International Journal on Artificial Intelligence Tools
Vol. XX, No. X (2006) 1–23

World Scientific
www.worldscientific.com

GENOPHONE: EVOLVING SOUNDS AND INTEGRAL PERFORMANCE PARAMETER MAPPINGS

JAMES MANDELIS

Creative Systems Lab, Informatics,
University of Sussex,
Brighton, United Kingdom
jamesm@cogs.susx.ac.uk

PHIL HUSBANDS

Informatics,
University of Sussex,
Brighton, United Kingdom
philh@cogs.susx.ac.uk

Received (Day Month Year)
Revised (Day Month Year)
Accepted (Day Month Year)

Abstract. This paper explores the application of evolutionary techniques to the design of novel sounds and their characteristics during performance. It is based on the "selective breeding" paradigm and as such dispensing with the need for detailed knowledge of the Sound Synthesis Techniques involved, in order to design sounds that are novel and of musical interest. This approach has been used successfully on several SSTs therefore validating it as an Adaptive Sound Meta-synthesis Technique. Additionally, mappings between the control and the parametric space are evolved as part of the sound setup. These mappings are used during performance.

Keywords: Adaptive; Hyper-instrument; Performance Mappings; Artificial Life.

1. Introduction

This paper describes the Genophone [9][10], a hyper instrument developed for sound synthesis and sound performance using the evolutionary paradigm of selective breeding as the driving process. Sound design on most current commercial systems relies heavily on an intimate knowledge of the SST (Sound Synthesis Technique) employed by the sound generator (hardware or software based). This intimate knowledge can only be achieved by investing long periods of time playing around with sounds and experimenting with how parameters change the nature of the sounds produced. Often it takes years to gain such experience. The system presented here attempts to aid the user in designing sounds and control mappings without the necessity for deep knowledge of the SSTs involved. This method of design is inspired by evolutionary techniques, where the user expresses how much particular sounds are liked and then uses these sounds to create new ones through variable mutation and genetic recombination. The aim of the system is

to encourage the creation of novel sounds and exploration rather than designing sounds that satisfy specific a priori criteria.

Through the use of "locked" parameter sets (whose values are unchangeable), variable control is exercised on the non-deterministic effect of the evolutionary processes, as illustrated in Fig.1. This feature, on the one hand, exercises some control over the shape evolution takes and on the other, allows a gradual familiarisation with the SST involved (if desired). Manual manipulation of the parameters for the particular SST is also provided, therefore allowing for precise control, if desired.

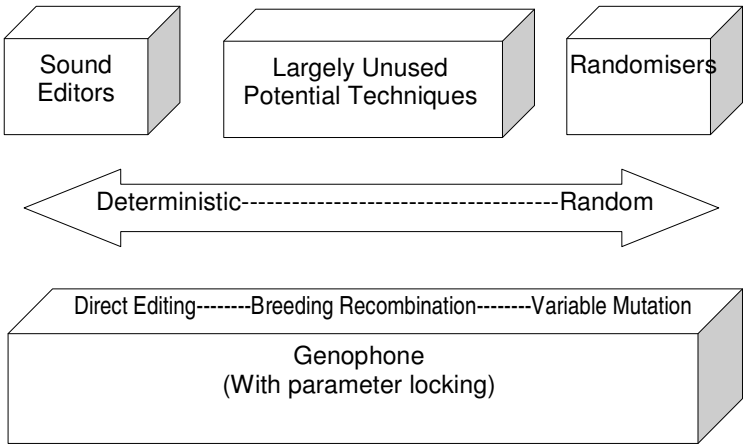

Fig. 1. Variable degrees of determinism in sound design

Genophone [9][10] is a "Hyperinstrument" or "Virtual Musical Instrument" [8][18][16][15], comprising of a dataglove, synthesiser and a PC that runs the evolutionary software. Real-time information from the glove is used to manipulate parameters that affect the sounds produced. The mapping of the finger flex and the sound changes is one-to-many. That is, a single finger flex (one of five) can control multiple parameters. This problem of mapping lower dimensionality performance controllers to higher dimensionality parameters [21][1][18] is also tackled within the same evolutionary framework. The resulting mappings are mainly used during performance by changing sound characteristics in real-time.

The selective breeding process generates System Exclusive MIDI messages (SysEx) for sound definitions, which are then sent to the synthesiser to be rendered. This level of abstraction facilitates the use of different external synthesisers with minimal effort. It also taps into the ability of commercial synthesisers to produce musical sounds by their design, and the existing wealth of sounds available for them. In previous attempts at using Artificial Life techniques for sound synthesis, a lower level definition was used,

therefore limiting the usage of the system to one particular piece of hardware (or software emulation) that employed only a single SST. Also, musical sounds tend to be much harder to evolve when lower level definitions are used [23][22].
This project has also shown that evolutionary methods can be used successfully on several sound-synthesis-techniques, demonstrating the feasibility of a generic approach . A number of users (about 20) with an interest in music and of various levels of musical skills and backgrounds interacted informally with the system after a brief introduction. Their general comments and feedback indicated that this approach is fast, often only a few generations are needed for evolving sounds that are interesting and of good quality. It is also very easy and fun to use, and easy to learn.

2. Related Work

There have been a number of other explorations of the use of evolutionary search and related techniques for sound generation. Most of these have approached the problem from a lower level than that used in the Genophone system described in this paper. Typically parameters of a particular single SST were directly manipulated in contrast to the approach taken here which operated at a higher level of abstraction – on System Exclusive MIDI messages -- facilitating the use of different external synthesisers and SSTs with minimal effort. Another important difference between this and previous work is that the Genophone system is concerned with the evolution of sounds and parameters mappings to allow their manipulation during performance, thus adding another dimension to evolutionary sound design.

One of the earliest related approaches was that of Johnson [5] who used a genetic algorithm to explore the sound space afforded by a granular synthesis technique where a sound event is built from a series of micro sound bursts (granules).. This paper articulated the possible advantages of using evolutionary search in sound design as a way of avoiding the need for deep knowledge of, and large amounts of experience with, the underlying synthesis techniques. An interactive genetic algorithm, in which fitnesses are assigned by a human, was successfully used to evolve sounds by manipulated parameters controlling the FOF granular sound synthesis algorithm [2].

A precursor of Johnson's work, and perhaps the first application of adaptive computing techniques to sound design, was Miranda's Chaosynth system, which dates from the early 1990s [14]. Here a cellular automaton was used to control some of the parameters in a granular synthesis technique. A cellular automaton is a grid of cells whose individual states change every cycle according to some rule that takes into account the values of all neighbouring cells. In the case of Chaosynth, values emanating from particular regions of the grid are used to control the frequency and duration values for the individual granules used to make up the sound.

Manzolli et al. [11] present a different low-level approach. Their Evolutionary Sound Synthesis Method (ESSynth) generates sequences of waveform variants by the application of genetic operators on an initial population of waveforms. An interactive fitness evaluation method is used to evolve new sounds by this direct manipulation of waveforms.

Recently, Dahlstedt has independently used an interactive evolutionary process, similar in outline to that employed in Genophone , to design sounds by manipulating the parameters of the underlying sound-generation engine [3]. This has been done in a generic way so that the system can be customised to operate with almost any hardware or software sound-generation engine. Dahlstedt points out that, as well as allowing musicians to design sounds without needing expert SST knowledge, evolutionary systems of this kind open up compositional possibilities based on "new kinds of structural relationships" which occur because "the sounds created during a breeding session are often audibly clearly interrelated" ([3]: 241).

Evolutionary systems have also been used for less exploratory kinds of sound design. For instance, Garcia [4] has developed methods to apply evolutionary search to the design of sound synthesis algorithms and demonstrated the efficacy of his approach by evolving various target sounds, including notes played on a piano, through the use of an automatic fitness function that measured how close the generated sound is to the target sound. Recently McDermott et al. [5] pursued related work using genetic algorithms operating on the parameter settings of an FM synthesizer, with the aim of mimicking known synthesized sounds. They compared a number of different automatic fitness measurement techniques to achieve this end.

Jon McCormack's Eden is an interesting example of a related application of evolutionary techniques in an art installation context [12]. In this system, agents populate an artificial world in which they can move around and make and hear sounds. These sonic agents must compete for limited resources in their environment, which is directly influenced by the artwork's audience. The agents generate sounds to attract mates and, because of the influence of the audience on the virtual environment, particularly the growth rate of virtual food, to attract the attention of the audience. In this work McCormack has demonstrated the successful use of an open-ended automatic evolutionary process to generate a highly engaging interactive artwork. This system illustrates a more implicit approach to fitness evaluation, with a fairly oblique interaction element. It would be interesting to explore similar direction in work more explicitly aimed at sound design.

3. System Description (Genophone)

3.1. *Hardware*

At its inception it was decided to construct the system in such a way that would require minimal modifications and cheap off the shelf components, as it was not clear whether the results would be positive. Also several labour shortcuts where taken. One of them was the choice of a particular synthesiser; at the time it was the only virtual synthesiser that supported several synthesis techniques. Rather than using several hardware or software sound engines, each with its implementation peculiarities, this multiple synthesis single engine was used instead. Also its ability to use performance knobs with assignable parameters was also a major issue of choice. The hijacking of the performance knobs by the home-made glove via a resistance-to-voltage conversion board saved the need for external DACs. That also allowed for performance without necessitating the need of a computer having to be attached.

The main hardware components are:

- A KORG Prophecy solo-synthesiser, which produces the actual sound.
- A PC with a MIDI interface, used for running the evolutionary software.
- A glove with five flex-sensors for each finger. It is mainly used as a performance orientated device.
- An interface board for signal conditioning. It translates the finger flex of the glove (expressed as variable resistance) to five voltage signals 0-5V appropriate for internal use on this specific synthesiser.

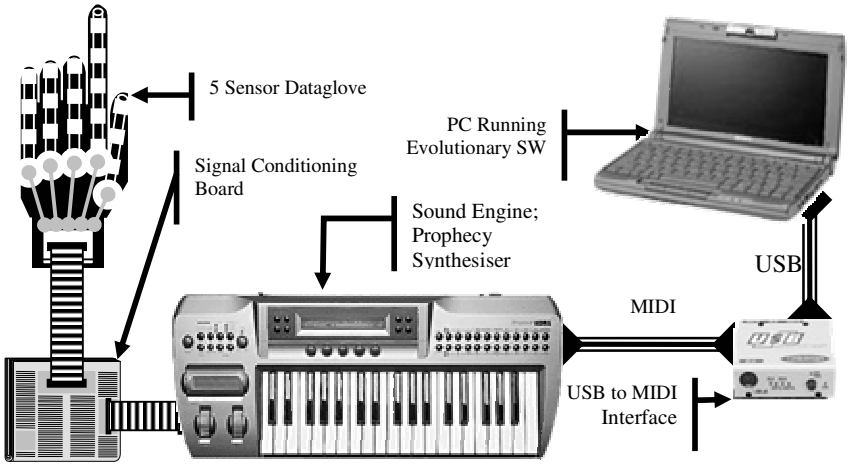

Fig. 2. Genophone; System Setup [9][10]

3.1.1. *Connectivity*

The PC is connected to the Synthesizer via MIDI. If the PC is equipped with a MIDI interface then it can be connected directly to the synthesizer. In this case, a USB to MIDI interface has been used with the specific notebook. The MIDI connection is only necessary when new sound definitions need to be exchanged between these two units. After that, the Synthesizer can be used for performance without the use of the PC connection.

The glove is connected to the resistance-to-voltage converter board via a 6-wire cable. One +5V and five returns.

The converter board is connected directly to the synthesizer's five input knobs (performance knobs), allowing finger flex positions to take the place of knob rotational positions.

3.1.2. *Resistance-To-Voltage Converter*

The resistance-to-voltage converter board is constructed on a strip-board, it has eight operational amplifiers (only five are used), in the form of two National Semiconductor Rail-to-Rail 4 x op-amp ICs (LMC6484). There are also capacitors for noise filtering and several resistors. The following schematic describes half the board (only 4 op-amps, one

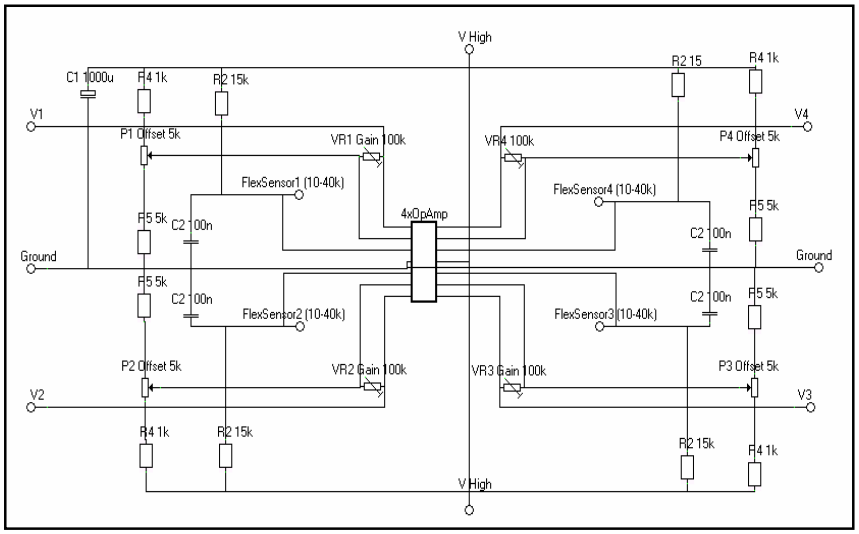

Fig. 3. Resistance-To-Voltage Converter schematic

IC), the rest of the board is duplicated. Terminals; V1 to V5, V High and Ground are connected to the synthesisers internal PCB board control knob (performance knobs) connections. Terminals; Flex Sensor 1 to 5, V High and Ground are connected to the

glove's flex sensors. This way the build-in control knobs are being highjacked by the glove, dispensing the need for external DA converters and MIDI encoders.

3.1.3. *Synthesiser*

The synthesiser used at this stage of the project is a KORG Prophecy solo-synthesiser. This particular synthesiser has been chosen because it supports seven different sound synthesis techniques. The available multitude of synthesis techniques is used to show the suitability of this type of interface to a variety of sound synthesis paradigms. The sounds are modelled by 200+ parameters and since the synthesiser is geared towards performance, its internal architecture allows for real-time manipulation of those parameters, via five "performance knobs". The knobs can be assigned to control up to four parameters each. Also the way those parameters are changed by the knob's rotation can be specified i.e. the lowest and highest value and the change function (linear, exponential, logarithmic).

The main relevant features of the Prophecy are:

- Seven different synthesis techniques are supported, ranging from analog synthesiser oscillators to physical models such as sax or bass guitar.
- The Oscillator block provides seven types of oscillators, such as Analog, VPM, and Physical Modelling, and also contains a Sub Oscillator and Noise Generator. The Wave Shape block can be set to either Clip or Resonant wave shaping, and determines how the waveform is shaped and the balance at which it is mixed with the original waveform. The Mixer Block determines the levels at which the two systems of Oscillator, Sub Oscillator, Noise Generator, and Feedback are sent to the Filter block. The Filter block provides two multi-mode filters (switchable between LPF/HPF/BPF/BRF), and can be placed in either series or parallel to control the output. The Amp block lets you independently control the level of each output signal. The Effect block provides seven types of effect; Distortion, Wah, Chorus/Flanger+Delay, Reverb, and Dual Parametric EQ. (You may select Chorus/Flanger+Delay or Reverb.)
- The Performance Editor function lets you assign parameters to each of the five knobs for real-time control. Four Performance Editor sets are provided. For each set, any of the more than 200 program parameters can be assigned to a knob, meaning that up to 4 parameters can be assigned for control by one knob. Performance Editor Settings are also stored independently for each sound and are part of its SysX description.

3.2. *Software Description*

3.2.1. *Overview*

There are three main structures used in the software system.

The *Instrument Template* describes the parameters used in making up a patch in a particular instrument (in this case the Prophecy). It defines things like SysEx headers, parameter names, type and range of values. It also defines sections of parameters (and sections of sections) that reflect the logical design of the instrument. The state of

parameters “Locked” or “Unlocked” is also stored here temporarily as well as the current parameter values (not persistent).
The *Values (Genotype)* is a particular set of values for the parameters defined above, i.e. of the required type in the required range. Each Genotype translates to a SysEx message that defines a sound.

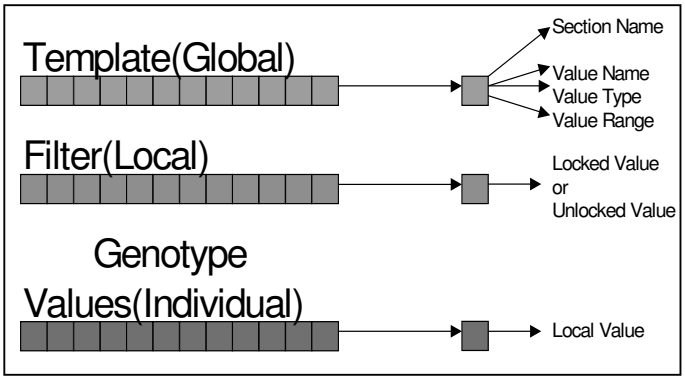

Fig. 4. Data Structures

The Filter is a list of “Unlocked” parameters. When a filter is applied to the current patch it has the effect of “Unlocking” any “Locked” parameters.
The state of parameters (“Locked” or “Unlocked”) is used for limiting the effects of

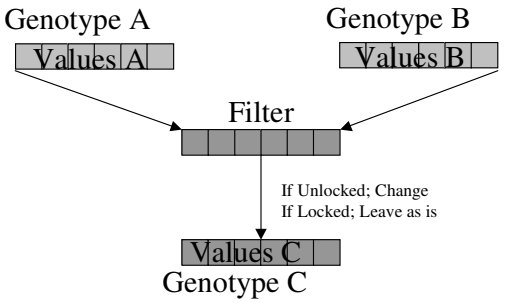

Fig. 5. Filter Action

loading new patches, mutation, and recombination. Filters are used to fill the gap between total knowledge of the SST (knowing how each parameter changes the sound) and total ignorance of the principles underlying the sound production. By enabling or disabling

sets of parameters it is possible to influence just parts of the instrument's logical structure, i.e. the effects, mixer or oscillator sections, etc.
The software written for this project is a multiple-document interface (MDI) application. It contains several tools/components for manipulating and viewing sound parameters. The application makes extensive use of drag-and-drop functionality for most operations.

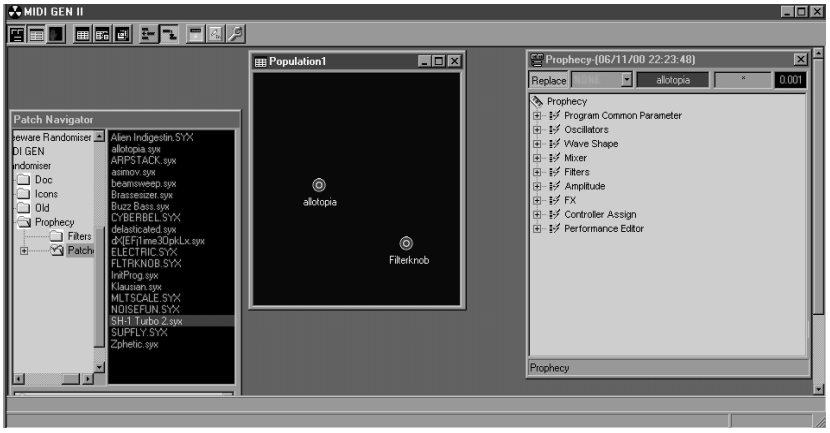

Fig. 6. Application Screenshot

A description of the application components follows.

3.2.2. *List View*

The first component is List View; its main function is to display the current patch's parameter values in a treeview form, lock/unlock parameters and loaded filters.

The patch files can be dragged-and-dropped into this component, which will display the new parameters, upload the patch to the synthesiser (optional) and play a note (optional) in order to preview it. The parameters are aggregated into sections (and sections of sections) that reflect the logical structure of the synthesiser.
Parameters and sections can be in two states, "Locked" and "Unlocked", these states can be applied recursively on sections. When a parameter is "Unlocked" its value can be changed freely, when "Locked" its value is frozen. Their icon in the List View indicates the state of parameters, and sections.

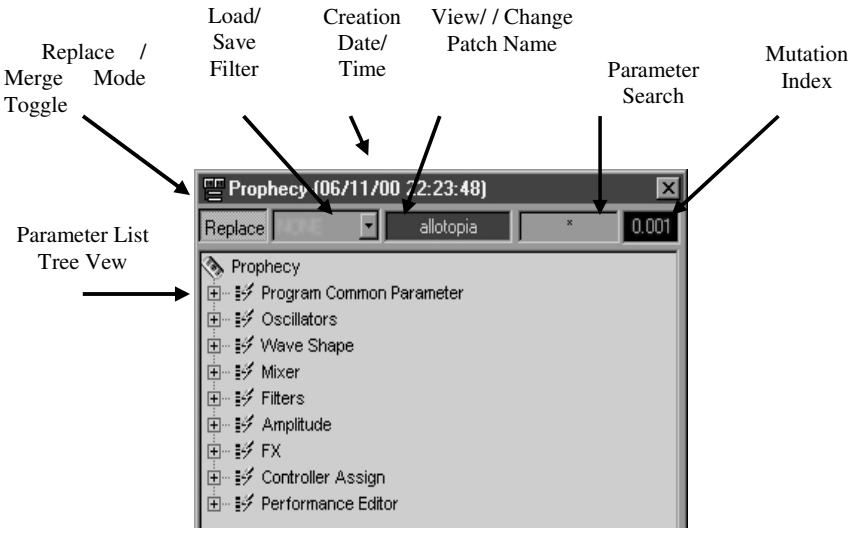

Fig. 7. Listview Tool

A collection of "Unlocked" parameters (only their state not their value) is called a "Filter". Filters allow parts of the patch to be frozen and parts to change freely, filters can be saved and loaded. Each time a Lock or Unlock operation takes place a new filter is created, which can be saved for later use.

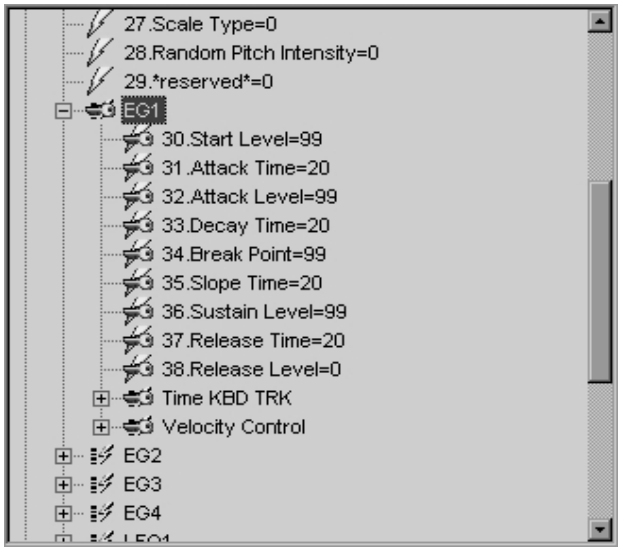

Fig. 8. Parameters in Listview Tool

The "Replace" option button, when pressed it changes to "Merge" and back. When a new patch is loaded the default behaviour ("Replace" mode) is to load the new patch afresh ignoring any current loaded filter and overwriting any existing values. In "Merge" mode, the values of the existing patch are overplayed with the values of the new patch, but only unlocked parameters defined by the current filter, the rest remain as they were.

Fig. 9. "Replace" option button

Also in List View it is possible to use a context menu (right click) for extra functions;

Lock – Locks/Unlocks (recursively) parameters and sections.
SysX –>
Load – Load a new patch from file.
Save – Save current patch in a file.
Download – Download patch from synthesiser and make it current.
Upload – Upload current patch to synthesiser.
MIDISetup – Enter MIDI set-up dialog.
Undo – Undo previous change, send or patch upload.
Randomise – Change the parameter(s) value to a random one between the Minimum & Maximum values.
Send – Upload this parameter's value.
Edit – Edit a parameter's value.
Auto Send – (Option) Send individual parameter changes as they happen.
Auto Upload – (Option) Upload new patches (whole patch), after change or load.
Auto Play Note – (Option) Play a note after the uploading a new patch.
Mutate – Mutate the parameter(s) value.

4. The Evolutionary System

The evolutionary algorithm used is interactive and relies on the user to assign relative fitnesses to the individuals in the population and to select which ones are to be used to create the next generation. This is facilitated by the use of a specially designed population window.

4.1.1. *Population Window and user defined relative fitness*

Population windows are containers for groups of patches and they support the following functionality.

- Files can be dropped into them from the Patch Navigator, List View and other Population windows.
- Pressing Delete, will remove individuals from the population.
- Double-clicking (or pressing Enter) on a selected individual loads it to the List View in "Replace" mode (which is then loaded to the synthesiser if Auto Upload is on).
- Dragging and dropping an individual from a Population to List View will take account of the filter mode, if in "Merge" mode, the current filter will be used to load the new values, so only "Unlocked" parameters will be changed.
- Multiple individuals can be selected either by Shift-Click or using the Lasso, as a group they can be deleted, dragged & dropped, or used as parents for reproduction.

For each individual in the population, its fitness value is derived from its relative height within the window (vertical axis position) which is decided by the user. In this way the user can show relative preferences for patches by positioning the more liked ones closer to the top. From this vertical placement the fitness of each individual is derived. This fitness is used by the recombination operators as a bias (see operator descriptions in section 4.1.3.). It is also used by the recombination operators for assigning to each offspring a temporary estimated fitness, which is derived from its parent fitness's weighted by the Euclidian distance of its parameters from each parent. After previewing the sounds, the fitness should be changed later by the user, by simple rearrangement of individuals within the new population window. The temporary fitness merely gives newly created individuals an initial position in the widow.

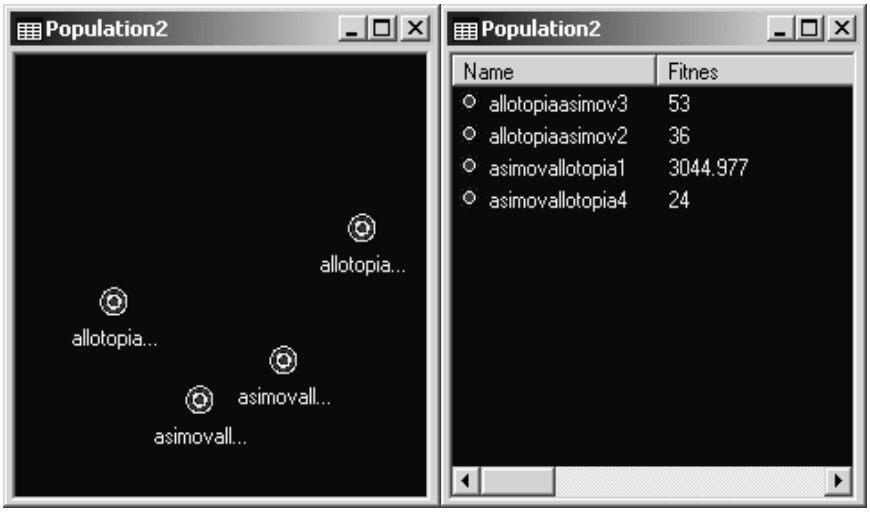

Fig. 10. Population Window Icon View and List View

Population windows can have two views. Double-Clicking on the background cycles through the views; Icon view where individual can be placed at different heights, and list view where can be sorted in a list.

4.1.2. *Mutation*

The variable mutation operation creates a new population of ten mutants based on a previously selected individual. The amount by which the parameters are mutated from their original value is governed by the *"Mutation Factor"*. It can have a value between 0 and 1, when 0 the new value is the same as the old, when 1 the new value can be anything within the valid range of values for that particular parameter. For instance; if mutation of factor 0.5 is applied to a parameter that can take values between 0 and 100 and with a current value of 50, its new value will be a randomly chosen one between 25 and 75.
Only the "Unlocked" parameters are mutated, "Locked" parameters just get copied across from the original patch. Each mutant parameter value is mutated according to the mutation factor.

4.1.3. *Reproduction*

When a number of individuals are selected, the "Reproduce" button creates a new population made up of two offspring from each possible combination of 2 parents.

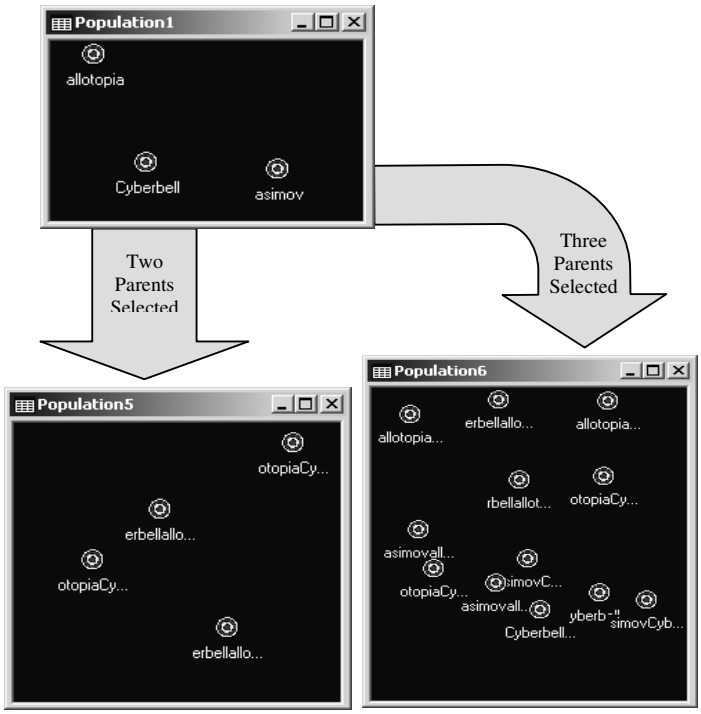

Fig. 11. Generation of Populations via Reproduction

Reproduction Recombination Types

Any number of individuals in a population can be highlighted as parents for operators to be applied (all possible couples will be used for recombination operators or just one for variable mutation). For each couple of selected individuals, a random crossover point is selected in the range of 1 to the number of *"Unlocked variables"* and an other at a wrapped offset derived by the number of *Unlocked Variables* x *"Depth Factor"*(0<,<1). Originally two offspring are produced, each a clone of a parent, and then their unlocked parameters are overwritten, but only those defined within the wrapped range between the two crossover points. As previously mentioned, each offspring is assigned a temporary "estimated" fitness derived from its parent's fitness's weighted by the number of parameters inherited from each parent.

Three possible recombination operators are available for the user to apply.

Swap Recombination

A random crossover point is selected in the range of 1 to the number of "Unlocked variables". The offspring parameters are overwritten by the other parent's, but only those defined within the range of 1 to the Crossover point for one, and from the Crossover point to the number of "Unlocked variables", for the other. Which parent is used first, in this parameter copying process, depends on whether the Crossover point is before or after the middle of the unlocked variables range and by which parent has the larger fitness.

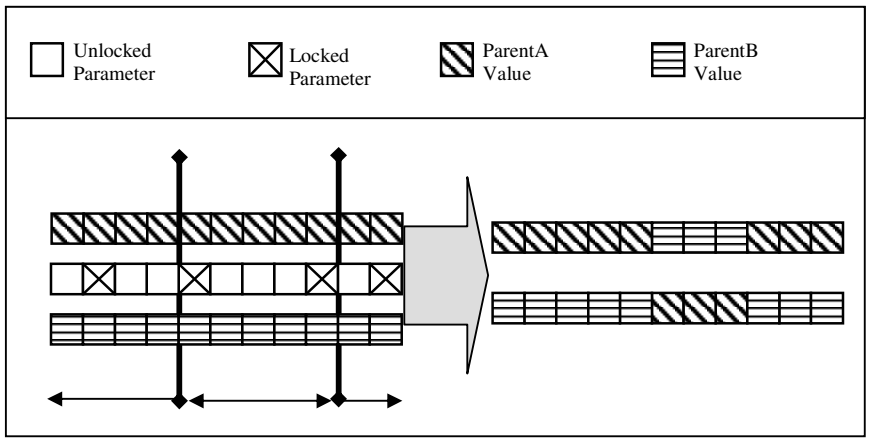

Fig. 12. Swap Recombination Action

Probabilistic Recombination

Two random crossover points are selected in the range of 1 to the number of "Unlocked variables". Only the parameters between the crossover points are overridden. The parameter values used for overwriting are selected probabilistically from the parents, where the probability is proportional to the parent's fitness.

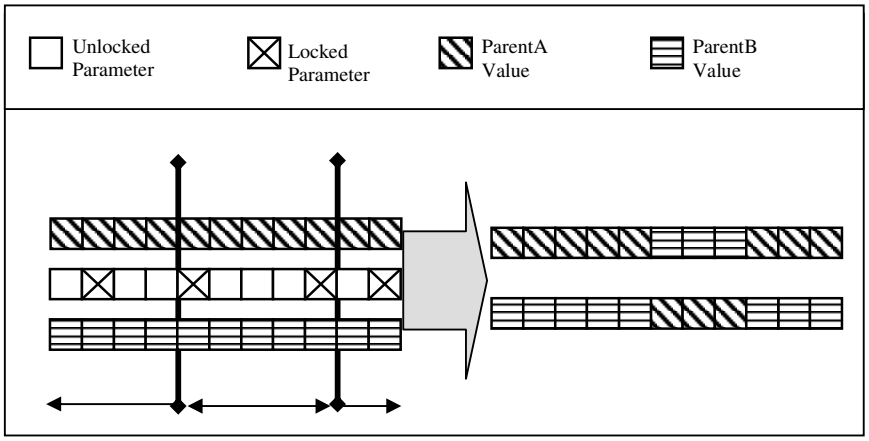

Fig. 13. Probabilistic Recombination Action

Interpolating Recombination

Two random crossover points are selected in the range of 1 to the number of "Unlocked variables". Only the parameters between the crossover points are overridden. The parameter values used for overwriting are derived as in-between the parent values weighted by their fitness. Consequently, in the two offspring, the region between the Crossover points is identical; the rest of the genotype is identical to one parent for one offspring and identical to the other parent for the other.

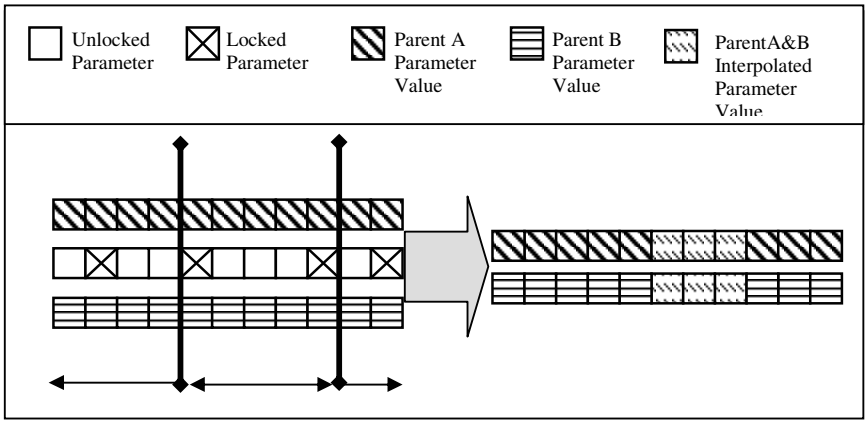

Fig. 14. Interpolating Recombination Action

4.1.4. *Discussion of Reproduction Operators*

A mix-and-match approach can be used for the operators; they can be applied at any stage and in any order. Also at any stage, new parents can be brought into the gene-pool, removed from it, or spawn new populations for seeding. Multiple strains can be evolved and maintained, and also used for speciation or backtracking from evolutionary dead-ends. From experimentation the following observations were made:

- *Variable Mutation* is quite effective when using a sensible *"Mutation Factor"*; this parameter has to be found by rule-of-thumb for each particular SST as well as the number of unlocked variables. It is a very subjective choice. *"Mutation Factor"* values usually range from 0.01 to 0.9.
- *Swap Recombination* is the one used most, it preserves clusters of parameters and their values, and its results are slightly more predictable. This is due to the high *epistatic* correlation of parameter genotype locus and parameter function.
- *Probabilistic Recombination* seemed more likely to disrupt clusters of related parameters, resulting to sounds more likely to be unfit (no sound produced at all, or was too alien from the parents to be usable), sometimes this could be considered to be an advantage since the resulting sounds though alien were quite attractive. One could say that results from this recombination type are more radical.
- *Interpolating Recombination* by using "in between" values quite often the sounds were indistinct, quiet or just silent. One could say that this recombination type is too conservative or uncommitted in its results.

5. Genophone and Virtual Musical Instruments

The system can be viewed as a Virtual Musical Instrument and as such can be considered as a new step in the development of musical instruments. Mulder has suggested the following classification and development of musical instruments [15] (Figs. 15-17):
The first step, according to Mulder, is traditional acoustic instruments that are manipulated in a certain way in order to produce the sounds. The next development is the use of electronics in order to apply sound effects on acoustic instruments. The manipulations remain essentially the same. His comments on the characteristics of these types of instruments are: "Limited timbral control, gesture set and user adaptivity. Sound source is located at gesture."[15]

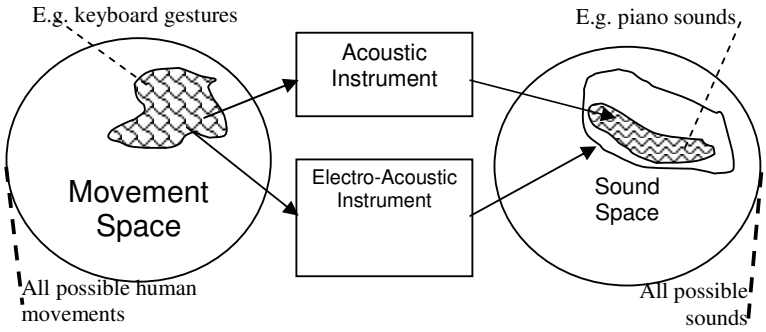

Fig. 15. Steps 1 & 2 of instrument development [15]

The next step, suggested by Mulder, are Electronic Musical Instruments, where the essential manipulations of a piano produce sounds that mimic other acoustic or electronic instruments. His comments on the characteristics of these types of instruments are: "Expanded timbral control, though hardly accessible in real-time and discretized; gesture set adaptivity still limited. Sound emission can be displaced."[15]

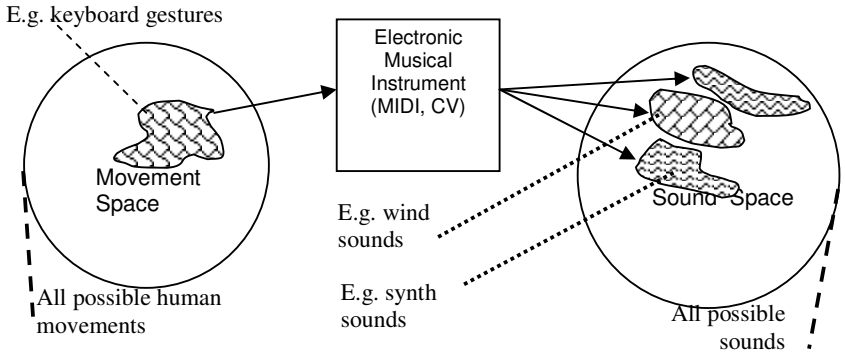

Fig. 16. Step 3 of instrument development [15]

The next step, suggested by Mulder, involves Virtual Musical Instruments where gestures from motion caption devices are used to drive sound engines. His comments on the characteristics of these types of instruments are: "Expanded real-time, continuous timbral control; gesture-set user designed via breeding. Any gestures or movements can be mapped to any class of sounds."[15]

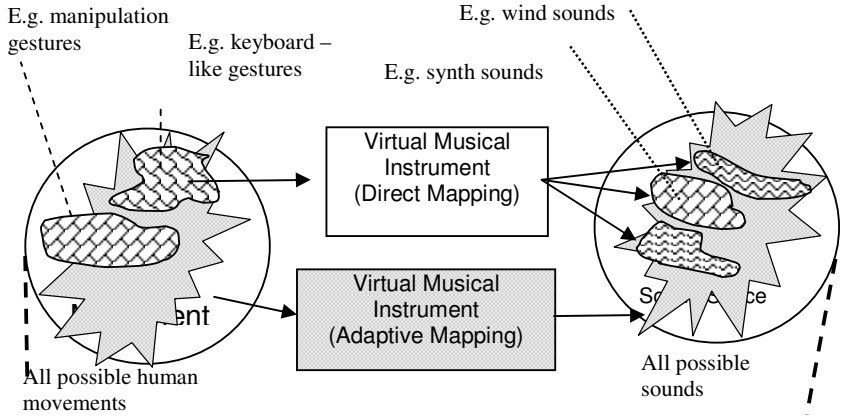

Fig. 17. Steps 4 [15] & 5 [9][10]of instrument development

As an improvement to the last step, and an extension to the overall classification, we suggest a new class. It involves VMIs that provide a framework for adaptive generation of sounds and their gesture mappings. Genophone [9][10] belongs to this new class of Adaptive VMIs and exhibits the following characteristics: Expanded real-time, continuous timbral control; gesture-set is user designed via breeding. Any gestures or movements can be mapped to any class of sounds where both the mappings and the sounds are subject to the same evolutionary forces applied by the user.

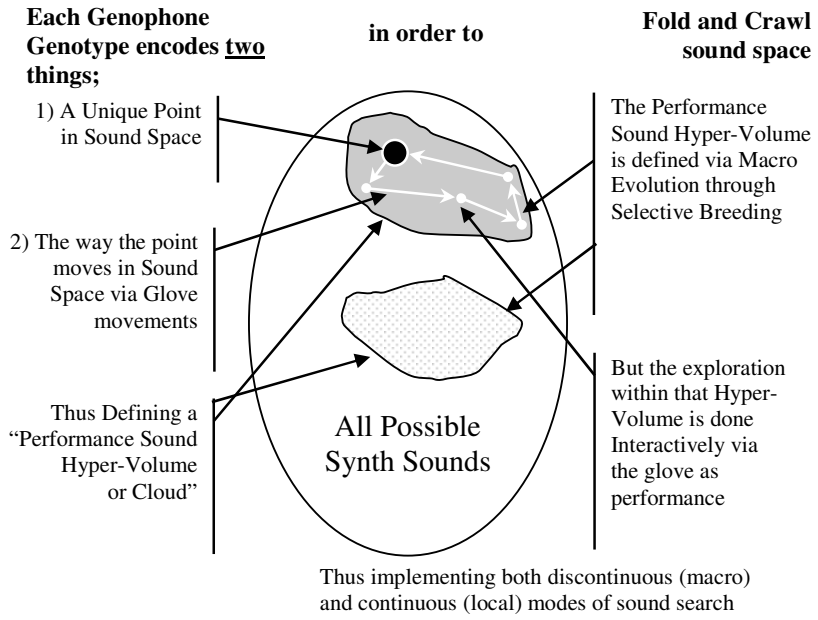

Fig. 18. Genophone operation

6. Experiences with Genophone

The preliminary results from this project are encouraging and will be followed by system enhancements that will allow more complex experiments to be performed and move into the next phase. Most of the initial aims for this pilot phase have been satisfied and are summarized below. Samples of experiments performed and a PowerPoint presentation can be found at; http://www.informatics.sussex.ac.uk/users/jamesm/Genophone/

6.1. *Usage Modes*

The system has been used in three distinct modes of operation:

(a) As a Solo Instrument; where the right hand plays a melody on the keyboard while the left hand is changing the sound via the glove. For those lacking keyboard skills, results were much better when a sequencer was used to play familiar melodies via MIDI, while at the same time the sound was manipulated with the glove. In this way it is possible take a "sterile" sounding MIDI file and breath a lot of life into it.

(b) As a Single Event / Sound Effect Generator: a single note is played either as a continuous drone or until the sound expires. The glove is not often used in this mode with the exception of drones. The Sound Effect Generator mode was generally easier to use, and produced a lot of single event sounds that were very rich and dynamic. Quite often they did not play well musically on the keyboard when attempts were made to build phrases from sequences of them, but had enough structure and complexity to be satisfying in themselves, often providing their own melodic or rhythmic framework. Drone sounds where also produced in this way, sometimes using the glove to change their characteristics.

(c) As a Pattern Arpeggiator; where an arpeggiated pattern is chorded with the right hand, while the left hand is changing the sound via the glove. Many users found the Pattern Arpeggiator mode the most fun to use; it has an instant appeal due to the responsiveness of the glove and rhythmic structures can be created in a very intuitive way. Also the repetition of the phrase facilitates the perception and prediction of the sound changes within a rhythmic framework.

It is obvious from the above that there is an appropriate mode of operation for creating separate parts of a musical track, whether these parts are rhythmic, melodic, drones or single events.

6.2. *Hand Rearing vs. Hand Design*

The ease of use of the interface was a surprising outcome. The interactive selective breeding paradigm is an accessible one and users were able to breed complex sounds after only a brief introduction. The sounds produced were of such a quality that it would take someone with quite a bit of experience in the SST involved if they were to be programmed manually, which would be much slower. The overall process is exploratory rather than goal orientated; it is not designed to satisfy a priori sound specifications, i.e. "I would like to produce a bell sound". It does not preclude the possibility of doing so in indirect ways, though. For instance, if bell or bell-like sounds are used for seeding the initial population, then is conceivable that a satisfactory bell sound will be produced within a few generations of selective breeding and variable mutation.

In a sense the system is a tool for facilitating the exploration of novel sounds and performances rather than the design of specific sounds. The design of specific sounds is supported by the onboard operation of the synthesiser as well as third party software editors such as Progenie. By contrast Genophone was designed to harness and facilitate the use of genetic recombination which in biological systems is a creative process in itself. A biological analogy would be the breeding of animals or plants which humans have employed for millennia. When pigeons are bread, for example, it is not normal (at

least not yet) to employ gene level manipulations via genetic engineering. Instead, macrocosmic manipulations such as artificial insemination or pair choices are enough to manipulate the genome as a whole and consequently the resulting offspring. It is self evident that such macrocosmic manipulations are far easier than advanced genetic engineering which requires a very large amount of specific technical knowledge, one that we have just started getting to grips with. If specific changes are required from the organism, then genetic engineering can conceivably implement these changes, albeit with the associated costs in effort. In exactly the same manner Genophone provides the same macroevolutionary manipulations that are employed in organic breeding. In addition, via the glove manipulations, it provides a local direct and interactive exploration that facilitates smaller changes when used as a performance tool. A biological analogue of this last facility and way of manipulation does not exist but it is analogous to dogs breeding where not only the appearance (the starting sound) but also the temperament and the character (glove manipulations) of the dog are bread and manipulated. The option of locking whole sections, groups or individual parameters / genes, provides an added layer of control on the evolutionary process that can conceivably bridge the gap between a totally free-form evolution and the tight control offered by an editor. The inspiration for parameter locking came from the way genes are activated and deactivated in biological genomes, producing epigenetic evolutionary effects.

6.3. *Meta-SST*

Different SSTs can be used without the use of Specific Domain Knowledge. It was an initial requirement that no specific domain knowledge should be used in the system. That is, the parameters are treated as going into a black box, no knowledge of their function is kept in the system. As a result a new SST can be added by just specifying the System Exclusive Implementation Chart of the new synthesiser. As a down side, when sounds are produced that are interesting but have a serious flaw, e.g. they are very quiet, then the there is no evolutionary way in the current framework to address the problem. The only solution is to either selectively breed part of the genotype that is suspected of being responsible, or manually tweak individual values until the desired result is achieved.

6.4. *Recombination vs. Mutation*

Experiments with Genophone have shown that the Evolutionary Paradigm can be successfully applied to the creation of novel sounds, often of surprising complexity. It seems that viable (fit) parameter sections are preserved through the genetic recombination, as it is also the case with Genetic Algorithm optimisation. In other words, if the initial sounds used to seed the initial population are professionally designed ones, then the offspring are likely to be of comparable quality. This is also shown by the observation that genetic recombination produces higher quality results than if mutation is used alone. In implementations [23] where no genetic recombination is used, and mutation or a type of “genetic space crawling” is used instead, it is much harder to produce sounds that are complex and of high (subjective) quality.

6.5. *MIDI SysEx & Musicality*

The level of abstraction used (SysEx MIDI) was vindicated. Synthesisers are designed for musical sounds and aspects such as keyboard mapping are already implemented. If an approach had been taken where the parameters were encoding low-level sound production [22], it would have been much harder to produce musically acceptable sounds.

6.6. *Expressivity & Mapping*

Experience with the glove has proved it to be very responsive and expressive. The original professionally programmed sounds that are used for seeding contain mappings that are relevant to the parameters used by the particular patch. These mappings seemed to be preserved and combined better by the *Probabilistic Recombination* operator (due to the high epistatic correlation). Although there are no widely accepted a priori mappings between hand movements and sound changes (since they are also evolved), they are usually easily internalised by the player [7]. That is, after a few finger flexions the brain seems to be able to assimilate the correspondence of movements and sound changes. This was further facilitated by using arpeggiated phrases, it seems that phrases are processed better by short term memory in order for the brain to notice the sound changes and play with the rhythm. The easy internalisation of the mapping actually came as a surprise; it was thought originally, that some unchanging, directly programmed mapping would have to be used, in order for the brain to learn the movement-to-sound mapping. One example of such mapping is "sound sculpturing" [17] in which the sound is represented by a three-dimensional object where changes in its shape are translated to changes in sound. It was also thought that some kind of structured language would have to be used for describing those (evolved or not) mappings [6][8][18][16][15][20] if no direct one-to-one mapping was used. The use of such formalisation [23] has not been necessary yet, partly because the Prophecy implements its own one-to-many (1 to 4) mapping formalisation through SysEx parameters, and partly because the relatively low dimensionality of the input device (dataglove) which has only five degrees of freedom.

7. Future Directions

It would be interesting to see if the ease of internalising mappings is retained when input devices of more channels are used i.e. more than five. In the future, when different synthesisers and input devices (with more degrees of freedom) are used, the issue of a mapping formalisation will have to be readdressed. Also the two processes for *sound* evolution and *motion-to-sound-mapping* evolution will have to be separated from the same genotype. More operators are currently being developed and tested. Since this is an exploration system each operator has unique properties that can be used appropriately to guide the search.

Acknowledgments

Special thanks to the late Andrew Gartland,-Jones also to Jon MacCormak, and Sam Woolf for their incisive feedback and stimulating discussions; to Prof. Margaret Boden for inspiring JM on computers and creativity during his most impressionable years.

References

1. Choi, I., Bargar R., Goudeseune C,. "A manifold interface for a high dimensional control interface." Proceedings of the ICM conference (Banff, Canada), 385-392. San Francisco CA, USA: International Computer Music Association, 1995.
2. Clarke, J. (1992) An FOF synthesis tutorial, *Appendix 4 of the CSound manual*. Also at http://mitpress.mit.edu/e-books/csound/fpage/tut/FOF/FOF.html
3. Dahlstedt, P. (2001) "Creating and exploring huge parameter spaces: interactive evolution as a tool for sound generation". In *Proceedings of the 2001 International Computer Music Conference*, pp. 235–242. Havana, Cuba: ICMA
4. Garcia, R. (2001) "Growing sound synthesizers using evolutionary methods". In *Proceedings ALMMA 2001: Artificial Life Models for Musical Applications Workshop, (ECAL 2001)*, ed. Eleonara Bilotta, Eduardo R. Miranda, Pietro Pantano and Peter Todd. Consenza, Italy: Editoriale Bios.
5. Johnson, C. (1999) "Exploring the sound-space of synthesis algorithms using interactive genetic algorithms". In *Proceedings of AISB'99 Symposium on AI and Musical Creativity*, ed. Angelo Patrizio, Geraint Wiggins and Helen Pain, pp. 20–27. Brighton, UK: AISB.
6. Keane, D. & Gross, P., "The MIDI baton," Proceedings International Computer Music Conference, Columbus, Ohio, USA. San Francisco CA, USA: International Computer Music Association, 1989.
7. Krefeld, V., "The Hand in the Web: An interview with Michel Waisvisz," *Computer Music Journal*, 14 (2), pp. 28-33, 1990.
8. Machover, T. & Chung, J., "Hyperinstruments: Musically intelligent and interactive performance and creativity systems," Proceedings International Computer Music Conference, Columbus, Ohio, USA. San Fransisco CA, USA: International Computer Music Association, 1989.
9. Mandelis, J., "Genophone: An Evolutionary Approach to Sound Synthesis and Performance," Proceedings ALMMA 2001: Artificial Life Models for Musical Applications Workshop, Prague, Czech Republic: Editoriale Bios, pp. 37-50, 2001. www.informatics.susx.ac.uk/users/jamesm/Papers/ECAL(2001)ALMMAMandelis.ps
10. Mandelis, J., "Adaptive Hyperinstruments: Applying Evolutionary Techniques to Sound Synthesis and Performance," Proceedings NIME 2002: New Interfaces for Musical Expression, Dublin, Ireland, pp. 192-193, 2002. www.informatics.susx.ac.uk/users/jamesm/Papers/NIME(2002)Mandelis.pdf
11. Manzolli, J., Maia, A., Fornari, J., Damiani, F. (2001) , The evolutionary sound synthesis method , Proceedings of the ninth ACM international conference on Multimedia, 585-587, ACM Press New York
12. McCormack, J. (2003) "Evolving sonic ecosystems." *Kybernetes: The International Journal of Systems & Cybernetics* **32**(1/2), 184–202.
13. James McDermott and Niall J. L. Griffith and Michael O'Neill (2005), Toward User-Directed Evolution of Sound Synthesis Parameters, In F. Rothlauf et al. [19], 517-526, Springer.
14. Miranda, E. R. (1995b) "Granular synthesis of sound by means of cellular automata". *Leonardo* **28**(4), 297–300.

15. Mulder, A.G.E., "Virtual Musical Instruments: Accessing the Sound Synthesis Universe as a Performer," Proceedings of the First Brazilian Symposium on Computer Music, pp. 243-250, 1994.
16. Mulder, A.G.E. Fels, S.S. & Mase, K., "Empty-handed Gesture Analysis in Max/FTS," Proceedings of the AIMI international workshop on Kansei - the technology of emotion, Antonio Camurri (ed.), pp 87-90, 1997.
17. Mulder, A.G.E. Fels, S.S. & Mase, K., "Mapping virtual object manipulation to sound variation," IPSJ SIG notes Vol. 97, No. 122, 97-MUS-23 (USA/Japan intercollege computer music festival), pp. 63-68, 1997.
18. Pressing, J., "Cybernetic issues in interactive performance systems," *Computer Music Journal*, 14 (1), pp. 12-25, 1990.
19. Franz Rothlauf et al. (Eds) (2005), Applications of Evolutionary Computing, Proc. EvoWorkshops 2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, and EvoSTOC, LNCS vol. **3449**, Springer.
20. Rovan, J.B. Wanderley, M.M. Dubnov, S. & Depalle, P., "Instrumental Gestural Mapping Strategies as Expressivity Determinants in Computer Music Performance," presented at "Kansei - The Technology of Emotion" workshop, 1997.
21. Wessel, D. and Wright, M. "Problems and Prospects for Intimate Musical Control of Computers," ACM SIGCHI, CHI '01 Workshop New Interfaces for Musical Expression (NIME'01) ,2000.
22. Woolf, S., "Sound Gallery: An Interactive Artificial Life Artwork," MSc Thesis, School of Cognitive and Computing Sciences, University of Sussex, UK, 1999.
23. Yee-King, M. (2000).,"AudioServe - an online system to evolve modular audio synthesis circuits," *MSc Thesis,* School of Cognitive and Computing Sciences, University of Sussex, UK, 2000.