LITTER MUTATORS
MOVING MORPHOLOGICAL MUTATION INSIDE TIME

Peter Castine
Compositeur et informaticien indépendant
p@castine.de

ABSTRACT

The notion of Morphological Mutations was introduced by Larry Polansky, gaining wider popularity with the implementation of Spectral Mutation in Tom Erbe's *SoundHack*. However, until now all implementations were outside time.[1] Furthermore, the theoretical framework used to date prevented inside time calculation of certain mutation processes. This is not a matter of CPU demands but of mathematics. This paper discusses an implementation of mutation and the modifications to the theory that were necessary to handle mutation processes interactively inside time. A raft of further extensions to existing mutation techniques were developed. Experiences in the development process are documented and new applications are investigated.

The implementation presented here runs as a set of external objects for the Max/MSP platform, now running under both Mac OS and Windows XP.

1. INTRODUCTION

The notion of morphological mutations was introduced by Polansky in a series of papers beginning in the late 80's [4][5][7]. In its simplest form, mutation can be understood as a kind of cross-fade between two musical structures. More complex mutations extend this notion dramatically, reaping results ranging from the wonderful to the bizarre. Originally applied at the level macro-events (pitches, intervals, harmonies, durations, etc.). The ideas were soon extended to the micro-events: direct processing of audio signals at the sample level.

Previous implementations process mutations exclusively outside time [6][7][8]. This paper describes a new set of external objects for Max/MSP implementing mutation inside time. These objects provide a unified set of tools with greater flexibility than their forerunners. Additionally, interactive experiments in novel configurations as well as time-variant handling of mutation parameters not possible in any other implementation can now be conveniently built.

To perform mutations inside time it was necessary to develop a new method for controlling "clumping", an important step in certain stochastic mutation algorithms. This is discussed in section 5. A new member of the mutation family called "Weighted Contour Mutation" (WCM) has been developed and will be introduced in below. Finally, several applications of the mutation objects will be presented.

[1] The designations *inside time* and *outside time* are used throughout this paper in preference to the more popular but problematic terms "real-time" and "non-real time".

2. OVERVIEW (OR: MASTERING MORPHOLOGICAL MUTATIONS IN UNDER A MINUTE)

Mutation can be viewed as starting from two *morphologies*: abstract sequences of events, typically represented as cardinal values. These morphologies are called Source and Target. Both are first processed by a Splitter function, dividing Source and Target into mutable and immutable components. The archetypal Splitter functions will classify scalar data into sign and magnitude. For example, Unsigned (Magnitude) Mutations can be described as assigning magnitudes to the mutable component and sign to the immutable component. (See Fig. 1)

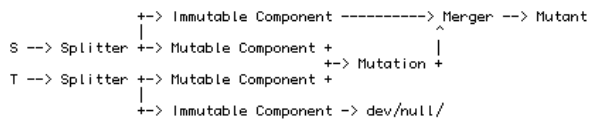

Figure 1 Generic Overview of the Mutation Process.

The core of all mutations is some form of cross-fade between the mutable components of the Source and Target morphologies. The relative strength of presence of Source/Target mutable components in the mutation operation is controlled by a parameter labelled Ω, in the unit range. The Merger function is normally the inverse of the Splitter function.

Typically, data are cardinally scaled. Standard mutation can be either *relative* (i.e., the data stream is viewed as a stream of differences from one time point to the next) or *absolute* (all data are evaluated relative to some absolute reference). In either case, the Splitter will look at the sign and absolute magnitude of the Source and Target data, with three variants:

- Contour mutations: here the sign of the interval is the mutable component; the immutable component is the absolute magnitude of the interval
- Unsigned (or magnitude) mutations: here the absolute magnitude of the interval is the mutable component; the immutable component is the sign of the interval
- Signed mutations: here, the signed magnitude of the interval is the mutable component, leaving a "nil" immutable component.

The standard mutation functions are classified as either uniform or irregular. In uniform mutations, the cross-fade process involves taking a weighted arithmetic mean between the two mutable components. In irregular mutations, either the source value or the target value is taken unchanged to produce the mutation; one of the two values is chosen at random, using a stochastic process. The relative probability of choosing the source or target is determined by the value of Ω.

Conventionally, the three different splitter types are combined with the uniform/irregular dichotomy to yield five different mutation types named as follows:

- UUIM: Uniform Unsigned Interval Mutation.
- IUIM: Irregular Unsigned Interval Mutation
- USIM: Uniform Signed Interval Mutation
- ISIM: Irregular Signed Interval Mutation
- LCM: Linear Contour Mutation

LCM is an irregular mutation using the interval sign as the mutable component. The asymmetry in the above enumeration is because the sign of an interval does not obviously lend itself to a uniform treatment. Nevertheless, this asymmetry prompted the author to investigate the possibilities of using sign as the uniform mutable component. The result is discussed in section 6 below.

3. DESIGN GOALS

The initial goal was to implement time-domain and spectral mutations in Max/MSP, allowing interactive experimentation with the effects of mutation parameters. The "define parameters; process; play; try again" cycle inherent in processing outside time makes it almost impossible to gain a feel for how these parameters effect the resulting sound—arguably one of a composer's central concerns. An additional motivation was the prospect of being able to use the mutation algorithms as a processing technique in interactive performance. This would serve to give the composer greater flexibility—in previous implementations, many of the mutation parameters were static, whereas an interactive implementation should allow the user to vary any and all of the parameters while processing is taking place.

A further, long-term goal was to provide a tool that would aid investigation into variations of the abstract description of mutation algorithms described above.

4. THE OBJECTS

Three external objects have been implemented for mutation algorithms:

- lp.tim~ Time Domain Mutation
- lp.frim~ Spectral/Frequency Domain Mutation
- lp.vim Event-level Mutation of numeric data

All objects can be initialized to a given mutation algorithm, mutation index, clumping value (discussed below), and a delta emphasis value (used in relative intervals). All of these parameters can be modified at any time, even while the mutation process is running. The user can switch between absolute and relative intervals at any time.

An undocumented detail of SoundHack's Spectral Mutation is that FFT bins are automatically grouped into third-octave bands during irregular mutations. In lp.frim~ the user can optionally specify an arbitrary "banding" interval ranging from whole octaves to 1/15-octave. This has proven to be a subtle but valuable tool in irregular Spectral Mutation.

5. A NEW METHOD FOR "CLUMPING"

Irregular mutations have a tendency to generate "crunchy" or "scratchy" sounds, particularly in the time domain. With some input signals there is also a tendency for irregular mutations in the frequency domain to introduce fairly static frequency bands that can sound like a gentle whistling or have a bell-like character of considerable charm. These differences can be clearly audible, although they also depend on the other mutation parameters as well as the source and target signals. To give the composer greater control over these artefacts, a "clumping factor" was introduced [7]. This can be described as a granularity of the randomness in the stochastic process whereby a decision is made whether to use the Source or Target values for the mutable component.

The value of the clumping factor, labelled γ, lies in the unit range. At zero, the random choice between Source and Target mutable values is a simple Bernoulli process. As the clumping factor increases, the Bernoulli process is modified such that the probability of changing state between successive data points decreases, while maintaining the original, long-term distribution of Source and Target components as specified by Ω.

Previous implementations of clumping require specifying the length of the mutation process and follow a notion of "randomness with feedback" propagated by Ames [1][2]. This approach was the single greatest obstacle to an implementation of irregular mutations inside time, since it relied on knowing the size of the entire data set. This is an acceptable assumption to make when working outside time with fixed-length data. However, mutations inside time are necessarily of indeterminate length.

Initial experiments using arbitrary lengths as the basis for clumping calculations proved to introduce undesirable regularities in the resulting audio signals. Also, the goal of allowing the clumping index to be changed interactively is effectively impossible with a fixed mutation length. At the very least, using Amesian feedback would have introduced undesirable latencies when changing the value of the clumping parameter.

An alternative approach adequate for clumping inside time was found by applying the theory of Markov Chains. Indeed, Markov Chains seem to this author to be a more natural basis for clumping than Amesian feedback.

In essence, an irregular mutation can be viewed as having two states: "using source interval" and "using target interval" (abbreviated as S and T, respectively). Polansky's formula for modifying the mutation index based on the clumping index, γ, is

$$\Omega' = \Omega^{1-\gamma} \qquad (1)$$

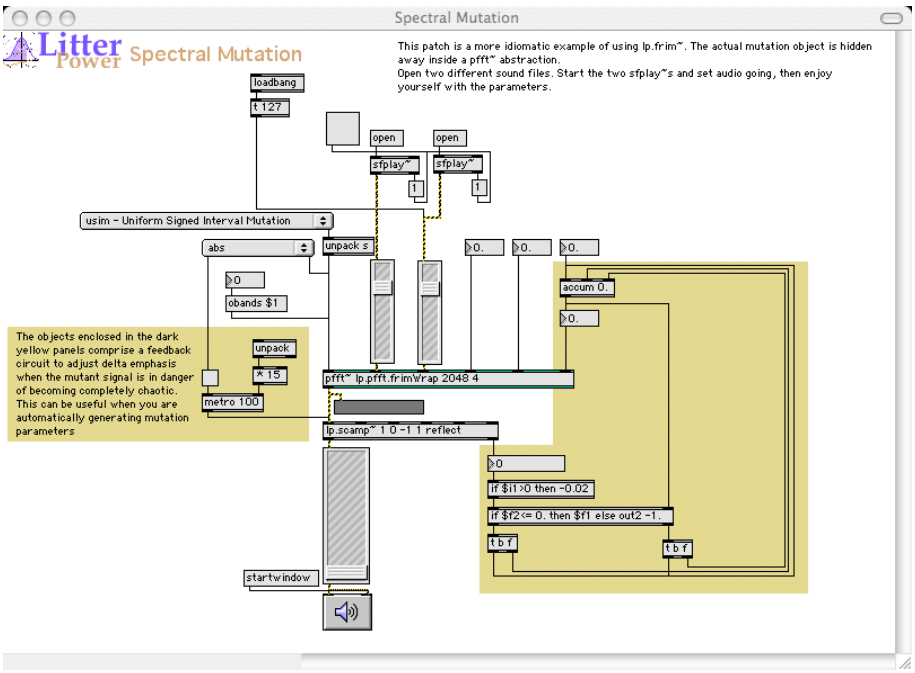

Figure 2 Patch demonstrating a delta-epsilon feedback loop to prevent the audio signal from "exploding."

This was taken as a natural starting point for the probability of "choose T for the next state when the current state is T" (represented as p[Ti, Ti-1]). The probability of p[Si, Ti-1] is then necessarily $1 - \Omega'$. The question was: what probabilities should be chosen for p[Ti, Si-1] and p[Si, Si-1] so that the probabilities of the entire Markov process result in the following equality?

$$p[T] = p[Ti, Ti\text{-}1] + p[Ti, Si\text{-}1] = \Omega \qquad (2)$$

The probabilities for p[Ti, Si-1] and p[Si, Si-1] ought, apparently, to be a function of Ω and γ, with

$$p[Ti, Si\ 1] = f(\Omega,\gamma) \qquad (3)$$

$$p[Si, Si\ 1] = 1 - f(\Omega,\gamma) \qquad (4)$$

Setting up equations for the probabilities p[T] and p[S] implicit in this Markov process and solving for $f(\Omega,\gamma)$ results in the identity:

$$f(\Omega,\gamma) = (1-\Omega')\frac{\Omega}{1-\Omega} \qquad (5)$$

which has the properties of being efficient to implement and applicable to arbitrarily long sequences of events.

6. WEIGHTED CONTOUR MUTATION (WCM)

Whereas mutations based on signed and unsigned intervals both come in "irregular" and "uniform" flavours, previous implementations only recognize an irregular form of contour mutation, Linear Contour Mutation (LCM).

In contour mutations, the mutable component of the source and target data vectors can take on only three values at any point: going up, going down, or remaining constant (more compactly: 1, -1, or 0). If the mutant is only allowed to take on these three values, an irregular mutation appears to be the only option. However, viewed through the abstract approach described in Section 2, there is no intrinsic reason not to implement a uniform version of Contour Mutation. If we look on the mutant contour as a component that can be weighted, controlling what percentage of the source magnitude is to be used when calculating the final mutant value, the formula for WCM (using notation from [5]) can be given as:

$$M_i = M_j + S_{mag}\left(S_{sgn} + \Omega\left(T_{sgn} - S_{sgn}\right)\right) \qquad (6)$$

Although this variant may not be as immediately convincing as LCM, the "Weighted Contour Mutation" (WCM) has proved to be an acoustically attractive resource. The mutants produced by WCM sound vaguely similar those produced by USIM, although retaining an individual characteristic. WCM can be best described as "gentler, kinder" than the other mutation algorithms, which can often produce chaotic and even brutal sonic effects. WCM has proved to be a welcome addition to the mutation toolkit.

7. EXPERIENCE AND FUTURE DIRECTIONS

The mutation objects are available as part of the Litter Power Package [3]. The Starter Pack version of Litter Power includes the lp.tim~ object; lp.frim~ and lp.vim are included in the Professional Bundle.

The author's compositions "The Door: Six Lines" and "The Door: Theme, Lines, Canon" make extensive use of the mutation objects, particularly the ability to let all mutation parameters vary over time in arbitrarily complex ways. The objects have also been used in live performance (the author's *Music for 31 January* and *Music for 11 September*) and in generative compositions, for instance *Radioph*, jointly composed by Zbigniew Karkowski and the author, or the audio-visual installation *realiTV* developed with video artist John Dekron.

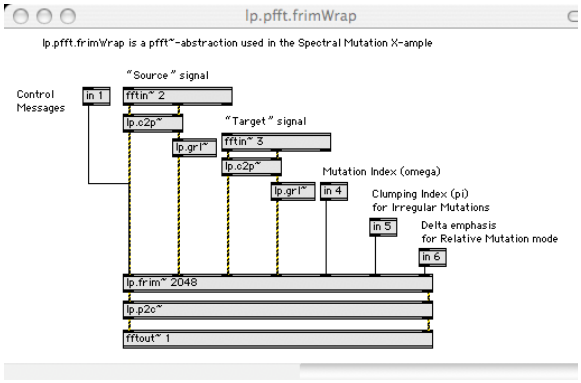

Figure 3: Core of the Spectral Mutation patch in Fig. 2.

Mutation processes can easily become chaotic. Although this is a powerful resource in composition outside time, means for controlling the chaos may be necessary in the context of sound installations or other generative situations. A feedback loop controlling the delta emphasis parameter is an effective means of controlling chaos. One example of programming this kind control is given in Fig. 2. The core processing is performed inside the pfft~ object found in the centre of the patch. The contents of this subpatch are shown in Fig. 3.

Additional applications include:

- Experiments with event-level mutations to control mutations of text texts in an interactive situation.
- Deeper exploration of event-level mutation as an interactive compositional tool, analogous to the approach described by Vaggione [9].
- Interactive experimentation with mutation parameters: this has proved to be a great aid in developing a better understanding of how these parameters interact.
- Combinations of mutation processes beyond the "canonical" LCM/UUIM and LCM/IUIM. These objects combined with the programmability of the Max/MSP environment allow for arbitrarily complex composites.

Some composers have found the wealth of parameters and options in mutation algorithms overwhelming. Recognizing that some will prefer a less complex approach, a simple, one-parameter-controls-everything object named lp.emeric~ has been developed. A further object with the working title lp.minerva~ is under development. The goal is to provide a middle ground between lp.frim~'s flexibility and complexity, on the one hand, and lp.emeric~'s simplicity.

Recently, Verfaille and Depalle [10] have touched on Morphological Mutation, viewing it as a special case of their source-filter model for adaptive effects processing. The convolution of their approach with the flexibility provided by Max/MSP-based tools for mutation processing promises to be a fertile ground for new audio processing techniques.

Finally, work is now under way exploring possibilities for applying mutation techniques to video. Whereas a simple pixel-by-pixel approach has shown itself to be of only limited appeal, work in the spectral domain promises to generate a compelling and attractive new visual technique.

8. ACKNOWLEDGEMENTS

Thanks to Larry Polansky and Tom Erbe for insights into the implementation details of Morphological Mutations in HMLS and SoundHack. Max Neuhaus was a dedicated beta tester and contributed useful wrapper abstractions. Akira Rabelais' software was an unending source of entertainment. Thanks also to the Bergen senter for elektronisk kunst, where I was able to turn a raw hack into reliable software during a guest residency. Readers' comments and suggestions are gratefully acknowledged. Additional support and ideas came from Kasper T. Toeplitz and Zbigniew Karkowski. Particular thanks to Wendy and Danny, who thought it all sounded pretty weird.

9. REFERENCES

[1] Ames. Ch. "A Catalog of statistical distributions: Techniques for transforming determinate, random, and chaotic populations", *Leonardo Music Journal*, 1991.

[2] Ames. Ch. "A Catalog of sequence generators: Accounting for proximity, pattern, exclusion, balance and/or randomness." *Leonardo Music Journal*, 1992.

[3] Castine, P. *Litter Power*. 4-15 Music & Technology, Berlin, 2002-07.

[4] Polansky, L. "Morphological metrics: An introduction to a theory of formal distances", *Proceedings of the International Computer Music Conference*, Champaign-Urbana, 1987.

[5] — "More on Morphological Mutations: Recent Techniques and Developments *Proceedings of the International Computer Music Conference*, San Jose, 1992

[6] Polansky, L. and Erbe, T. "Spectral Mutation in SoundHack: A brief description." *Proceedings of the International Computer Music Conference*, Banff, 1995.

[7] Polansky, L. and McKinney, M. "Morphological Mutation functions: Applications to motivic transformation and a new class of cross-synthesis techniques." *Proceedings of the International Computer Music Conference*, Montréal, 1991.

[8] Rabelais, A. *Agreïphontes Lyre*. <http://akirarabelais.com/software/software.html>.

[9] Vaggione, H. "Vers une approche transformationnelle en CAO", *Les Actes de "Journées d'Informatique Musicales"*, Caen, 1996

[10] Verfaille, V. and Depalle, Ph. A. "Adaptive effects based on STFT, using a source-filter model". *7th International Conference on Digital Audio Effects (DaFX)*. Naples, 2004.