TIMEWARP: A GRAPHICAL TOOL FOR THE CONTROL OF POLYPHONIC SMOOTHLY VARYING TEMPOS

John MacCallum, Andrew Schmeder

Center for New Music and Audio Technologies
Department of Music
University of California, Berkeley
{john,andy}@cnmat.berkeley.edu

ABSTRACT

We present a parametric method for the variable control of tempo with specification of shape and phase alignment constraints, and its implementation as a graphical interface written in C for the Max/MSP environment. This tool aids in the construction and management of polyphonic streams of independent, fixed or smoothly-varying tempos suitable for live performance or computer-generated scores. We also present a brief history of how polyphonic tempo manipulation has been used in context and compare our work with two recent projects to implement similar tools.

1. INTRODUCTION

Since the beginning of the 20th century, much compositional energy has been devoted to the exploration of time-varying tempos and compositions where multiple players perform in different, but constant tempos. As early as 1930, Henry Cowell recognized that technology could come to the aid of the composer/performer who wished to explore rhythmic complexity when he commissioned Léon Theremin to construct the *Rhythmicon* or *Polyrhythmophone*. Notable attempts to explore this world without the aid of technology include Iannis Xenakis' *Pléïades* (1979) and Gérard Grisey's *Tempus ex Machina* (1979) both of which require that the individual performers play in slightly different tempos (as close as two metronome clicks apart!) so that their parts gradually "slide" apart[1].

Another notable work is the third movement of György Ligeti's *Kammerkonzert* (1969–70) in which the conductor beats a different tempo for each player and leaves them to continue playing at that speed. The resulting rhythmic texture is of course completely dependent on the ability of the conductor to conjure up the correct tempos and that of each player to continue in the given tempo without deviation, indifferent to the rhythmic chaos surrounding them. In this case, there is much that the computer could do to aid in the performance of the *Kammerkonzert* depending on the goals of the ensemble. One would be right to question the appropriate level of accuracy—is the piece about machine-like precision, or man's struggle to achieve it? Individual click-tracks for the ensemble and conductor, all set to begin at precisely the same time and hold their tempos with machine-precision, would satisfy the former, while a single click-track in the ear of the conductor that would not only change tempo, but could perhaps prepare the conductor with subdivisions that reflect the coming tempo would leave the matter of precision in the hands of the performers who would be left to struggle to maintain their tempos.

The tempo canons of Nancarrow beg similar questions of intent—one should be tempted to ask if Nancarrow would have preferred the accuracy afforded by the computer, or whether the slight imperfections in his piano roll would have suited him better. In either case, the tool that we present in this paper makes the construction of such canons trivial.

Edmund Campion's *ADKOM (A Different Kind of Measure)* [2] (2001–present, commissioned by the Drumming Ensemble of Porto, Portugal) for percussion quartet, developed at (CNMAT) with the aid of Musical Systems Designer Matthew Wright, requires a separate click-track for each performer. The tempos in these click-tracks vary over the course of the piece, at times resulting in extremely complex polyrhythms due to their divergence, and at other times causing the performers to come into phase with incredible accuracy that is all but impossible to achieve without the aid of technology.

Finally, John MacCallum's ... *almost like hail* ... for solo percussion and live electronics contains tempo maps realized by the computer that far exceed the limits of human performability. These are heard as the performer struggles to achieve machine-like precision while following his own tempo map.

[1] It should be noted that metronomes now often used to aid in the performance of both Xenakis' *Pléïades* and Grisey's *Tempus ex Machina*.

[2] http://cnmat.berkeley.edu/node/7713

2. BACKGROUND AND MOTIVATION

The motivation of this work is to provide musicians with a tool that will allow them to create and perform music in otherwise impossibly complex and dynamically changing metronomic relationships to one another. Additionally, we wish to be able to control the rhythmic phase relationship between performers at arbitrary points in time (e.g., we may want to specify that two performers will be an 1/8th or a 1/16th note apart when they arrive at a given tempo—see figure 4). Although it is simple enough to construct click-tracks that will gradually change tempo, it is a non-trivial problem to bring them into some arbitrary polyrhythmic relationship at any given point in time. There are many ways to approach this problem; in this section, we critique two recent projects as a way to develop a set of constraints that we will impose on our implementation.

2.1. Tempocurver

In order to arrive at the desired conditions at the end of a segment of a tempo map, the *tempocurver*, written by Matthew Wright at CNMAT as part of Edmund Campion's *ADKOM* project, holds the starting tempo until enough unwrapped phase has accumulated that a linear tempo function will satisfy all three ending parameters. This can, depending on the requested values, produce a delay between the requested and actual start time that is unacceptably long.

The *tempocurver* operates on a queue of time, tempo, wrapped phase destinations and maintains its state as it works through its queue. This means that in order to start at some arbitrary time in the middle of the map, e.g. for the purposes of rehearsal, the state at that point must be computed from the beginning. That may not be practical given a very large composition with many voices. The queue system also becomes problematic when one wishes to scale an entire tempo map. In this case, one would have to operate on the queue itself which is not currently possible, or edit the data before it is input into the *tempocurver*. The ability to scale the tempo map, either globally or locally is critical when working with performers who may want to work through a section at a slower pace for the purposes of rehearsal.

2.2. Timegrid

Schacher and Neukom's graphical application *timegrid* [3] overcomes many of these problems, but a few remain. *timegrid* allows for the synchronization of multiple voices at certain points in time, but does not allow the user to specify the rhythmic relationship between these two voices. For example, we may want two voices to arrive at the same tempo at the same time, but for one to be exactly half a beat ahead or behind the other. Additionally, the composer has no choice over the type of curve when editing in this mode—the user enters points on a grid that represent frequency at a given time and the algorithm finds a suitable curve that will pass through all of them. In all other modes, the user must allow one degree of freedom.

In the mode where the user may draw arbitrary tempo functions, *timegrid* does not enforce the use of well-defined functions, e.g. the user may draw a function that folds back on itself. Although a loop that is created in *timegrid* does not actually cause time to fold back on itself, we believe it creates confusion where there should be none.

2.3. Goals and Constraints

Polyphonic The tool must be able to support an arbitrary number of voices that can change tempo smoothly and independently. We must also be able to specify the polyrhythmic relationship between any of these voices at any given time.

Accuracy at all requested points The user-requested state at a point must be accurate, which is to say that the user should not have to allow a parameter (time, tempo, position within a beat) to vary for the sake of the algorithm. The only degree of freedom we are left to manipulate then is the shape of the curve itself.

Well-Defined Semantics The system must be consistent both mathematically and with respect to the user interface. The tempo is strictly positive and the phase accumulation is monotonic when time is moving forward.

Possibly discontinuous It can be useful to introduce a discontinuity in either the tempo or beat functions. As a trivial example, consider a series of steady quarter-note beats. We may wish to shift the position of these beats by half of a beat (or some other quantity) *without* a smooth transition.

Robust We must also allow the user the flexibility to shape the path through time without having to consider the underlying algorithm. To this end, the algorithm must never fail to produce a result even in the case of the most extreme tempo changes over short durations.

Stateless The tool should be a stateless transfer function that will permit us to move through the time map at any global rate we choose. Additionally, we require the ability to jump to arbitrary points in time.

Signal Domain The output of the system should have arbitrary accuracy and precision with respect to time enabling sample-accurate calculations in the signal domain.

Functional Composition The output of one temporal transformation can be used as the input to any other transformation.

Jaffe [2] and later, Wessel et al. [4], proposed a monotonic transfer function that could be used to warp "score" or "clock" time into "performance" time which is not dissimilar to the tool that we present here. It should also be noted in passing that this concept of a two dimensional representation of "score" time versus "performed" time can be of immense value as an analytical tool for quantifying musical phrasing. For a thorough review of such topics, see Desain and Honing [1].

3. METHODS

3.1. Definitions

Let T denote a tempo as the number of beats in a second (also called frequency). The instantaneous tempo $T(t)$ is equal to the derivative of the phase, $p(t)$.

$$T(t) = \lim_{\triangle t \to 0} \frac{p(t+\triangle t) - p(t)}{\triangle t} \tag{1}$$

We say that a beat occurs when the wrapped phase is equal to 0

$$b(t) = \delta(p(t) - \lfloor p(t) \rfloor) \tag{2}$$

where δ is the Dirac-delta function. (See Fig. 1)

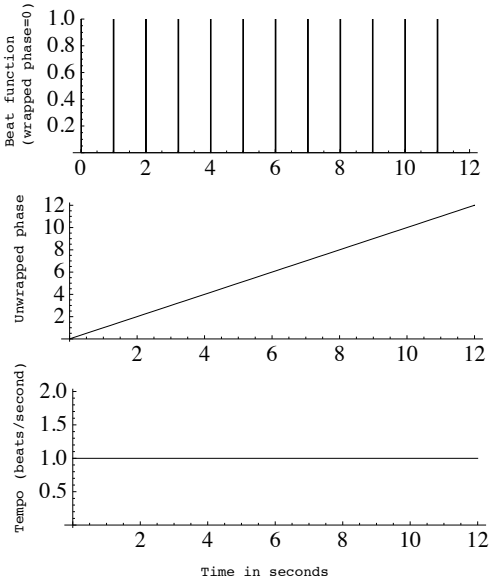

Figure 1. Beat, phase, and tempo functions for a fixed tempo of one beat per second.

3.2. Tempo Warping

A tempo warping function, parameterized by $w = (T_s, T_e, t_s, t_e, \alpha, \beta)$, changing from tempo T_s to T_e over the time interval $[t_s, t_e]$ with shape parameters α and β is

$$T_w(t) = (T_e - T_s) g_w(t) + T_s \tag{3}$$

$$g_w(t) = \begin{cases} 0 & t < t_s \\ I(\frac{t-t_s}{t_e-t_s}, \alpha, \beta) & t_s \le t \le t_e \\ 1 & t > t_e \end{cases} \tag{4}$$

where $I(t, \alpha, \beta)$ is the regularized beta function

$$I(t, \alpha, \beta) = \frac{\int_0^t u^{(\alpha-1)}(1-u)^{(\beta-1)} du}{\int_0^1 u^{(\alpha-1)}(1-u)^{(\beta-1)} du} \tag{5}$$

and α and β must be greater than 0. Note that $g_w(t)$ is monotonic and bounded between $[0, 1]$.

The corresponding warped unwrapped phase function $p_w(t)$ is

$$p_w(t) = \int_{-\infty}^{t} T_w(t) dt \tag{6}$$

3.3. Tempo Warping With Phase Alignment

Suppose that we want to specify the value of the wrapped phase to be θ at time t_e.

$$(p_w(t_e) - \lfloor p_w(t_e) \rfloor) = \theta \subseteq [0, 1] \tag{7}$$

We then define an error quantity ε' that is the amount by which the function over- or undershot the requested phase.

$$\varepsilon' = \theta - (p_w(t_e) - \lfloor p_w(t_e) \rfloor) \tag{8}$$

To minimize the magnitude of error correction, we select the alternative having least absolute value of

$$\varepsilon = \{\varepsilon', \varepsilon' + 1, \varepsilon' - 1\} \tag{9}$$

so that $-0.5 \le \varepsilon \le 0.5$.

To realize the phase alignment, a secondary warping function is applied over the time interval $[t_{e1}, t_{e2}] \subseteq [t_s, t_e]$ with shape parameters α_e and β_e to account for the value of ε. The warped, unwrapped aligned phase function, parameterized by $w' = (T_s, T_e, t_s, t_e, \alpha, \beta, \theta, t_{e1}, t_{e2}, \alpha_e, \beta_e)$, is

$$p_{w'}(t) = \begin{cases} p_w(t) & t_0 \le t \le t_{e1} \\ p_w(t) + \varepsilon g_{(t_{e1}, t_{e2}, \alpha_e, \beta_e)}(t) & t_{e1} \le t \le t_{e2} \\ p_w(t) + \varepsilon & t_{e2} \le t \le t_1 \end{cases} \tag{10}$$

4. IMPLEMENTATION

4.1. Interface

This tool has been implemented as a graphical user interface object for the Max/MSP environment that operates on and outputs a phase signal. Points can be entered into the space by clicking and dragging them around. Lines representing a linear change in tempo automatically connect two adjacent points, and beats are shown as vertical lines that extend from the tempo line to the bottom of the window. Each point is made up of two concentric circles, with the inner circle and its position along the y-axis representing the arrival tempo, and the outer the departure tempo. Arrival and departure phase are represented by two green triangles with phase being incremented as the triangles move clockwise around the circles. These parameters, and all others (the values of α and β , the error correction region, etc.) can be entered numerically or interactively with a pointer.

4.2. Error Visualization

With our object, one can visualize the error correction requirement to realize a phase alignment constraint in two different ways. The message "show_tempo_correction" will

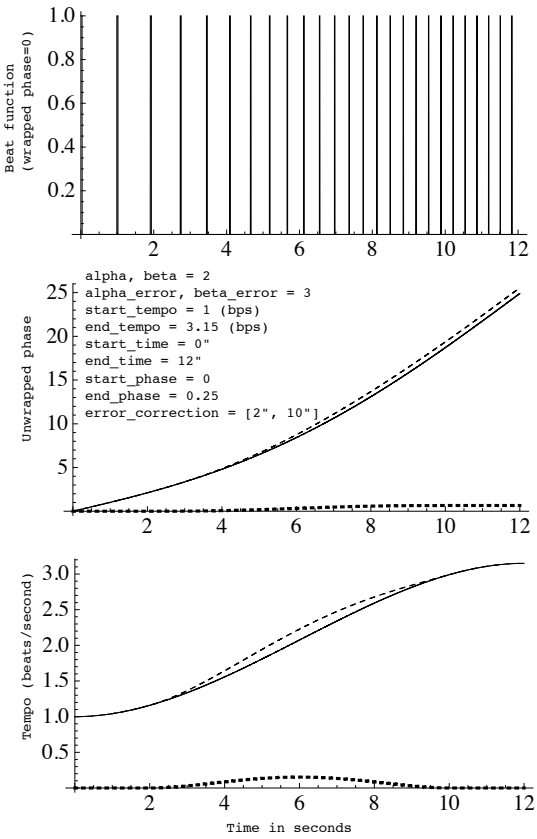

Figure 2. Beat, phase, and tempo functions (solid = unaligned, dashed = aligned, dotted = difference).

show the error correction region as a highlighted area that can be adjusted by the user, as well as the warped curve superimposed over the uncorrected curve. The message "show_beat_correction" will display a second set of beats (vertical lines) reflecting their corrected positions in time. Additionally, the amount of error being compensated for is output numerically and can be used to minimize the warping of the curve, i.e. the user could search for a set of conditions that would result in 0 error.

4.3. Polyphony

An arbitrary number of voices may be created on the fly, but since the number of outlets a Max/MSP object has is fixed at instantiation time, we use a simple bussing scheme via a second object called *tw_bus~*. A simple two voice polyphonic tempo map can be seen in figure 4.

5. CONCLUSION AND FUTURE WORK

The tool presented here facilitates the compositional exploration of extremely complex rhythmic textures and structures in a way that has previously been unavailable. In the near term, work will focus on use of the tool in a compositional context and especially on the problem of how best to

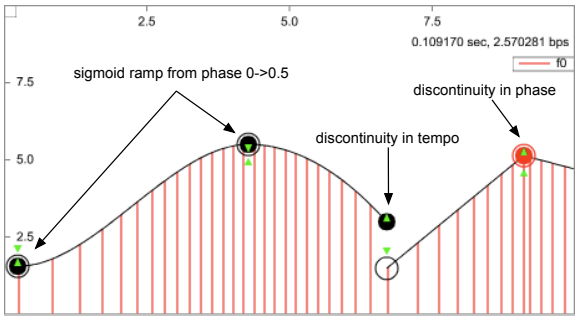

Figure 3. Tempo editor interface

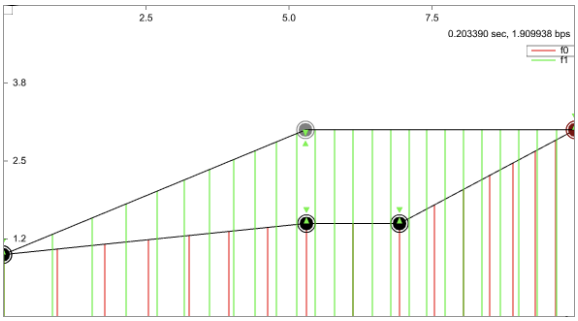

Figure 4. A simple polyphonic tempo map showing two points of synchronization.

notate music produced with such a tool. We recognize that the challenges inherent in the performance of this type of music may well demand different notational strategies.

As discussed at the beginning of this paper, the question of machine-precision is an interesting one, and is not always musically desirable. With that in mind, we plan to implement a microtiming model that can be applied to the output of the object.

Finally, this tool should facilitate more exploration into the way that we perceive rhythm and tempo.

6. REFERENCES

[1] P. Desain and H. Honing, "Tempo curves considered harmful," *Contemporary Music Review*, vol. 7, no. 2, pp. 123–138, 1993.

[2] D. Jaffe, "Ensemble timing in computer music," *Computer Music Journal*, vol. 9, no. 4, pp. 38–48, 1985.

[3] J. C. Schacher and M. Neukom, "Where's the beat? tools for dynamic tempo calculations," in *Proceedings of the International Computer Music Conference*, 2007.

[4] D. Wessel, D. Bristow, and Z. Settel, "Control of phrasing and articulation in synthesis," in *Proceedings of the International Computer Music Conference*, 1987.