Rene Wooller
Queensland University of Technology
Victoria Park Road
Kelvin Grove, 4059
Australia
r.wooller@qut.edu.au

Review of Compositional Morphing: Works, Techniques, Applications and Possibilities

Abstract

This paper presents a review of compositional morphing systems, briefly examining the current techniques used and touching on the related motivations, applications and possibilities. Firstly, terms will be explained, the artistic purposes and application domains related to morphing discussed, background examined and the scope of the review defined. A number of significant works will then be briefly described. Following this, each system will be positioned within a framework which allows the reader to visualise and compare the nature of the various techniques used. The applicability of the techniques is then discussed. The article concludes with a summary of the state of morphing, pointing to possibilities for future research.

Introduction

Many projects in algorithmic composition have been carried out that explore various morphing techniques, for different purposes. While there are collections of more or less relevant reference material (Polansky 2005), no survey of compositional morphing in particular is available. The review provided here will be useful as a repository of ideas as well as a reference point for ascertaining the state and direction of the field and identifying areas for future development.

But first of all, what *is* morphing? The general definition of morphing given by Larry Polansky (1992) is mostly followed here, albeit with simpler terms. It is conceived as a process that utilises two musical patterns to create a "mutant" or "morph" that may work as a transition between the two. Formally,

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

where S is the source pattern, T, the target, M, the resulting mutant and, Ω, a variable to determine how closely M will resemble S or T. Ω is normalised such that when $\Omega = 0$, $M = S$ and when $\Omega = 1$, $M = T$.

It is important for us to distinguish between "interpolation", "morphing" and "transformation". Interpolation is a technique for the estimating unknown values between known points. Morphing, however, is more concerned with aesthetic integration of separate music; interpolation may or may not be used as part of this. A transformation is any method for changing a given piece of music, including morphing.

Two artistic purposes that motivate the use of morphing in music can be distinguished, with associated applications. One is to create a *transition* whereby S, M and T can be positioned in sequence to create a smooth and/or coherent whole. This has potential in multimedia, computer games, live DJ mixing or any context where automatic transitioning between music is required. Another artistic purpose can be to create a *mutant*, whereby M has aesthetic interest in the way it shares properties of S and T, but may or may not function as a transition. This is more applicable to computer assisted composition and musical experimentation.

This paper is particularly concerned with "Compositional" Morphing (CM) where the musical data being morphed consists of note-events with parameters such as onset, pitch, duration and dynamic. This is contrasted with morphing other aspects of music such as sound (Polansky and Erbe 1996) or expression (Canazza, Poli, Drioli, Roda and Vidolin 2001). Classifying a musical algorithm as "compositional" is difficult, due to conversions to various non-note-level representations that facilitate the design and execution of the algorithm. As well as this, sonic parameters can sometimes operate in a compositional way, for example, severe amplitude modulation on a continuous sound could be perceived as a series of "note-events". This review concentrates mostly on algorithms that take note input and generate note output, while others that generate note output only may be covered less comprehensively.

Background in music

The origin of the concept of morphing in music is difficult to define clearly, however the idea has been explored by composers for at least three centuries with the beginning of the classical period, which emphasised key modulation and contrasting emotions within the same work. Oppenheim (1995) points to the development of the Sonata from 1780 which often includes a transitional section. Although the Sonata is not particularly concerned with morphing, the composer deals with sections of contrasting themes. Various composers have explicitly dealt with the concept of metamorphosis, including Cage's 1938 series (Cage 1938), Hindemith's 1943 adaption of Weber themes (Hindemith 1989) and Philip Glass (Glass 2000).

Particularly relevant aspects of traditional music theory include modulation techniques, pivot notes and chords and bridge sections. These techniques have been dealt with previously (Wooller and Brown 2005) and are absent from this review which focuses instead on computationally explicit algorithmic systems.

Experimental composers and theorists have also considered various aspects of morphing. Tenney (Tenney and Polansky 1979) touched on issues relating to the similarity and difference of temporal gestalt units. Rosenboom speculated on transitional topologies and the stochastic induction of perturbations within mutation functions (Rosenboom 1982). Composers who developed software systems that embody similar ideas within the context of compositional morphing are discussed below.

Compositional Morphing Systems

Music IV and GRIN94

Max Mathews created the first musical morphing algorithm in 1966 on the MUSIC IV platform (Mathews and L.Rosler 1969). This work developed as a demonstration of the algorithmic possibilities of Mathews' and Rosler's graphical input program, GRIN94. In this system, a monophonic melody was represented with separate functions, or envelopes, for each dimension of amplitude, frequency, duration and glissando. The frequency functions were made of flat (gradient 0) segments for the tone of each note, while the amplitude function was used to accent the first beat in each measure. Glissando was not used in the morphing example. The discrete note durations, or inter-onset times, required conversion to a continuous function in order to become algebraically manipulable.

Mapping discrete start times to a continuous domain is a problem that can be approached in many ways; Mathews' and Rosler's technique is particularly ingenious. To generate a melody, a note would be created and the duration function would be sampled at that point. The sampled value would specify the inter-onset distance to the next note and sample point. The "self-synchonising" form of the duration function required that each segment has a gradient of -1, so that if the sampling is ahead or behind a certain amount, the next note and sample point will be in time.

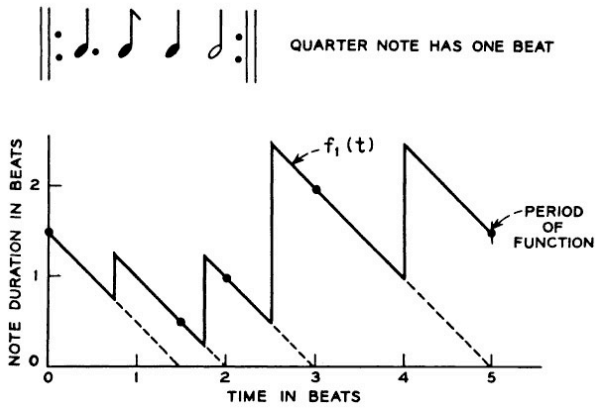

Figure 1. *Self-synchronizing function for inter-onsets (reprinted with permission).*

Half-way between each note was chosen arbitrarily as the start and end points for each of these segments. Combining self-synchronising functions with others will result only in other self-synchronising functions and so the coherent quantisation of durations to known values is inherent in the style of representation.

Having dealt with the problem of continuous representation, morphing becomes simply a matter of combining the functions in each pattern, using Ω to weight each one. For its time, a somewhat convincing result was recorded and produced on the vinyl accompanying the book (Mathews and L.Rosler 1969). It morphs from *the British grenadiers* to *When Johnny comes marching home* and back.

HMSL

The Heirarchical Music Specification Language (HMSL) was developed by Phil Burke, Larry Polansky and David Rosenboom from 1980, and is implemented in FORTH (Burke, Polansky and Rosenboom 2005). It is partially inspired by the musical theories of Jim Tenney, including the notion that general patterns should be easily mappable to various levels of a music hierarchy. Polansky developed code in HMSL to aid the experimental morphing of music within some of his compositions, including *distance music* (Polansky 1987), *Bedhaya Guthrie/Bedhaya Sadra for Voices, Kemanak, Melody Instruments, and Accompanimental Javanese Gamelan* (Polansky 1996), *51 Melodies* (Polansky 1991), *Two Children's Songs*(Polansky 1992) and *Road to Chimachum*. MIDI renderings of these last three can be heard online (Polansky 2006).

This music was based on a theoretical framework developed by Polansky that explores and extends the application of mathematical set theory and similarity theory to experimental music. These ideas have been presented at conferences (Polansky and McKinney 1991) (Polansky 1992) and covered more comprehensively in journal publications (Polansky 1996). To summarise the primary aspects; S and T are conceived as ordered sets. Given this representation, various analytical metrics can be applied to obtain some notion of distance between the patterns and, conversely, mutation algorithms can generate music at a specific distance, Ω, between two sets. The various approaches are classified according to their foci and techniques: interval magnitude (difference between one item in the set and the next) or direction (up or down), linear (processing the set from start to finish) or combinatorial (utilising intervals from each item in the set to every other item), unordered (non-structural statistics) or ordered (utilising the sequential order of the pattern). HMSL is unsupported by modern operating systems however much of it has been ported to Java as the Java Music Specification Language (JMSL, below).

MMorph / DMorph

Danny Oppenheim first published a short paper on morphing, presenting the MMorph software (Oppenheim 1995). Since then, it was incorporated as a computer-assisted algorithmic composition tool (CAAC) within DMix, and is discussed in much detail as a patent (Oppenheim 1995). The algorithm is realtime, interactive and deals with n-source morphs. This extends the original definition of the morphing function to include more than two input patterns,

$$f(S_1, S_{\ldots}, S_n, \Omega) = M$$

As a result, the emphasis is also less on automatic transitioning (from source to target) and more on the creation of a musical hybrid.

Oppenheim's general procedure is to group notes from each source together based on some kind of similarity logic. The note properties of all notes in the group are interpolated and this value is used to create a new note. There are two different generic implementation of this procedure: time-warped grouping, creates groups based on the order of the notes, and time synchronous grouping, which creates groups based on the similarity of note-onset.

Musical demos are no longer available from IBM's website, however, Oppenheim is able to send them (music@us.ibm.com).

JMSL

Nick Didkovsky and Phil Burke have extended the capabilities of JMSL beyond the original HMSL (Didkovsky and Burke 2006). Particular aspects of JMSL which are relevant to morphing are the "Binary copy buffer transform" (BCBT) (Didkovsky and Burke 2004) and an applet called the Shubert Impromptu Morpher (Didkovsky 1997). The BCBT is a function that is part of the score editing window in JMSL (Didkovsky and Burke 2004), where the user can copy segments of music into two different buffers. The BCBT function then uses a morphing algorithm to combine the two buffers and paste the result onto the score. In this way, buffer one is S, buffer two is T, $\Omega = 0.5$, and the pasted result is M. The "Zipper interleave transform" is a morphing algorithm that comes with JMSL which iterates through S and T, alternately placing an element from one or the other into M. Through the extensible code design there is great potential for users of JMSL to create custom BCBT plug-ins for the score editor.

Didkovsky's Shubert Impromtu Morpher applet stochastically generates music from statistics obtained by analysing a Schubert performance, as S. T is user-defined values of the statistics. The user controls Ω, and can disable the interpolation of individual statistical parameters.

The Musifier

Jonus Edlund has developed an adaptive music system, the Musifier (Edlund 2004), which utilises compositional morphing as a key component. The Musifier performs n-source morphing on different themes provided by the composer. The intention is for a computer game engine to continually adjust the weight Ω for each theme, based upon the prominence of various game state elements.

The details of the morphing techniques that Edlund uses are secret; however, musical demonstrations are available for download. More recently a web application has been made which allows a user to specify the weights of four different themes (Edlund 2006). A particularly useful advance is apparent simply through listening to the examples. The problem of morphing between parts of different timbre has been adequately handled in MIDI by cross-fading the volume of parts on two different channels and sending identical note events to both channels. To speculate, an abstract harmonic representation may have been implemented to provide unified movement to harmonic parts. Rhythmic segments appear to be treated as indivisible gestalt units.

Edlund uses three criteria for adaptive music and morphing: responsiveness, continuity and complexity. Responsiveness is how well the system responds to change. Continuity is concerned with matching the contour of the changes and changing smoothly. Complexity is how well the algorithm can convert many-dimensional game-state data into equally dimensional musical data and generate suitable music from it.

Beat space

Momeni and Wessel have developed software using MAX/MSP which morphs between parameter states on a 2D surface. Gaussian kernels are used to control the prominence of each parameter state on the surface (Momeni and Wessel 2003). While this software is primarily concerned with morphing sonic parameters, the Beat Space component deal with musical material, morphing between parameters that control probabilities of beat generation within a certain eighth-note slots. S and T are deterministically represented such that the probability for any slot can be only 0 or 1, while M is generated from the non-deterministic interpolations.

LEMu2: Morpheus

I have developed LEMu 2: Morpheus since 2004. As with DMorph, this system is interactive in that the user can select algorithms, control Ω (using MIDI or mouse) and edit S and T in realtime while the morph is running. When the user is not controlling Ω, the transition is executed automatically, such that $\Omega = t$. Unlike DMorph and The Musifier, Morpheus performs 2-source (source, target), rather than n-source morphs.

The first two algorithms developed for this system, weighted-selection and Markov morphing, are already described (Wooller and Brown 2005). Developments since then have been deterministic rather than stochastic: manual note-priority morph and abstraction-interpolation.

Manual note-priority morph allows the composer to manually assign a "priority" rating, from 0 to 1, to individual notes. While there are a number of possible ways an algorithm could use this information, the note-priority morph simply filters notes according their priority and the value of Ω.

Abstraction-interpolation is essentially identical to Mathew's, although it uses a different technique for the mapping of discrete note-onsets into the continuous domain. Segments have a gradient of 0. Notes are created at points where the accumulated area of the function up to the current point equals the current value squared. The tracking of area is more computationally intensive and it has been difficult to obtain rhythmically coherent

results, but this approach has potential for responsiveness to realtime changes and combinations of very different inter-onsets. This software, as well as musical examples, is available for download on the LEMu website (Wooller 2005).

Discussion of techniques

A framework has been developed, whereby particular techniques and systems are conceived and related in terms of function and context. The former is about wether the algorithm abstracts information from the data (analytic), transforms the data content (transformational), or creates raw musical data from abstracted information (generative). The latter indicates the level of contextual information which informs the algorithm. These are continuous descriptors, which allow for the techniques to be positioned on two axes, as in figure 2. Further details and justification of this framework and the associated terminology is available (Wooller, Brown, Miranda, Berry and Diederic 2005).

Within Music IV the music is being analysed to extract continuous functions for each dimension. The interpolation of values from source and target functions is a transformational algorithm and the rendering of the functions into note events is generative. This is a low context process as only the data at any particular time is used; there is no look ahead or contextual memory.

Polansky has explored many techniques for combining sequences of note events to his pieces, notably *51 Melodies*. The techniques used vary in the utilisation of contextual information. "Linear-ordered" algorithms have access to the narrow context of immediate notes as sequences are processed from start to finish. "Combinatorial-unordered" algorithms can incorporate data from the whole sequence. The algorithms use various forms of data other than absolute values, for each dimension of pitch and start-time: relative (intervals), contour (gradient), magnitude, direction. Changing to and from these representations requires some degree of analysis and generation.

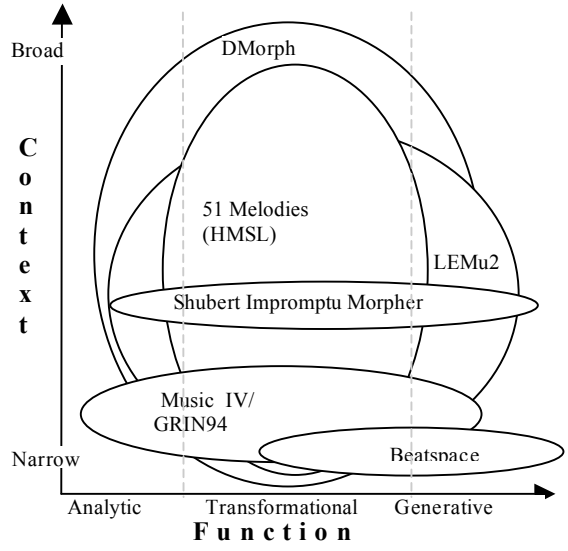

Figure 2. *Morphing systems compared using framework*

The Shubert Impromptu morpher used an analytical process to extract general statistical information from the original piano performance. Information is obtained from the whole piece however much of the temporal-structural dimension is omitted, making for a medium-level of contextual breadth. The interpolation of statistics is transformational, and the stochastic creation of notes from statistics is generative. Beatspace is similarly generative and transformational but without an analytical component or the contextual breadth of rich statistical inputs.

Within DMorph (Oppenheim 1995), the creation of note groupings from each of the input sequences amounts to an analytic process, where higher-level connections between note parameters within the input are found. Motivic analysis is also used to create immutable segments. This provides greater breadth of musical contextual information to the algorithm. The interpolation functions applied to parameters within note groups is transformational and the pairing of interpolated parameters to create notes is generative. DMorph has a number of control parameters which provides additional external context.

The Markov-morph algorithm within LEMu2 analyses the probabilities of note sequences to a user-defined depth.

Applicability of techniques

While there seem to be a number of applications for compositional morphing, it is yet to become widely used. Despite the popularity of computer games, the percentage of CPU allocated to sound and music is minimal, with graphical effects being top-priority (Brown 2006;Edlund 2006). In this way the potential of morphing to be widely adopted within the games industry is stifled because increases in CPU speeds relate mostly to increases in visual effects, rather than interactive musical or sonic complexity.

In live electronic music, morphing has some potential as an alternative to mixing with the ability to transitions from more stylistically divergent pieces of music and to do so in a way that is interesting at a compositional level. This is only really applicable for producer/DJs who intend to perform their own music live or have access to the master files of the songs they are mixing. Perhaps the biggest obstacle with this application is the fact that many people are accustomed to DJ mixing, while the aesthetics of morphing, despite being interesting, are still quite alien. This was discovered through a substantial amount of qualitative feedback from preliminary trials and focus groups on LEMu2. Perhaps it could be overcome through more gradual introduction of morphing techniques.

Computer assisted composition systems that use morphing have a small user base. In the case of the JMSL score editor, this is more likely due to the immaturity of the software. In the case of DMix, it could be lack of publicity, support and availability.

Conclusion

The various systems reviewed comprise the current state of compositional morphing. Despite an array of techniques available, compositional morphing is yet to be

commonly used. When listening to the morphing demos from the different systems it becomes apparent that a number of problems have been overcome, while others remain, being more fundamental to the nature of the task.

Mathews' original system overcame the problem of representing note-onset with a continuous function. The various morphing algorithms in HMSL advanced contemporary procedural music. Oppenheim strove for a more mainstream musical aesthetic and his approach of grouping note events allowed DMorph to easily handle n-sources. Oppenheim experimented with morphing between parts with very contrasting musical roles, illustrating the fundamental problem of morphing music and parts that are extremely different. Edlund eased this problem by sending identical note events to two parts of different timbres and cross-fading them. From LEMu2, the Markov models have been applied with some degree of success when used on melodic parts in particular.

Questions that are yet to be comprehensively investigated are:

- How can structural form be applied within a morph?
- What are useful methods for inter-part communication (Edlund may have begun this)?
- What makes a transition musically "Coherent", even when it is not "Smooth"? For example, suddenly contrasting sections within a morph can sometimes work well musically.
- What approaches are possible, other than layering (eg cross-fade/weighted selection/priority filtering), recombination (HMSL), probabilistic generation, note grouping-interpolation (DMix) and functional abstraction-interpolation (Music IV)?

Acknowledgements

Special thanks to L. Polansky, D. Oppenheim, N. Didkovsky for help with research, Andrew Brown and the ACMC reviewers for comments and John Wiley & Sons for granting permission to reprint.

References

Brown, A. 2006. Pers. Comm. *RE Working with Auran*.

Burke, P. Polansky, L. and Rosenboom, D. 2005. *HMSL, SoftSynth*. http://www.softsynth.com/hmsl/ (7th March, 2006).

Cage, J. 1938. "Metamorphosis for piano I-V". *John Cage: Early Piano Music (CD)*, ECM.

Canazza, S. Poli, G. D. Drioli, C. Roda, A. and Vidolin, A. 2001. "Expressive Morphing for Interactive Performance of Musical Scores" *First International Conference on WEB Delivering of Music (WEDELMUSIC`01)*, IEEE Computer Society.

Didkovsky, N. 1997. *Shubert Impromptu Morpher*. http://www.punosmusic.com/pages/schubert/schubertapplet.html (16th February, 2006).

Didkovsky, N. and Burke, P. 2006. *JMSL overview*, algomus. http://www.algomusic.com/jmsl/ (7th March, 2006).

Didkovsky, N. and Burke, P. 2004. *JMSL Tutorial: JScore Implementing your own Binary Copy Buffer Transform, part 2*. http://www.algomusic.com/jmsl/tutorial/jscoretoot07.html (16th February 2006).

Edlund, J. 2006. "Morphing Demo Server" *InterAmus Music Systems*. http://www.interamus.com (13th April 2006).

Edlund, J. 2006. Pers. Comm. *RE Game Developer's Conference*.

Edlund, J. 2004. "The Virtues of the Musifier: A Matter of View" *Interamus*. http://www.interamus.com/techTalk/musificationAndView.html (16th February 2006).

Glass, P. 2000. "Metamorphosis I-V" *Glass Cage*. Arabesque Recordings.

Hindemith, P. 1989. "Symphonic metamorphosis of themes by Weber" *Hindemith (CD)*. Cleveland, OH, Telarc**:** 8-10.

Mathews, M.V. and Rosler, L. 1969. "Graphical Language for the Scores of Computer-generated Sounds" *Music by Computers*. ed, H. V. Foerster and J. W. Beauchamp. New York: John Wiley and Sons Inc. 84-114.

Momeni, A. and Wessel, D. 2003. "Characterizing and Controlling Musical Material Intuitively with Geometric Models" Proceedings of the *2003 Conference on New Interfaces for Musical Expression NIME2003*. Montreal, Canada**.** 54-62.

Oppenheim, D. 1995. "Interactive system for compositional morphing of music in realtime". USA, International Business Machines Corporation.

Oppenheim, D.V. 1995. "Demonstrating MMorph: A System for Morphing Music in Real-time". *ICMC95 International Computer Music Conference*. Banff , Canada. ICMA**.** 479-480.

Polansky, L. 1991. *51 Melodies*. Artifact Recordings/Frog Peak Music.

Polansky, L. 1996. "Bedhaya Guthrie/Bedhaya Sadra for Voices, Kemanak, Melody Instruments, and Accompanimental Javanese Gamelan" *Perspectives of New Music*. 34. 28-55.

Polansky, L. 1987. "Distance Music I-VI for any number of programmer/performers and live, programmable computer music systems" *Perspectives of New Music*. 25. 537-544.

Polansky, L. 2006. *midifiles.demo.front*, Dartmouth college. http://eamusic.dartmouth.edu/~larry/midifiles.demo.front.html (8th March 2006).

Polansky, L. 1992. "More on Morphological Mutations: Recent Techniques and Developments" *ICMC*. San Jose**.** 57-60.

Polansky, L. 1996. "Morphological Metrics" *Journal of New Music Research*. 25. 289-368.

Polansky, L. 2005. *Morphological Metrics and Mutations FAQ*. http://eamusic.dartmouth.edu/~larry/mutationsFAQ.html (15th February, 2006).

Polansky, L. and Erbe, T. 1996. "Spectral Mutation in Soundhack" *Computer Music Journal*. 20. 92-101.

Polansky, L. and McKinney, M. 1991. "Morphological Mutation Functions: Applications to Motivic Transformations and to a New Class of Cross-Synthesis Techniques" *ICMC*. Montreal.

Rosenboom, D. 1982. "The Qualities of Change: "On Being Invisible: Steps Towards Transitional To-

pologies of Musical Form". Oakland, CA: Mills College.
Tenney, J. and Polansky, L. 1979. "Temporal Gestalt Perception in Music" *Journal of Music Theory*. 24. 205-241.
Wooller, R. 2005. *Morph Music Demos*. Queensland University of Technology. http://www.lemu.org/download.html#l2sw (4th November 2005).
Wooller, R. and Brown, A. 2005. "Investigating morphing algorithms for generative music" *Third Iteration*. Monash University, Melbourne.
Wooller, R. Brown, A.R. Miranda, E.R. Berry, R. and Diederic, J. 2005. "A framework for comparing algorithmic music systems (in press)" *Symposium on Generative Arts Practice (GAP)*. University of Western Sydney.