Morphological Mutation Functions
Applications to Motivic Transformation and a New Class of Cross-Synthesis Techniques

Larry Polansky and Martin McKinney
Bregman Electro-Acoustic Music Studio
Dartmouth College, Hanover, NH 03755
email: larry.polansky@mac.dartmouth.edu; martin.mckinney@mac.dartmouth.edu

Abstract: This paper is an introduction to a class of algorithms for melodic, formal and timbral based transformation using *mutation functions.* These functions use formalized *morphological metrics*, or *distance functions* on *morphologies* (e.g shapes, motives, melodies, waveforms, or any sets of ordered data). The mathematical/musical parameters of these functions are presented formally, along with applications to melodic transformation and cross-synthesis, called *mutation synthesis*, which uses these functions to modulate one waveform into another along the time and frequency domains.

General Introduction

The concepts of *distance* and *similarity* are fundamental to a discussion of form. In order to decide, at the very least, if two objects are distinct, a concept of similarity is needed. Measuring distance in musical morphologies (melodies, duration sequences, shapes in other parameters, and large scale forms) is difficult, since several aspects of musical morphology must be taken into account, including, but not limited to: *contour, intervallic magnitude, combinatoriality* (that is, how much interrelationship of component parts to consider), and aspects of *invariance* under certain transformations. Polansky (1987) has outlined several simple distance functions, called *morphological metrics*, which attempt to isolate several factors of similarity.

By making the idea of metrics *generative*, rather than analytic, we develop simple transformations, called *morphological mutations*, which yield interesting and fertile procedures for "cross-fading" one morphology into another along perceptual axes. These functions have an implied distance function, called the *index*, which can be thought of as an *inverse metric*.

Introduction to Mutation Functions

Mutation functions are of the general form:

$$f(S,T,\Omega) = M$$

where S is a *source morphology* (melody, duration sequence, ordered set of samples, etc.), T is a *target morphology*, M is the resultant *mutant*, and Ω is some value between 0 and 1. Intermediate values for Ω yield morphologies which have morphological characteristics (depending on the mutation function) somewhere between the source and target [McKinney, 1991].

Although each mutation function "cross-fades" S into T through M, they do so in different ways, and there are several mutation parameters which determine the *trajectory* of the mutation and, more importantly, what morphological property of S and T is used for the transformation. As a simple example, a pure *contour mutation* with $\Omega = 1$ will yield a morphology that is only identical to T in contour (direction of intervals). In other words, a contour mutation with $\Omega = 1$ will produce a morphology T' for which:

$$d(T,T') = 0$$

under some metric function d with a well-defined notion of *contour* [Polansky, 1987], [Polansky and Bassein, 1990].

Types of Mutation

Several classes of mutation are developed based upon aspects of morphology: *order, interval contour, interval magnitude*, and *linearity vs. combinatoriality* of intervals. *Contour mutations* transform the source into the target solely on the basis of *interval direction* (or sign). *Magnitude mutations* use the interval magnitude, signed or unsigned (the former is a combination of magnitude and contour).

Figure 1a shows a four element morphology with different contours between each pair of successive elements: the contour between elements A and B is positive (+1); the contour between elements B and C is null (0); and the contour between elements C and D is negative (-1). In *Figure 1a*, we are only considering what Polansky has called *linear contour* [Polansky and Bassein, 1990], [Polansky, 1987], the relationships between adjacent elements of a morphology.

Interval magnitude is the absolute magnitude difference between two elements in a morphology.

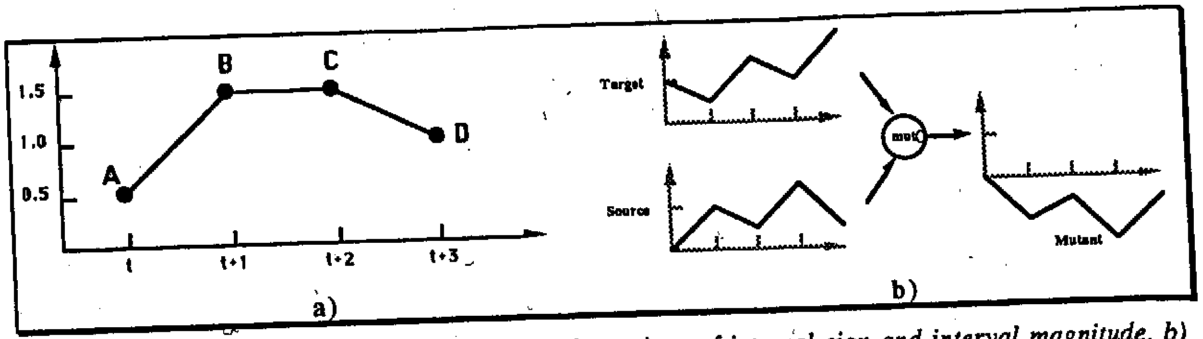

Figure 1. a) A sample morphology illustrating the notions of interval sign and interval magnitude. b) Contour Mutation with an index of 0.5 and clumping value of 0.0.

In *Figure 1a*, the interval magnitude between points A and B is 1.0, between points B and C is 0.0, and between points C and D is 0.5.

For more information on the associated metrics for these functions see [Polansky, 1987]. The actual *interval function*, called Δ, may be completely general, and defined specifically for a given parameter. For example, it can be arithmetic, ratiometric, or even as simple as "equal to or not." All of the mutation equations below can be rewritten to incorporate this more general notion of the interval Δ between two elements of a morphology (a detailed exploration of this is beyond the scope of this paper).

The *mutation index*, Ω, functions as a kind of "knob" in these mutations. In timbral mutations, Ω becomes the actual index of modulation, and can be a time-varying function (at audio or sub-audio rates), a real-time input, a constant, a sample itself, or a simple lookup table (like a ramp or sinusoid).

Contour Mutations

The *contour mutation* function mutates a percentage of the intervals of a source morphology by assigning the contour of the respective interval in the target morphology to the interval magnitude of the source. The index determines what *percentage* of source intervals, that differ in contour from their respective target intervals, get mutated (take on the contour of the target). By only considering respective intervals from the source and target that differ in sign, the index of the mutation becomes a linear and exhaustive "measure" of the metric space between the source and the target. A contour mutation with an index of 0.0 would produce a mutant identical to the source. With an index of 1.0, the mutant would have all of the interval magnitudes of the source and the interval signs of the target. Note that the above only describes a *linear* contour mutation, in other words, *not all* of the contour relationships between all elements of the morphology are considered (like, for example, the contour relationship between the 2nd and 5th elements).

Clumping

Although the index in the *contour mutation* function determines what percentage of intervals get mutated, it does not determine exactly which intervals get mutated for a given mutation. There are many ways the mutated intervals can be distributed throughout the resultant morphology. So far we have experimented with the use of a fixed *clumping value*, ∂, and *stochastic clumping*.

The *clumping value* ∂ of a mutation function determines what percentage of the mutated intervals are mutated *consecutively*. ∂ ranges from 0.0 to 1.0: a value of 0.0 spreads the mutated intervals uniformly across the mutated morphology and a value of 1.0 groups all of the mutated intervals into one clump of mutated intervals. For example, with $\Omega = 0.5$ and $\partial = 1.0$, the first half of the morphology would be mutated (clumping is as high as possible), the second half left unchanged.

With *stochastic clumping*, each interval in the source with a different contour than its respective interval in the target is selected to be mutated with a probability equal to the index. Thus, for two morphologies, successive mutations using stochastic clumping and the same index could (and probably will) result in different mutation morphologies, creating a different *trajectory of mutation.*

Clumping is critical in using mutation functions to go from one morphology to another by varying index. If stochastic clumping is used, it is possible that the perception of the linear progression of mutant morphologies will be less evident than if the same clumping value is used throughout the progression, as there could be much variation in the order of mutated intervals.

Contour Mutation

The simple *linear contour mutation* function is as follows:

$$M_i = M_{i-1} + \frac{(T_i - T_{i-1})}{|T_i - T_{i-1}|} * |S_i - S_{i-1}|$$

for mutated intervals, and

$$M_i = M_{i-1} + (S_i - S_{i-1})$$

for non-mutated intervals, for all $M_i \in M$, all $S_i \in S$, and all $T_i \in T$ assuming the source S, the target T and the mutant M are all equal length morphologies (we will assume these conditions for all equations below unless otherwise stated) and where the index Ω and the clumping value ∂ determine whether the interval will be mutated as described above. *Figure 1b* shows the *contour mutation*, with $\Omega = 0.5$ and $\partial = 0.0$, on two simple contours. Since $\partial = 0.0$, the mutated intervals are spread evenly throughout the contour, and every second interval is mutated starting with the first interval.

Magnitude Mutations

Regularity

Mutation functions that operate on the interval magnitude can have one of two types of indexes. A *Uniform* index mutates all interval magnitudes of the source by a percentage of the difference of the source interval magnitudes and the target interval magnitudes. The index determines the percentage of the difference. Thus the mutation interval magnitude, M_{int}, is calculated by:

$$M_{int} = S_{int} + index * (T_{int} - S_{int})$$

where S_{int} is the respective source interval magnitude and T_{int} is the respective target interval magnitude. *Figure 2a* shows an *Interval Magnitude* mutation, with a *Uniform* index of 0.5, of two simple morphologies. Each interval magnitude in the mutant is one half of the difference of the respective interval magnitudes in the source and target.

An *Irregular* index mutates a percentage of the source interval magnitudes by applying the full target interval magnitude of the target's respective interval. The index determines what percentage of the mutant's intervals assume the target's interval magnitude. The rest of the mutant's intervals keep the source's interval magnitudes. *Figure 2b* shows an *Interval Magnitude* mutation, with an Irregular index of 0.5, of two simple morphologies. In this example, every second interval magnitude is the same as the source, starting with the first one, and every interval magnitude between those is the same as the target. The mutated intervals (mutated intervals refer to the intervals in the mutant that have the target's respective interval magnitude) are spread evenly throughout the mutant. *Clumping*, as described above, is used to distribute the mutated intervals.

The *Contour Mutation* only uses an *Irregular* index — contour mutation of a specific interval is an all or none operation. The distinction between *Uniform* and *Irregular* indexes is the *quality* of the index.

Signed and Unsigned Mutations

Mutation functions are also classified by whether or not they operate on the *interval sign* of the morphology. The *Contour Mutation* function is inherently signed. Mutation functions that operate on the interval magnitude, however, can be *Signed* or *Unsigned*. A *Signed* mutation function gives the mutated interval the sign (or contour) of the respective target interval and an *Unsigned* mutation function gives the mutated interval the sign (or contour) of the respective source interval.

Some Magnitude mutations

Four magnitude mutations are defined based on the two characteristics just described:

1) *Uniform Signed Interval Magnitude Mutation (Absolute Magnitude Mutation)*
2) *Irregular Signed Interval Magnitude Mutation*
3) *Uniform Unsigned Interval Magnitude Mutation*
4) *Irregular Unsigned Interval Magnitude Mutation*

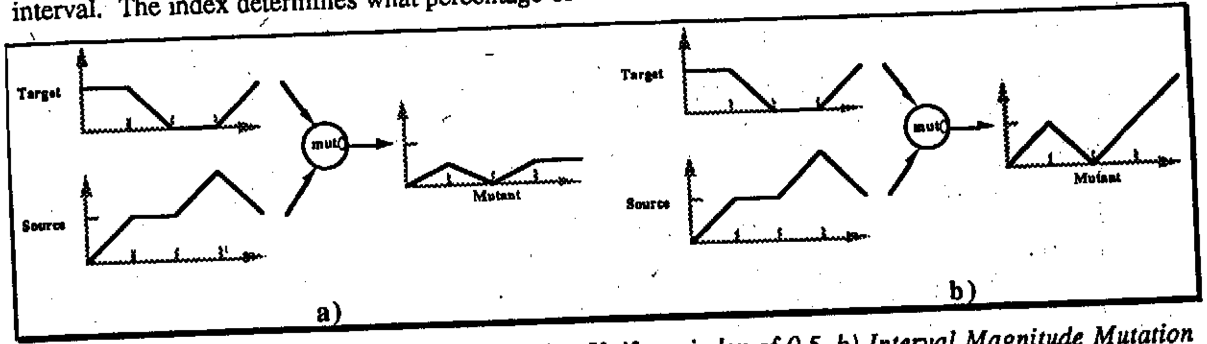

Figure 2. a) Interval Magnitude Mutation with a Uniform index of 0.5. b) Interval Magnitude Mutation with an Irregular index of 0.5.

Uniform Signed Interval Magnitude Mutation (USIM) (or The Absolute Magnitude Mutation - AM)

The *Absolute Magnitude* mutation mutates absolute values of the source and target elements, rather than *intervals*. This is essentially the same as the *USIM* if the first value of the source and the target are the same. The function is:

$$M_i = S_i + \Omega * (T_i - S_i)$$

This mutation is a simple crossfade between the source and target. *Figure 3a* shows the *AM* mutation, with $\Omega = 0.5$. Each element (point) in the mutant is half way between the respective elements in the source and target.

Irregular Signed Interval Magnitude Mutation (ISIM)

The *ISIM* is the same as the *Absolute Magnitude* mutation but the *ISIM* has an *Irregular* index. As in the *USIM*, since the mutant intervals contain the sign and interval magnitude of the target, when the index is 1.0, the mutant *is* the target. The function is:

$M_i = M_{i-1} + (T_i - T_{i-1})$ *for mutated intervals*

$M_i = M_{i-1} + (S_i - S_{i-1})$ *for non-mutated intervals*

where the index Ω and the clumping value ∂ determine whether the interval will be mutated in one of the several manners described above.

Figure 3b shows the *ISIM*, with $\Omega = 0.5$ and $\partial = 0.0$. Since $\partial = 0.0$, the mutated intervals are spread evenly throughout the contour, and every second interval is mutated, starting with the first interval.

Uniform Unsigned Interval Magnitude Mutation (UUIM)

The *UUIM* applies the source interval sign to a percentage difference (depending on the index) of the source and target interval magnitudes.

The function is:

$$M_i = M_{i-1} + \frac{(S_i - S_{i-1})}{|S_i - S_{i-1}|} * (|S_i - S_{i-1}| + \Omega * |(|T_i - T_{i-1}| - |S_i - S_{i-1}|)|)$$

Note that:

$$\frac{(S_i - S_{i-1})}{|S_i - S_{i-1}|}$$

is just the sign of the source interval.

Figure 4a shows the *UUIM*, with $\Omega = 0.5$. Each interval has an absolute magnitude that is half way in between the absolute magnitudes of the respective intervals in the source and the target. Each interval has the same sign as the source.

Irregular Unsigned Interval Magnitude Mutation (IUIM)

The *IUIM* mutates a percentage of the source intervals (determined by the index) by assigning the sign of the source and the complete magnitude of the respective target interval to the mutant. An interval that is not mutated keeps its source interval magnitude and sign. Because this mutation function has an *Irregular* index, a clumping value ∂ determines the distribution of mutated intervals in the mutant.

The function is:

$$M_i = M_{i-1} + \frac{(S_i - S_{i-1})}{|S_i - S_{i-1}|} * |T_i - T_{i-1}|$$

for mutated intervals, and

$$M_i = M_{i-1} + (S_i - S_{i-1})$$

for non-mutated intervals, where the index Ω and the clumping value ∂ determine whether the interval will be mutated in one of the several manners described above.

Figure 4b shows the *IUIM*, with $\Omega = 0.5$ and $\partial = 0.0$. Since $\partial = 0.0$, the mutated intervals are spread evenly throughout the contour, and every second interval is mutated starting with the first interval.

Mutation Function Relationships

An important relationship is that of the *contour mutation* to the *Unsigned Interval*

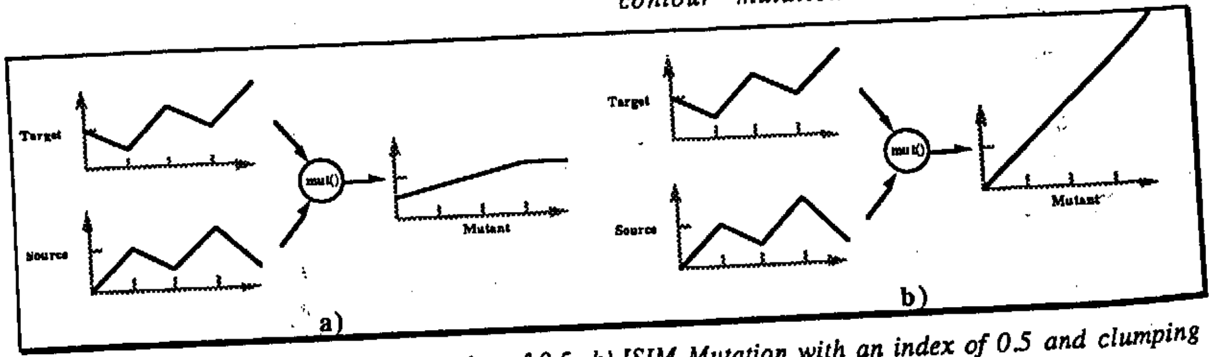

Figure 3. a) AM Mutation with an index of 0.5 b) ISIM Mutation with an index of 0.5 and clumping value of 0.0. ISIM Mutation with an index of 0.5 and clumping value of 0.0.

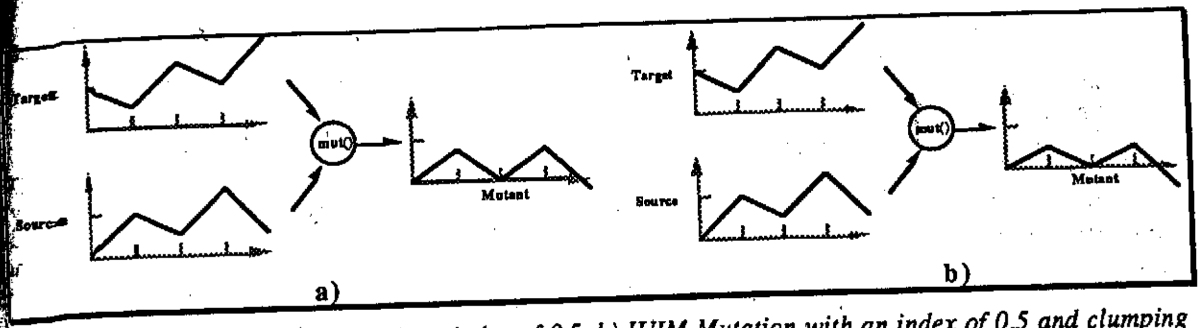

Figure 4. a) UUIM Mutation with an index of 0.5. b) IUIM Mutation with an index of 0.5 and clumping value of 0.0.

Magnitude mutation functions. These are mutually exclusive and, with indexes of 1.0, are collectively exhaustive of the metric space they define. A *contour mutation* with an index of 1.0 followed by an *Unsigned Interval Magnitude* (*Uniform* or *Irregular* index) with an index of 1.0 using the output of the *contour mutation* as the source and the original target as the target, yields a mutant that is *identical* to the target. *Figure 5* is an example of this combination of mutation functions on two simple contours. The resulting mutant has the *contour* of the target in the first mutation and the *interval magnitudes* of the target in the second. Thus, combining these mutations, in either order, provides a means for progressing from a source morphology to a target. Using time varying indexes, modifying the way they approach 1.0, and by altering ∂, one can create different trajectories for the mutation (see also *Figure 6*). The mutation will, however, always become the target when both of the indexes are 1.0.

Mutation Synthesis

The authors have applied the mutation functions to digital sound samples, in both the time and frequency domain. Because the mutation functions are non-linear, it is difficult to predict what the output (mutant) will sound like or describe its frequency content (a typical method for analyzing sound). This suggests an experimental and exhaustive approach to applying the mutation functions to digital sound and evaluating the output aurally. The mutation functions were coded in C on a NeXT™ Computer and designed to operate on sounds of the NeXT sound file format [McKinney, 1991].

Time Domain

Sounds are mutated in the time domain with the functions describe above by treating a sound as a morphology and its individual sample values (discrete points of the signal amplitude) as elements of that morphology.

Normalization and Filtering

The inherent instability of the *unsigned* mutation functions is an obstacle in applying them to time domain sound samples. Typically sources and targets have different contours and interval magnitudes, and applying *just* the interval magnitudes or *just* the contour of one to the other *offsets* the absolute magnitude of the resultant, especially for periodic signals. Since the offset is the same for each period of the signal, the overall offset grows by that constant amount every period. As the input signals grow in length, the offset grows and, if straight normalization is used, the desired signal component amplitudes decrease until they are inaudible. Typically it does not take a very long signal (on the order of one second at a 44100 sampling rate) before the desired signal is completely inaudible.

One method of eliminating this offset is to fit a straight line to the signal using linear regression and subtract the line from the signal. This does not significantly distort the desired signal and allows it to expand when normalized. A straight line approximation of a signal is obtained by the following equation:

$$y = \left(\frac{\frac{\sum x^2 - (\sum x)^2}{n}}{\frac{\sum(x*y) - \sum x * \sum y}{n}} \right) * x$$

This method works for steady state signals. For time varying signals, however, the offset created by the mutation function may be positive for part of the signal and negative for another part, so a straight line subtraction is not sufficient. Higher degree polynomials could be fit to the mutant, but that would not be very efficient. After a closer look at the *rate* (or slope) of the offset created by the mutation functions, it was discovered to generally be below the audio range. In effect, it is an inaudible residue from the mutation functions themselves. By applying a *high pass filter* with a cutoff frequency just below the hearing threshold (about 16 Hz) all of the residue disappears and the mutant can be fully normalized. Because the residue is inaudible, filtering it out does not destroy the validity of the mutation functions as cross-synthesis functions.

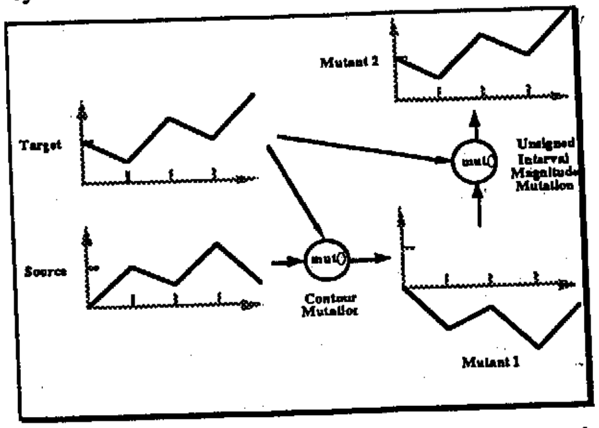

Figure 5. Contour/UIM mutation with indexes of 1.0

Frequency Domain

Work on mutations of the frequency magnitude of sound signals is still in the preliminary stages, with some very promising results. The basic method for mutating two sound signals in the frequency domain is to take a running Fourier analysis of both the source and the target and perform the mutation on the magnitude window. First, an N-point Fast Fourier Transform (FFT) is taken of both Source and the Target sounds, which is obtained by:

$$X(k) = \sum_{n=0}^{N-1} x(n) * e^{-j(2\pi kn/N)}$$

where: k is angular frequency ranging from 0 to 2π, N is the length of the *FFT*, and $x(n)$ is the time domain signal.

Under the current NeXT implementation, the length of the FFT is user selectable, and the overlap size of the windows is one half of the FFT length. This introduces a trade-off of high frequency resolution and low window (to signal) resolution for long FFTs and the reverse for short FFTs. The size of the FFT has not been experimented with very much at this stage. The FFT is expressed as a sum of complex exponentials because:

$$e^{jw} = \cos w + j\sin w$$

and thus each frequency can be represented as complex exponential or alternatively as a magnitude and phase where:

$$Magnitude = \sqrt{a^2 + b}$$

$$Phase = \tan^{-1}\left(\frac{b}{a}\right)$$

where: a is the amplitude of the real (cosine) part and b is the amplitude of the imaginary (sine) part.

In this first implementation of the mutation functions, the frequency magnitude of the signals was mutated and the phase was simply crossfaded using the following method:

$$Phase = index * targetPhase + (1 - index) * sourcePhase$$

At first, phase was ignored but the signals in sequential windows did not line up properly. This method of crossfading the phase was the simplest way to stay within the basic constraints of the mutation functions: when the index is zero, the mutant should resemble the source and when the index is one, it should resemble the target. One can see that this function is a simple *Absolute Magnitude* mutation. In future implementations, mutating the phase as well as the magnitude of the frequencies with the same mutation function will be investigated. Mutating the frequency data in their complex representation will be experimented with as well.

After the frequency data is mutated the inverse FFT is taken and each window is multiplied by a crossfade window and overlapped with the preceding and following windows. In the current implementation, before the inverse FFT is taken, the frequency magnitude data is normalized so there are no negative frequencies, which could have resulted from the mutation function. A crossfade window provides a clean flow from one window to the next.

Melodic Example

The following melodic example, *Figure 6*, uses a combined mutation: *Linear Contour* with stochastic clumping, followed by *IUIM (Irregular Unsigned Interval Magnitude)*, again with stochastic clumping. This example mutates the first several measures of a fiddle tune (the source), "The Road to Chimacum" by Patricia Spaeth [Williams, page 13], to an excerpt (of equal length) from Bill Cole's transcription of John Coltrane's solo from the tune "Countdown" (the target) [Cole, page 105]. "... Chimacum" is transposed up a major sixth so that both of the tunes can start on the same pitch, and only pitches are mutated (all the rhythms in the source and target are eighth notes). This example shows 9 melodies, more or less equal spaced mutations from the source ($\Omega = 0.0$) to the target ($\Omega = 1.0$), although values for $\Omega = .2$ and $\Omega = .8$ are omitted for space reasons only.

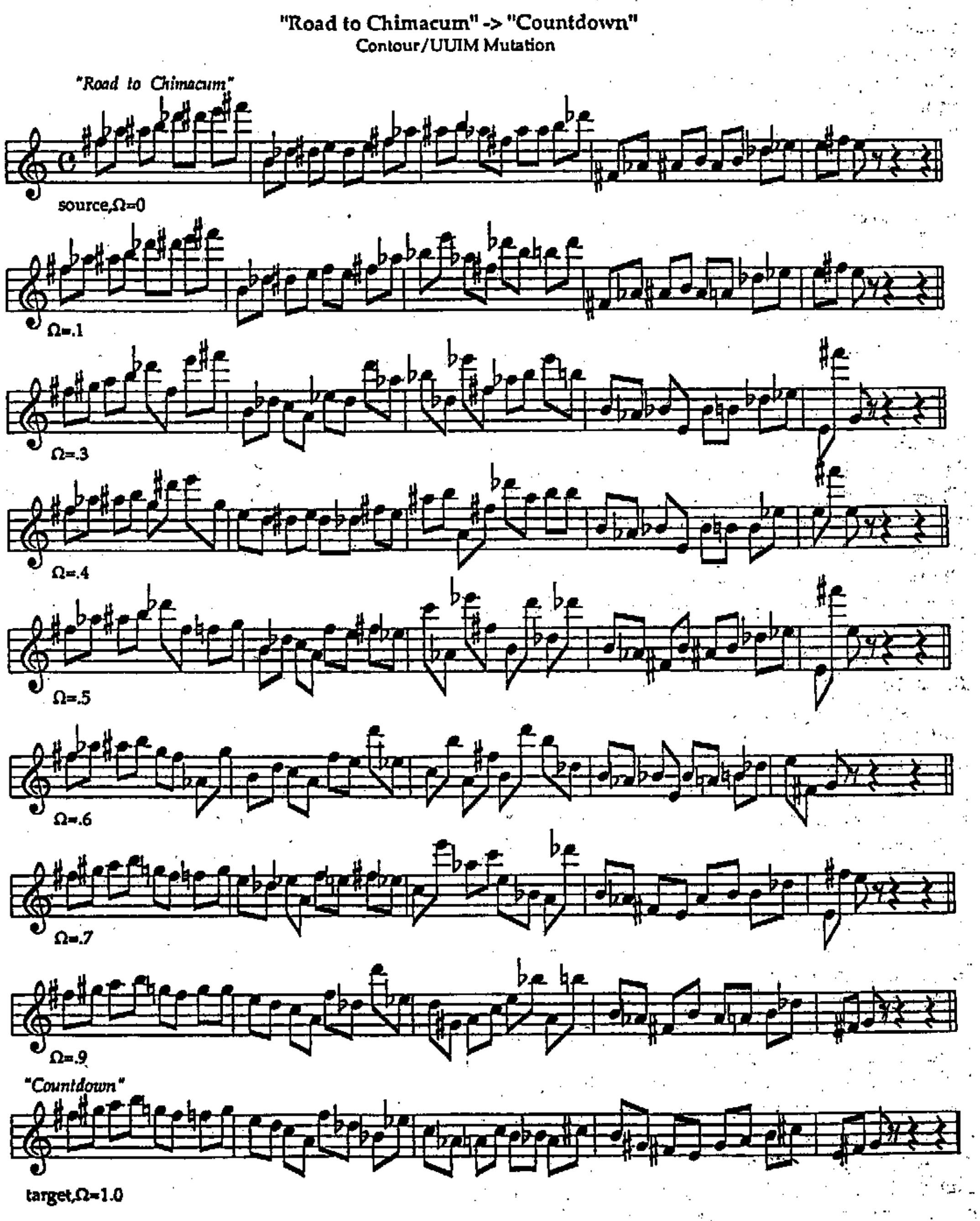

Figure 6. Melodic Example

This example was written algorithmically in HMSL and saved to FINALE as a MIDIFile.

In this example, the *contour* and *IUIM* mutations are not synchronized, each one stochastically chooses intervals to mutate from the source towards the target for each mutated melody, independently of the other mutation. In other words, the same HMSL program run several times, would produce different intermediary mutations, but the general trajectory of mutation would be the same, eventually generating the target ($\Omega = 1.0$). A slightly different interval function is used in this example, useful in normalizing the mutations. All interval calculations are made to the first element of the morphologies. That is, a simple signed interval is of the form $S_i - S_1$, instead $S_i - S_{i-1}$.

Current and Future Work

These mutations have been implemented on two different platforms, in the computer music language *HMSL* and on the *NeXT* computer. The HMSL implementation allows for sophisticated real-time melodic mutations, as well as waveform mutation (also real-time) on the Commodore Amiga and on the Macintosh-based *DigiDesign AudioMedia* or *Sound Accelerator* cards. The use of HMSL allows for these mutations to be controlled in a variety of interactive contexts. The mutations have been used in several pieces by Polansky, including *Bedhaya Sadra/Bedhaya Guthrie*, (1990a), and *3 Studies*, for performers and live interactive computers (1990b).

On the NeXT machine, McKinney has implemented a simple, user-friendly platform for experimentation with time and frequency domain mutation of soundfiles. This implementation allows the user to specify values for clumping, index, index waveforms, types of mutation, and so on, and has so far been the primary source for beginning to understand the sonic aspects of mutation synthesis [McKinney, 1991].

Work on mutation functions and mutation synthesis is just beginning, and many theoretical and practical issues need addressing, including: 1) continued expansion of these two platforms; 2) *implementation of 56000 code inside of HMSL* for a user-friendly real-time mutation synthesis platform, to be used in performance at both the event and timbral levels; 3) a *formal and mathematical understanding* of the relationships between mutation functions and standard synthesis algorithms, so that, for example, resulting spectra will be more predictable; 4) implementation of fast algorithms for *combinatorial mutations*, both at the melodic and waveform level; 5) more sophisticated and accurate *normalization* and *filtering* techniques; 6) algorithms and methods for mutating morphologies of unequal lengths; 7) a *systematic set of experiments* to gain sonic understanding of these functions, starting with simple waveforms, constant phases, and known indexes (as in waveshaping synthesis); 8) more work on the effects of phase differences in the time domain; 9) experiments with different values and types of *clumping*; 10) techniques for mutations of morphologies of unequal length; 11) more experimentation in the weighted concatenation of different mutation functions. We have already begun to work on most of these issues, but space limitations prevent us from describing them in more detail here.

References

Cole, Bill. 1976. *John Coltrane*. Schirmer Books. New York.

McKinney, Martin. 1991. "Mutation Synthesis." Master's Thesis, Dartmouth College M.A Program in Electro-Acoustic Music.

Polansky, Larry, and Bassein, Richard. 1990. "Possible and Impossible Melody: Some Formal Aspects of Contour." Paper presented at the Society for Music Theory Conference, Oakland, CA. Submitted for publication.

Polansky, Larry. 1990a. *Bedhaya Sadra/Bedhaya Guthrie*. Score, for voices, kemanak, and gamelan instruments. Frog Peak Music.

— 1990b. *3 Studies*. Software. Frog Peak Music.

— 1987. "Morphological Metrics: An introduction to a theory of formal distances." In *Proc. of the ICMC*. Compiled by James Beauchamp. San Francisco. Computer Music Association.

Williams, Vivian, editor. 1986. *141 Brand New Old Time Fiddle Tunes (by Pacific Northwest Composers). Volume 2*. Voyager Publications. Seattle.