parameter visialisations that are diaplayed alongside all of the presets to give the user more feedback into what changes they are making, and to aid in understanding how the synthesis algorithm works.

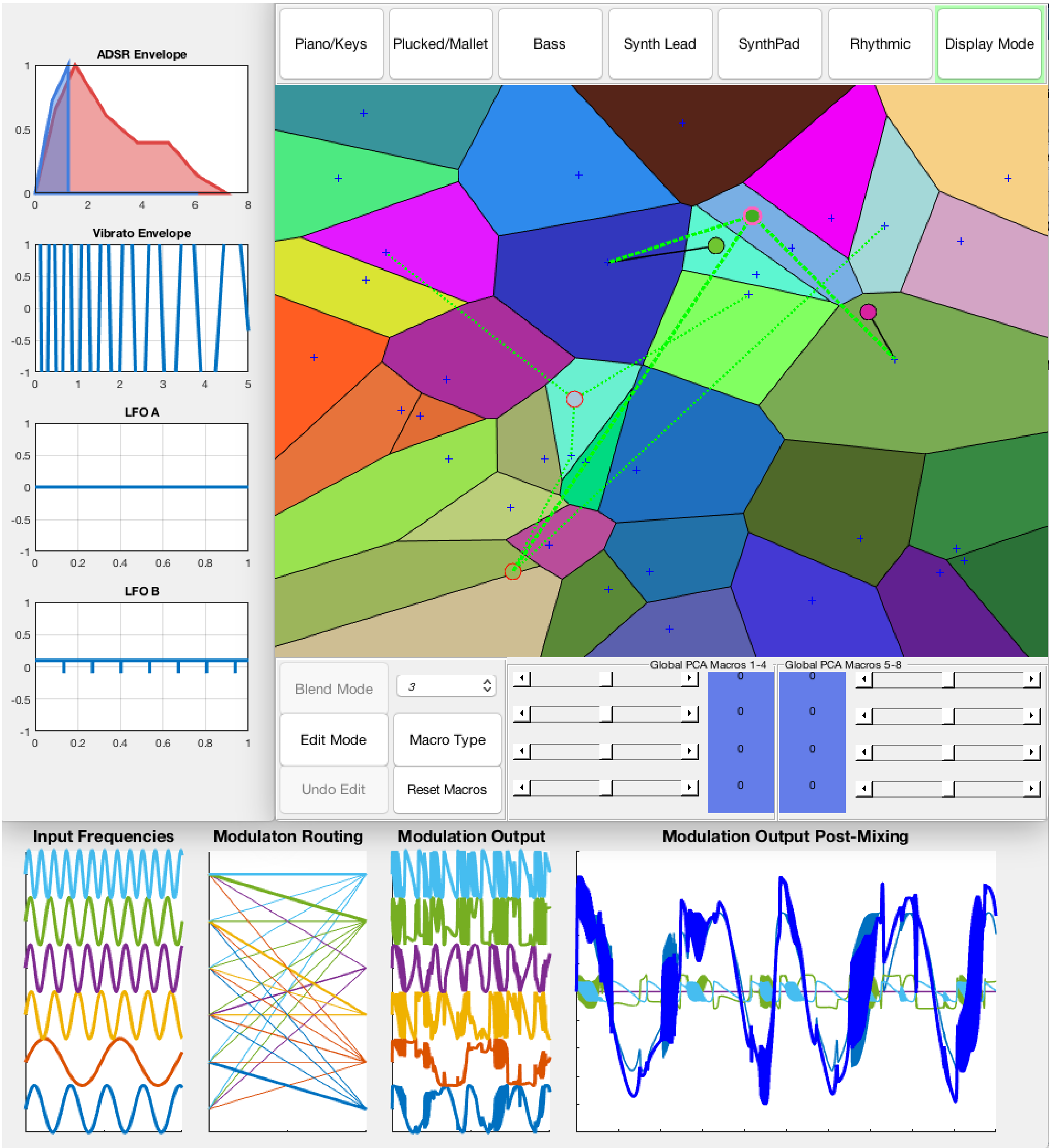


Figure 4.4: Combined Interface - PCA view

Chapter 5

Design and Evaluation of Interfaces

5.1 Traditional Interface

This interface was created in the MATLAB App Designer toolbox. As the parameters are varied, they are sent in real time over OSC to the synthesiser.

Figure 5.1: Traditional Interface - Screen 2

5.1.1 Strengths

A skilled user can use knowledge of synthesis to construct any preset they can conceive of. The entire parameter space can be searched. For sectional of the synthesiser archececture that are more intuitive, such as the LFOs and envelopes, this works very well.

5.1.2 Weaknesses

Slow serial control of all of the parameters. Even if the user knows exactly what each parameter should be set to, it will still take several minutes to go through all of the 96 knobs and set them to the correct value. Parts of the synthesis architecture are non-intuitive and fiddly to use, In particular the 36 phase modulation routing parameters shown in Figure 5.1. If the user doesn't know exactly what they want, or how to acheive a particular sound, this interface can be extremely inefficient and frustrating.

5.2 Selection Interface

This interface was created as a Matlab 'figure based application'.

5.2.1 Global PCA vs Time/Timbre PCA

The PCA is calculated in two different ways for different sections of the interface. In 'Global PCA', each of the 36 presets' 96 parameters arranged in a $36 * 96$ array. PCA is carried out on this array to produce the Global PCA Weights and Scores.

In Time/Timbre PCA, the 96 parameters are partitioned into two sets: 72 which affect timbre, and 24 which affect variation of the sound over time (I.e all of the envelope and LFO parameters). PCA is then carried out seperately on the resulting $36 * 72$, and $36 * 24$ arrays to produce the Time PCA Weights, and the Timbre PCA Weights.

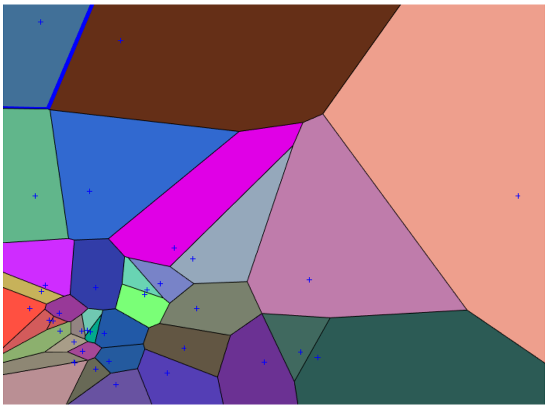
The Global PCA Weights are used for the XY-RGB mapping of the Voronoi diagram, the Global PCA Macros, and for colouring the Blending Interface. The Time/Timbre PCA Weights are just used for the Time/TImbre Macro Controls.

The user is given the option to switch between the Global, and TIme/Timbre Macro Controls, as the global controls allow a greater ammount to the space to be searched, but the Time/Timbre controls can be more understandable. The Macro controls are the same for all presets, but centered on the currently selected preset. I.e. they allow a relative change of parameters not an absolute parameter change.

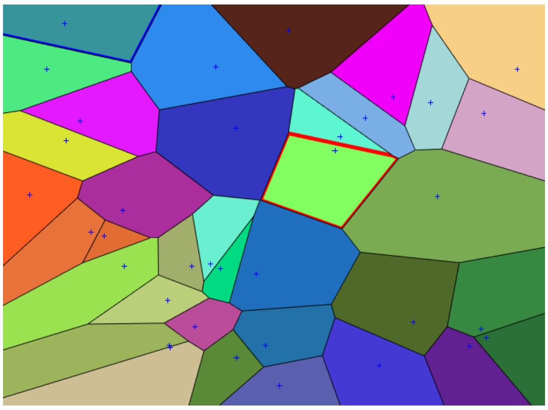
A possible extension to this approach is to allow parameter freezing, as in the blending interface, and have the PCA Macros automatically be recalculated based on the currently unfrozen parameters. Another possible extension is to create a unique set of macro controls for each preset, which may make macro controls more useful, but at the sacrifice of consistency between presets.

5.2.2 PCA + Histogram Equalisation Description

When using the Global PCA scores to calculate the XY-RGB position of each preset in the Voronoi diagram, there are some undesirable characteristics of the mapping produced. The sizes of the cells varies dramatically, and the majority of the presets usually get compressed



(a) PCA alone.



(b) PCA + Histogram equalisation

Figure 5.2: Selection Interface before and after Histogram Equalisation

into one of the corners, see Figure 5.2a This is because PCA is linear, and so 'outliers' will skew the diagram, so after rescaling to $[0, 1]$ to fit in the diagram, the 'inliers' will be compressed to a smaller range'. A similar effect occurs with the colour mapping, causing many of the colours to be similar to each other minimising the dynamic range of the diagram. (MORE DETAIL HERE WITH REFERENCE). In user interface design, users usually associate larger icons with greater importance, therefore it would be preferable for all the presets to be of a similar size, so as not to give uninteded meaning to particular presets.
To fix this issue, a variant of an image processing method known as histogram equalisation (REFERENCE) was used, resulting in the mapping shown in Figure 5.2b.

The histogram equalisation in one dimension is described in Figure 5.3. A histogram is created

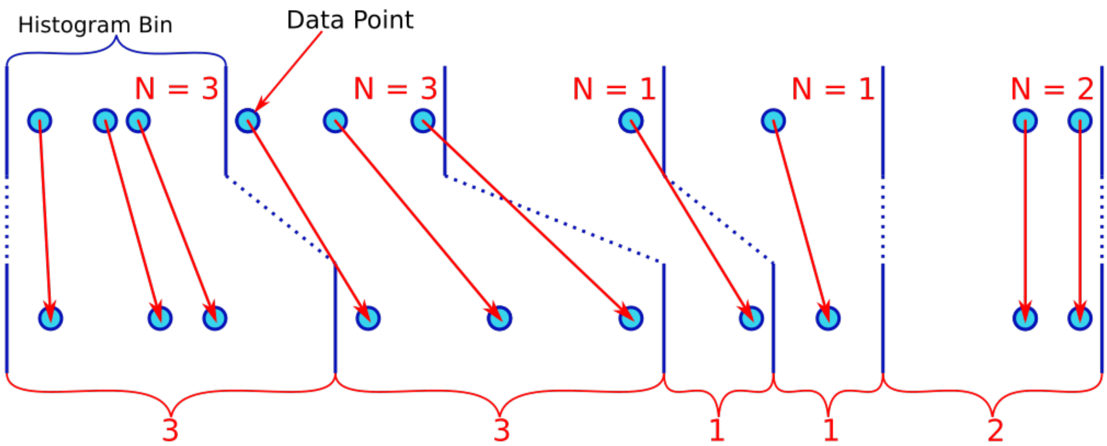


Figure 5.3: Histogram Equalisation

by splitting the data into a number of eqully spaced bins (6 bins was empirically found to work well in this case). The bins are then rescaled, such that the bin width is equal to the number of data points in the bin).

This process is carried out for the presets in all 5 of the XY-RGB dimensions, and then each dimension is mapped to the range $[0.05, 0.95]$, such that it will fit inside the axes of $[0, 1]$.

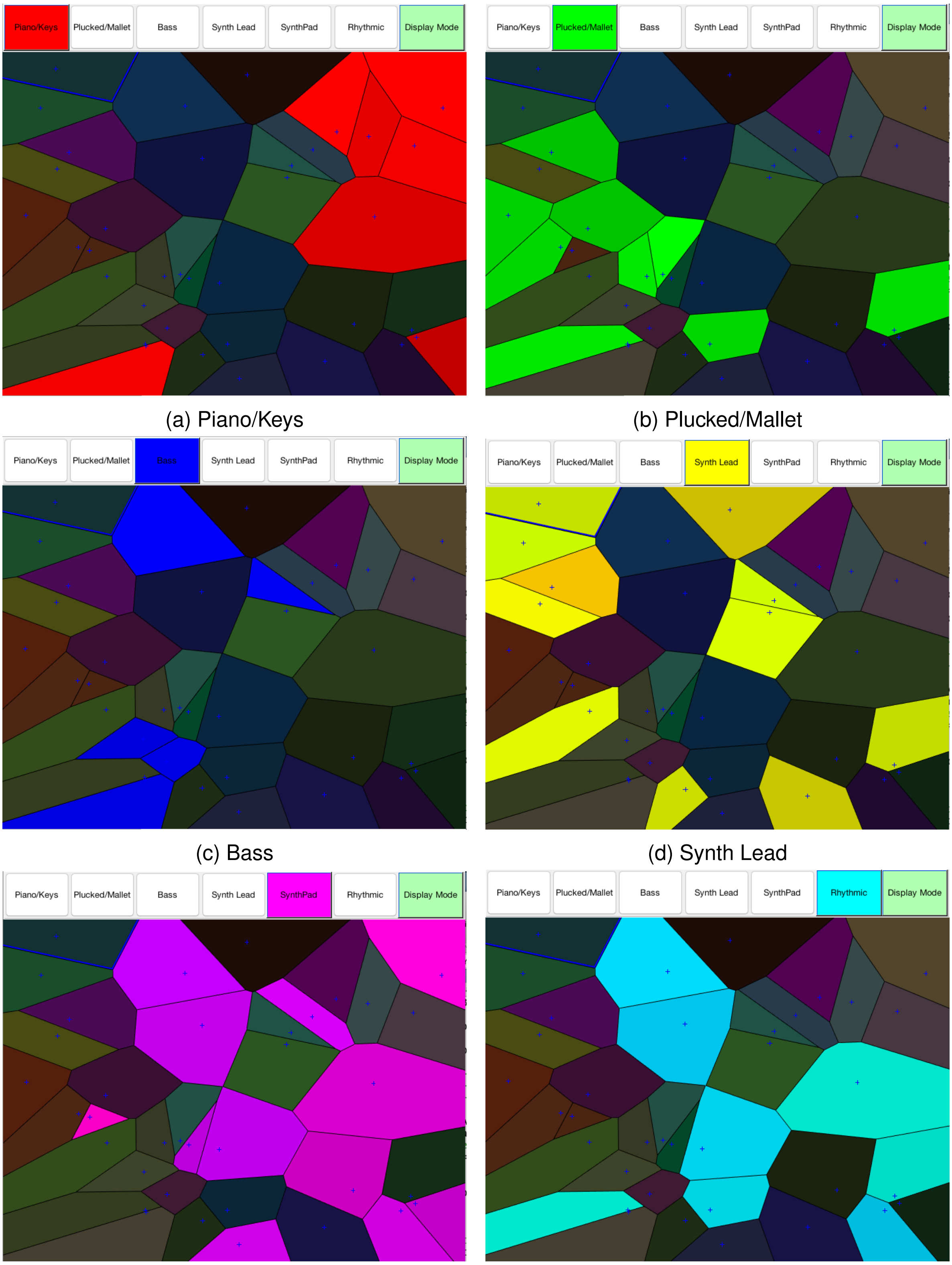
This approach, although simple, effectively redistributes the presets. As long as one of the bins are empty, the mapping in each dimension is peicewise linear and continuous. This means that the Combined Preset Markers can be positioned correctly on the graph by first calculating the PCA scores, then passing these scores through the mapping function. Due to the continuity of the mapping function, the Macro Controls will cause the preset markers to move smootly across the graph. ... FINISH WRITING

5.2.3 Demonstrations of preset group clustering

Having divided the presets into a set of (non-exlusive) categories, the clustering nature of the PCA process can be tested by viewing where the categories are placed in the diagram, see Figure 5.4. This demonstrates that the PCA process does indeed cluster the categories together, although some are better clustered than others. Part of this is due to the fact that the categorys are somewhat subjective, for example the decision between Synth Lead and Synth Pad, or between Piano/Keys and Plucked/Mallet.
Some of the categories also include a wide range of sounds, especially Rhythmic and Synth Lead, and so have a less consistent pattern in their parameters. It is also hard to draw colcusions about the clustering performance, due to the small sample size of presets, and the fact that many of the presets were made by using the Blending interface, so potentially have overly similar paramaters than if they'd been made from scratch by a person. To validate this approach more thoroughly, it would need to be applied to many 'real world' sets of presets of pre-existing commercial synthesisers, but several technical challenges (In particular lack of standardisation of control interfaces MORE DETAIL HERE) prevented this being possible during the sort timescale of this project.

5.2.4 Investigate how the PCA mapping scales with number of presets

asdasdasdsad



(a) Piano/Keys (b) Plucked/Mallet

(c) Bass (d) Synth Lead

(e) Synth Pad (f) Rhythmic

Figure 5.4: Preset Categories shown on Selection Interface

5.2.5 Quantify the extra variance the macro controls give

asdasdasdsa

5.2.6 Investigate Permutation Ambiguity

asdasdsa

5.3 Blending Interface

5.3.1 Detailed Description

This interface was created as a Matlab 'figure based application'.

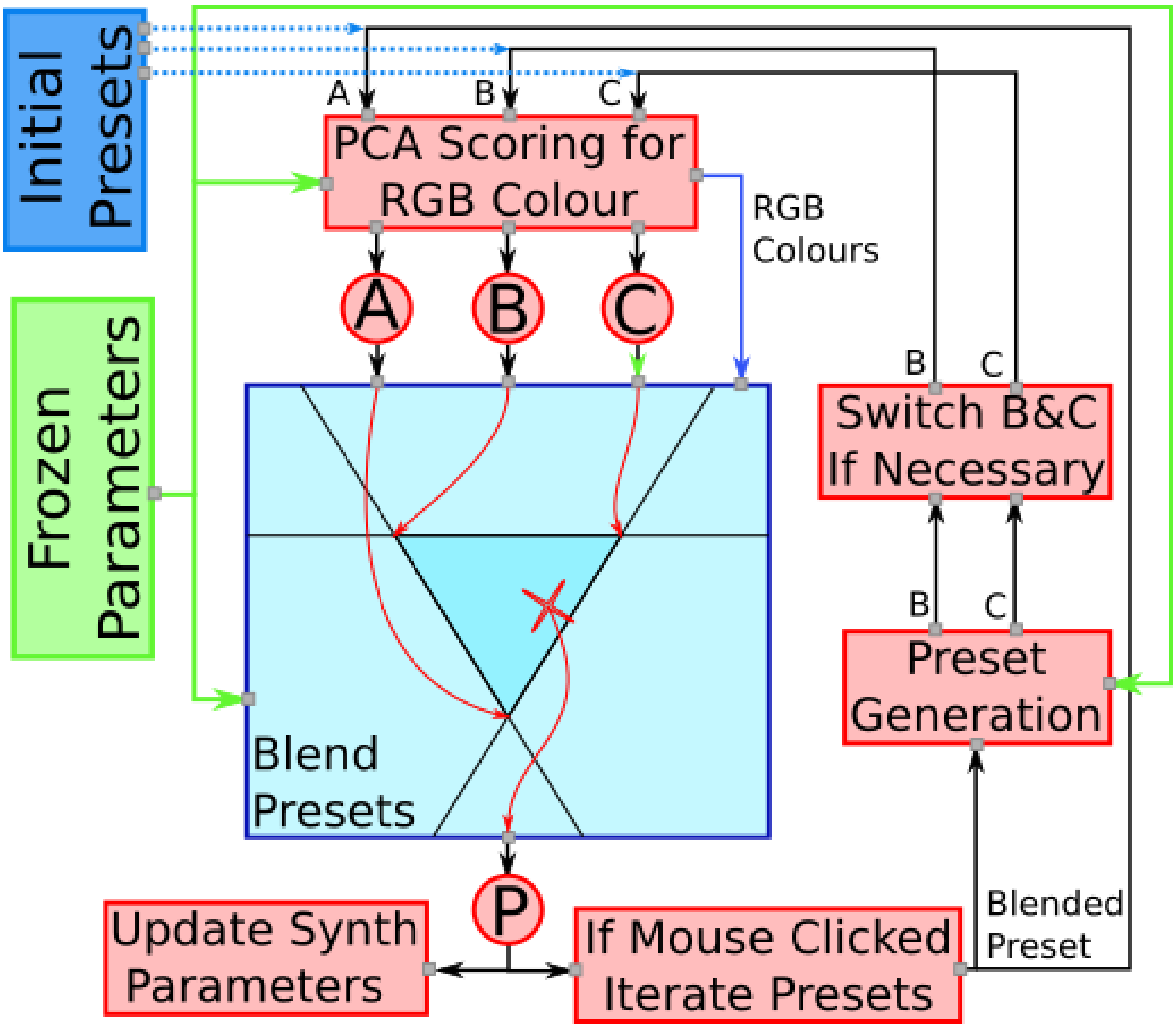


Figure 5.5: Blending Algorithm flow chart

The blending of presets is calculated as a non orthogonal vetor decomposition: (REFERENCE!)

$$\mathbf{P} = f(\alpha\mathbf{A} + \beta\mathbf{B} + \gamma\mathbf{C}) \tag{5.1}$$

Where $\mathbf{P}$ is the blended preset, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are presets A, B and C, f is a function which applies parameter constraints, and α, β and γ are calculated from the following matrix equation:

$$[\beta, \gamma]^T = (\mathbf{M}^T * \mathbf{M})vecM^T * [x, y], \qquad \alpha = 1 - (\beta + \gamma) \tag{5.2}$$

where x, and y are the x and y coordinates of the current cursor position, and $\mathbf{M} = [\mathbf{b}, \mathbf{c}]$, where $\mathbf{b}$, $\mathbf{c}$ are vectors from the A preset location to the B and C preset locations on the interface, as shown in Figure 5.6.

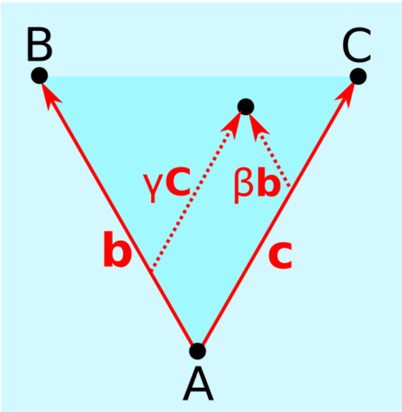


Figure 5.6: Non-orthogonal vector decomposition

The parameter constraint function f, applies the relevant constraints to each parameter. Most parameters are continuous with an upper and lower bound, $[l, u]$. For these parameters: $f_i(P_i) = \min(\max(P_i, l), u)$. For the Coarse frequency parameters, the continuous blended value is discretised back into the set $0.5, 1, 2, 3, ...$, which the desicion boundary between neighbouring number falling equidistant to the numbers, as shown in Figure 5.7.

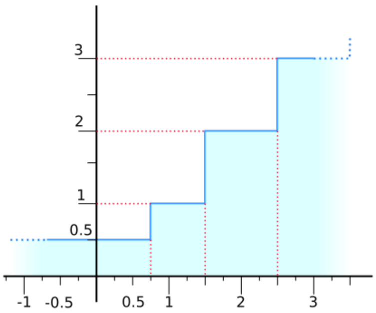


Figure 5.7: Mapping from continuous to coarse frequency

To make the iterative blending process more intuitive, and to maintain consistency between the interfaces, the RGB colouring using PCA from the Selection interface was used. This is done by using Equation 5.1 to calculate the blended preset at each of the points marked on Figure 5.8, then calculateing the PCA Scores for each of these points. $PC_{3,4,5}$ are mapped to RGB colour, and colour is linearly interpolated between the points. A more detailed colour space could be acheived by using more points, but there would be an associated performance tradeoff.

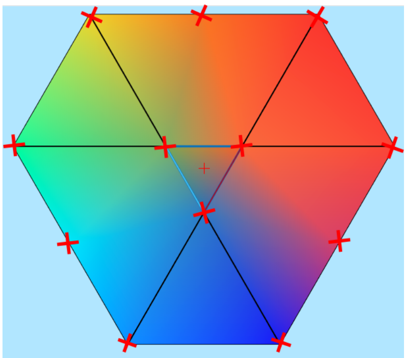


Figure 5.8: $PC_{3,4,5}$ applied to RGB colour of blending interface

Each time the presets ae iterated, and a new B and C are generated, the first principal component, PC_1 is calculated for B and C, and if $PC_1^B > PC_1^C$, then presets B and C are swapped. The effect of this is that the x direction in the Blending Interface corresponds to a direction of increasing PC_1. This is done because PC_1 is also used in the Selection Interface for the x coordinate of each preset, and so this process further maintains consistency between the two interfaces.

The Selection History display, allows the user to browse their previously selected presets, by clicking on nodes of the graph. Each time presets are iterated, the vector from A to the selelected coordinates, $\mathbf{p}_i$ is recorded. The graph is constucted by joining these vectors head-to-tail, as shown in Figure 5.9. The colours of each edge of the graph are chosen based on $PC_{3,4,5}$ of the presets. See Figure, 4.3 for a typical example of the Selection History plot.

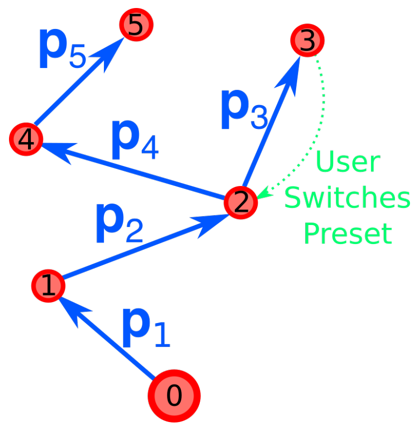


Figure 5.9: Selection History plot construction

It may be possible to use some of the information contained in the graph (if the data from many users is collected), as a way to refine the preset generation algorithm. If the generation algorithm is producing good presets, then most of the choices should be somewhere between A, B and C on the interface. If the algorithm is producing bad presets, most of the choices will be very close to, or below A. MORE DETAIL HERE

As part of the blending interface, the user has the option to freeze sections of the parameter space. The parameters are divided into 'Time' and 'Timbre' parameters, and then subdivided into a total of 12 smaller categories, see Figure 5.10. This makes the blending interface more useful, as it allows users with knowledge of synthesisers to fine-tune the blending process to their needs. This tends to lead to more useful generated presets, as it reduces the dimensionality

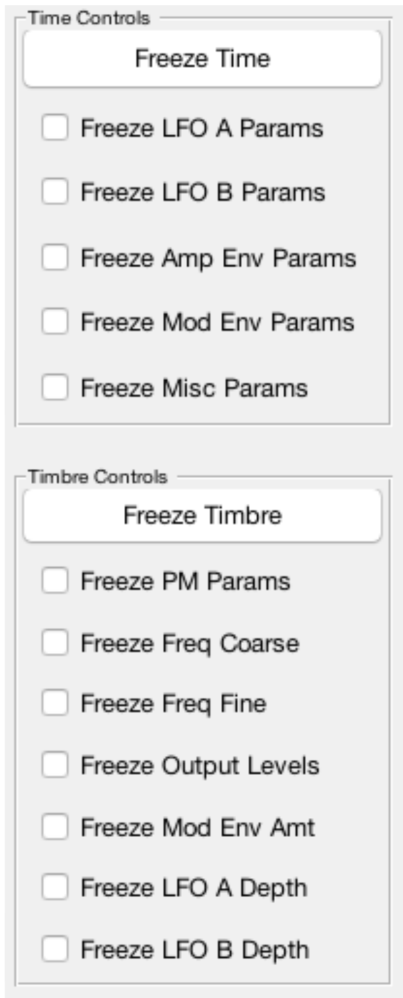


Figure 5.10: Parameter Freezing

of the search space (BACK THIS UP).

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

Due to the subjectivity of interface design, and as the three interfaces have quite different purposes, there are not many obvious metrics to use to compare the interfaces with each other. However, the key challenge is to make the combined interface better than the traditional interface alone. MORE WRITING

5.4.1 Perfect /Imperfect User Model

A way of evaluating the performance of the interface is to simulate a 'Perfect User', and an 'Imperfect User' carying out particular tasks. In particular the task is to use the interface to move from an initial preset, to a goal preset. This analysis won't account for any creativity based goals of the interface, but will help evaluate some of the practical comsiderations of the interface. (BAD WORDING¡ WORK ON THIS SECTION).

The Perfect User has perfect knowledge of the synth and the interface, so can always choose the optimum position for a particular knob, or choose the optimum blend of presets in the Blending Interface.

The Imperfect User has imperfect knowledge of the synth and the interface, so chooses interactions similarly to the Perfect User, but 'misses' by a certain ammount: $\Delta P_i = \Delta P_i^0 * (1 + \varepsilon)$, where ΔP_i^0 is the Perfect parameter change, and ε is a zero mean gaussian random variable with variance σ^2.

A key consideration for the Traditional Interface is the order which parameters are varied. In the 'Perfect Order', parameters are visited in decreasing order of importance. In the 'Random Order', parameters are visited in a comploetely random order. Real users most likely operate somewhere in the middle of these regimes. (REFERENCE FOR THIS¿???)

5.4.2 Error Metric

A parameter based error metric is used for these tasks due to the complexity of using a sound based error metric (AS DESCRIBED IN SECTION).

For each parameter, p_i, a weighted L_1 norm is used. The parameters are weighted based on an approximate measure of importance, and normalised by their parameter ranges. The total error is calculated as:

$$E = \sum_{i=1}^{N} \|p_i - p_i^{goal}\|_1 * \frac{w_i}{r_i} \tag{5.3}$$

where w_i is the scalar importance weighting for the ith parameter, and r_i is the range of values that parameter i can take, see Table 3.1.

The L_1 norm was chosen over the L_2 norm so that a few large parameter errors don't dominate the error metric and as no gradients are necessary. This is by no means a perfect metric for comparins synth settings, but it is good enough for our purposes. (IS IT?)

5.4.3 Comparison of isolated interfaces

asdasdsad

5.4.4 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.

Talk about how the selections made in the blending interface could be used to create a large dataset of sound preferences, which could be used to train a more sophisticated method of preset generation in the blending interface.

Talk about the hierarchical nature of synthesis parameters could be exploited to improve preset generation and comparison algorithms.

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/.