Center for Computer Research in Music and Acoustics October 1984

Department of Music
Report No. STAN-M-23

ENSEMBLE TIMING IN COMPUTER MUSIC

by

David A. Jaffe

Computer music systems have tended to deal in a clumsy manner with both inter-and intra-voice timing. A solution called a "time-map" is proposed. It provides a simple yet general formalism for describing ensemble timing interaction.

This paper is contained in the Proceedings of the International Computer Music Conference (ICMC) held at IRCAM in Paris, France, October 1984.

This research was supported by the System Development Foundation under Grant SDF #345. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Stanford University, any agency of the U. S. Government, or of sponsoring foundations.

/

Introduction

The use of computers in music allows for a wide variety of musical timing interactions which would be impossible or extremely difficult with human performers alone. However, computer music systems have tended to neglect or deal in a clumsy manner with both inter- and intra-voice timing and phrasing, often resulting in mechanical-sounding results. Attempts at relieving this oppresive rigidity using random jitter have proven inadequate and difficult to control. Typically, the more freedom given to individual parts, the more difficult the task of constraining coordination between parts. The solution to this problem is the "time-map", a simple yet general formalism for describing ensemble timing interaction. This technique provides a means of gaining precise, yet flexible phrasing control.

Ensemble Timing in Instrumental Music

As a point of departure, consider the ensemble timing interaction of a traditional string quartet. The quartet as a whole plays what is on the printed page with group tempo and phrasing adjustments. At the same time, each player occasionaly deviates in an individual manner from that trajectory. The performers, using the feedback they receive from listening to each other, adjust their individual timing to coincide precisely at important reference points. These points may appear on every beat as in strictly pulse-oriented music, every phrase as in music with more *rubato*, or they may appear only as occasional cues without reference to a basic pulse as in some modern pieces. The resultant timing, which will be referred to as "clock time" is quite different from the view of time as represented on the page of a given instrumentalist, which will be referred to as "basic time" with respect to that instrumentalist.

Ensemble Timing in Computer Music

It would be desirable to have a computer music timing system capable not only of matching the richness of instrumental timing but also of opening up new possibilities of ensemble relationships.

This suggests the term "ensemble timing" be interpreted broadly to mean *an interaction between two or more "voices", each of which may have its own view of time.* A *voice* is simply a process which produces a series of events. Events may be anything from individual notes to complex sounds of orchestral dimensions and may be composed of sub-events. Even a single voice playing with *rubato* can be considered to be participating in an ensemble timing situation in the sense that it is interacting with an implied pulsation.

Ensemble timing consists of two basic elements, *coordination at prescribed points of reference* and *prescribed independence between these points.*

Approaches to Timing

Before proceeding to a description of the time-map, several other common timing approaches will be examined. Each of these falls short, to some extent, in providing the desired degree of timing control.

Clock Time

The most basic approach to timing is simply to use clock time as the basic time for all voices. This method is cumbersome and inflexible at best. Music in which rhythm is clearly delineated seems to suffer the most from this approach. The result is mechanistic and rigid, similar to the timing of an instrumentalist rehearsing with a metronome. Such computer music has been compared to a "first reading" of an instrumental piece [Pierce 1984].

Tempo Perturbation

The next level of complexity involves introducing perturbation such as random numbers to the event durations. This helps break up the oppressive rigidity of the clock time approach. However, the increase in linear freedom is provided at the expense of precise control over vertical alignment. Typically, voices tend to gradually wander apart as the piece progresses. Furthermore, undesired overlaps may result when the tempo is increasing and gaps may result when the tempo is decreasing. This occurs even if the "perterbation" is simply a steady tempo change applied to a basic rhythm. Figure 1 shows a time-line graphic representation which illustrates the distortion in vertical alignment produced using this algorithm in such a case. The crux of the problem is the absence of any reference point and the fact that duration is computed as if the tempo were remaining constant for the duration of the event.

Running Onset Tally

These problems can be solved by keeping a running tally of the event onset-times with respect to the voice and perturbing always in reference to that quantity so that perterbation error does not accumulate. This is much closer to the way human musicians play. They do not "accumulate" error. Rather, they are always compensating for their "mistakes".

The seemingly innocent change of emphasis from *tempo*, the distance in time between events, to *onset-times*, the time at which events occur is actually of significant importance. This process is equivalent to integrating the tempo and suggests a more flexible method, which we call the *time-map*.

time-map

The time-map is defined as a continuous strictly increasing function, defined over the duration of the piece, which maps from one time scale to another time scale.* When the same map is applied to a group of voices, it is assured that those voices will stay perfectly synchronized. Figure 2a shows a sample time-map. Figure 2b is a piece of pseudo-ALGOL code illustrating how a time-map is accessed.

Thinking in instrumental terms, the time-map can be interpreted as a representation of the "warping" which occurs when an instrumentalist or conductor interprets a piece. The pre-deviated time in this case represents time as shown on the score before tempo and expression are added and the post-deviated time represents the actual onset-times of the notes played by the performer.

Advantages of time-maps

The time-map provides the composer with a means of constraining precisely where vertical alignment will occur. In addition, many independently warped time trajectories can be handled by constructing a different time-map for each voice. Wherever two maps intersect in pre-mapped time, there will be a vertical coincidence at that point in post-mapped time, providing there is an event in each voice at that point.

The time-map has several other advantages over other methods. Look-ahead is easy to do because the calculation does not depend on previously computed values. It is therefore possible to look into the future and find out what the time warping will be when an event finishes. Thus, unwanted gaps and overlaps can be avoided, even when the tempo is fluctuating wildly. Also, it easy to see what is going on with the tempo. The tempo is simply a function of time attained by differentiating the time-map, segment by segment. Finally, with the extension of function composition described below, the time-map method can be used to arrange time trajectories hierarchically thus making it possible to implement sophisticated ensemble timing situations.

What Type of Map?

The first time-maps tried were comprised entirely of connected line segments as shown in Figure 3. Such maps work well for examples in which only discrete tempo changes are required. But attempts at continuous tempo change using line segments tend to sound unnatural. This is because the tempo has "glitches" which result from the abrupt changes of tempo from one line segment to another. Although it is possible to contrive the effect of continuously changing tempo by constructing a time-map comprised of one segment per

* Discontinuos functions are invalid because they represent a jump ahead in time; decreasing functions are invalid because they represent a time reversal.

event, the very fact that this is necessary suggests that the representation is sub-optimal. The solution is to use a combination of linear and non-linear functions to comprise the map. Any such map may be used as long as it meets the restrictions given under the defintion of the time-map.

Ways Of Creating Maps

The most straightforward method for constructing a time-map is for the composer to specify it directly. However, musicians often prefer to think in terms of tempo rather than in terms of time-maps. Therefore, a front-end to the process can be added which allows the composer to specify a tempo function in terms of line segments. From this tempo function, the corresponding time-map can then be computed. In the tempo function, a segment with slope of zero represents a constant tempo and a segment with a non-zero slope represents a steadily changing tempo. The only restrictions on the tempo function is that it be strictly positive and defined over the interval of the piece. It need not be continuous, since tempo can change abruptly. The corresponding time-map is then derived by integrating this function, one segment at a time, producing a map comprised of connected quadratic segments. Such a parabolic map and the line segment tempo function from which it is derived are shown in Figure 4.

For specifying tempi which vary at non-constant rates, higher order polynomials could be specified in the tempo function. Of course, it is also possible to use functions other than polynomials, but sticking to polynomials insures that the integration will be easy to do.

Another means of specifying time-maps is to "perform" the clock time rhythm, tell the computer what the rhythm is in basic time, and then have the computer correlate the two and construct the map [Smith 1970]. This has significant implications for live performance via the inverse time-map which is discussed below.

Combining Maps

The string quartet ensemble situation described above, in which both individual and group phrasing and tempo fluctuation are present, can be generalized into the model of combining any number of time- maps. One of the most attractive methods of combining time-maps is the technique of "function composition". Let $g(t)$ and $f(t)$ be two maps, with g defined over the interval $[f(0), f(T)]$ and T equal to the duration of the piece. Then the composite function is defined as $g(f(t))$, where t is basic time. The tempo at time t (where it exists)* is the derivative of this function, given by the chain rule for differentiation:

$$[g(f(t))]' = g'(f(t))f'(t)$$

By the definition of a time-map, $g(t)$ and $f(t)$ are strictly positive and continuous. This implies $g(f(t))$ is also strictly positive and continuous. Thus the technique of function

* The tempo can not be said to exist at the instant it changes abruptly.

composition has the attractive property that ***the composite of two time-maps is another time-map.***

Implementation Constraints

The composite map can be computed in advance or it can be computed at execution time. Computing in advance may be more efficient for real-time uses. On the other hand, it is not always easy to do if the component time- maps consist of something other than first order splines because the order of the composite is the product of the orders of the constituents. Therefore, if real-time constraints will allow, it is preferable to do the function composition at execution time. Another advantage is that the knowledge of the constituents is preserved and is available for debugging and editing. This works especially nicely in a language such as LISP where function calls can be teated as data and an arbitrary map can be applied at run-time without precompilation.

Recursive Scheduler for Hierarchical Timing Control

Hierarchical control of time-maps provides a powerful domain in which to implement complex dependencies and explore new relationships in ensemble timing. One way of implementing hierarchical control is to use a tree-structured scheduler comprised of time-maps, voices and "mergers". Terminal nodes of the tree are voices which produce events. Non-terminal nodes are mergers. Each merger handles one or more event streams, each with its own time-map (which may be the same as the map of its sibling). Precisely, a merger works as follows: When it receives an event message from one of its children, it performs the appropriate time-mapping on that event and inserts it into a cue sorted by post-mapped onset time. It then sends the first event on the cue to the parent node and sends a message to the child from which that event came, asking for its next event. If there is no parent, the event is sent to the output object.

This implementation has the advantage that knowledge is modularized and local to the object to which it applies, and a merger or voice need know nothing about any maps which appear above it in the tree. Figure 5 shows an example tree in which four voices are mapped through four maps. Voices 2 and 3 (mapped by map J) are merged by merger X. This merged stream (mapped by map G) is then merged by merger Y with two other streams: voice 1 (mapped by map F) and voice 4 (mapped by maps K and H).

An Important Special Case - Local Warping

The Harvard Dictionary of Music [Apel 1969] defines the term *Rubato* as referring to two distinctly different types of tempo fluctuation. One is simply a free approach to tempo, traditionally applied to the entire musical texture. The other involves a "give-and-take" principal, in which the musician makes up for any lost time, and is traditionally applied to the melody independent of the accompaniement.

A special class of composite transformations has this "give-and-take" property. This class consists of transformations which leave the endpoints invariant. This allows segments of time to be locally warped without affecting neighboring segments.

A convenient implementation uses a library of normalized transformations. Each transformation is defined in the interval $[0, 1]$ with $f(0) = 0$ and $f(1) = 1$. The tranformation can then be applied over any $[a, b]$ by applying it with the appropriate scaling and offset. As long as the normalized transformation is a valid map, any application of that transformation will also be a valid map. It is hoped that further research into performance practice will provide prototypical examples of natural sounding transformations which can then be incorportated into the library.

Local transformations can also be used in a larger, compositional sense. For exmaple, consider the transformation $f(t) = t + sin(tb)a$, where a and b are parameters and t is time. The two parameters must be chosen with care; for some values f is not a valid time-map. This transformation makes up for lost time in any period of time for which tb goes through a multiple of π. In the composition *Silicon Valley Breakdown* for four-channel computer-generated tape written by the author in 1982, this transformation was used to create elastic cannons which coincide at prescribed times and "chase each other" in between these times. Each cannon is repeated a number of times using different values of a, which controls the amount of deviation from unison, and b, which controls how fast the sinusoidal ensemble timing deviation occurs. In addition, a phase offset is added to the sine calculation, providing control over where in the course of the cannon the unison relationship will occur.

Coordination with Live Performers

Sophisticated ensemble interaction between live performers and computers is a goal whose difficulties and potential rewards are great. Although an in-depth treatment of the subject is beyond the scope of this paper, an overview will be presented and the role of the time- map in this context will be described.

Most traditional approaches to computer/performer interaction fall into one of three categories: instrumentalists playing with a tape, passive electronic processing and keyboard performance. While interesting pieces have been created in each of these domains, in none does the computer participate as a true "member of the ensemble". Instead, it behaves like either the most egotistical soloist, oblivous of the other members of the ensemble, or like the most passive accompanist, never contributing anything of its own.

New software and hardware developments are making it possible for the computer to behave intelligently: to *hear, understand* and *respond*. At the very least, it would be desirable for the computer to be able to follow the tempo of the performers. To do this, the time-map again becomes useful, this time in its inverse form.

Inverse Maps - Extracting Warpings

The inverse map represents a description of the score as a function of the performance. Once the computer discovers the map, it can follow, respond, etc. The problem of how to determine the map is, however, non-trivial. The task is greatly simplified if the computer already knows the score. Yet, even in this case, a variety of complications arise. These include the possibility of errors in both signal-processing and performance. Also, since the computer is constrained to respond in real-time, it can not "wait and see what will happen", i.e. there can be no look-ahead into the future of the performance. Researchers who have been working in this area include Dannenberg and Vercoe [Dannenberg, 1984; Vercoe, 1984].

A more difficult problem is that of extracting the score from a live performance. This involves simultaneously discovering the score and constructing the map. One group doing such research is the automatic transcription project being conducted at CCRMA, Stanford. The goal of this project is nothing short of computer understanding from audio signals with no prior knowledge of the score [Foster et. al., 1982; Chafe et. al., 1982].

Conclusion

The time-map and inverse time-map provide links between the worlds of warped time and clock time and between the worlds of human and computer performance. When coupled with function composition, the time-map approach provides a powerful formalism for dealing with problems of warped versus clock time. In addition, it provides for a system of hierarchical time warping. The technique is a practical and efficient method useful both in simulating the effects of live performance situations and in devising new compositional concepts of ensemble timing.

Acknowledgements

Thanks to Johannes Goebel, Marc LeBrun, Bernard Mont-Reynaud, Andy Schloss, Bill Schottsaedt, Bob Shannon, and Julius Smith for input on this paper.

References

Apel, Willi, eds. 1969. *Harvard Dictionary of Music.* Cambridge, Mass: Belknap Press.

Chafe, C., B. Mont-Reynaud, and L. Rush. 1982. "Toward an Intelligent Editor for Digital Audio: Recognition of Musical Constructs." *Computer Music Journal* 6(1):30-41.

Dannenberg, Roger B. "An On-Line Algorithm for Real-Time Accompaniment". Paper presented at the International Computer Music Conference. IRCAM, Oct., 1984.

Foster, S., W. A. Schloss, and A. J. Rockmore. 1982. "Toward an Intelligent Editor of Digital Audio: Signal Processing Methods." *Computer Music Journal* 6(1): 7-17.

Pierce, John. 1984. Private communication.

Smith, Leland. 1970. "The Humanization of Computer Music". Paper presented at a confernce of the Audio Engineering Society, Atlantic City, N.J. 1970.

Vercoe, Barry. "The Synthetic Performer in the Context of Live Performance". Paper presented at the International Computer Music Conference. IRCAM, Oct., 1984.

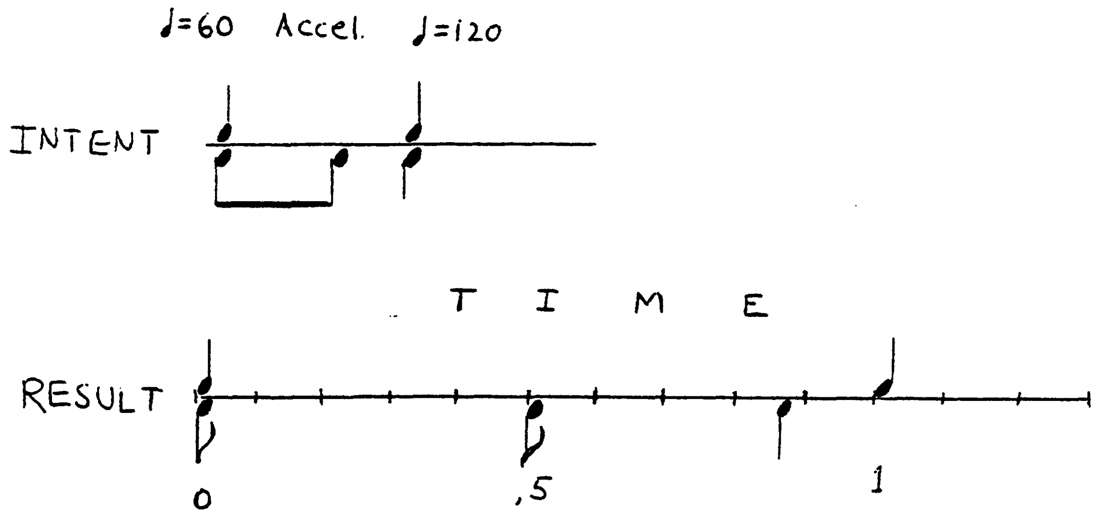

Figure 1 - A time-line graphic representation of the vertical alignment distortion resulting from tempo perturbation. Also shown is a representation in musical notation of the intended effect.

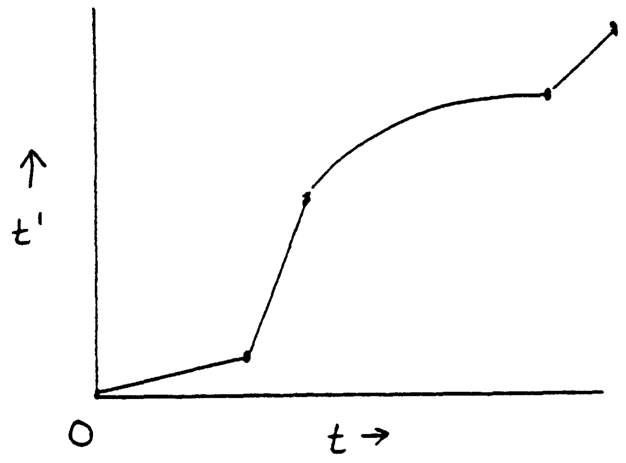

Figure 2a - A sample time map.

```
WHILE onset < total_duration DO
     BEGIN
     onset ← new_onset;
     duration ← Time_Map (onset + basic_duration) - onset;
     new_onset ← onset + duration;
     END;
```

Figure 2b - Pseudo-ALGOL code illustrating how a time map is accessed.

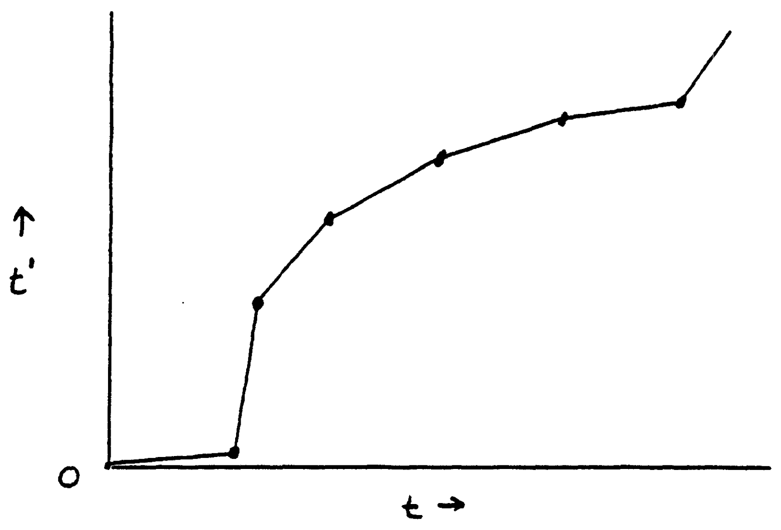

Figure 3 - A map comprised entirely of line segments. Abrupt changes in slope where segments meet represent abrupt changes in tempo.

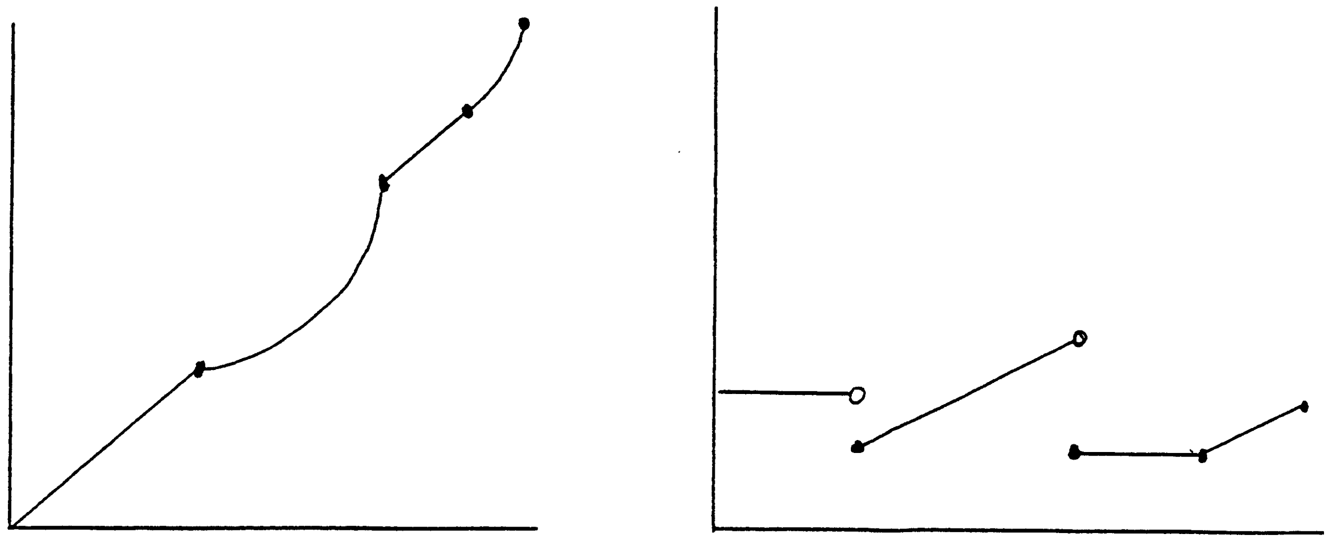

Figure 4 - A quadratic time map (left) and the line segment tempo function from which it was derived (right).

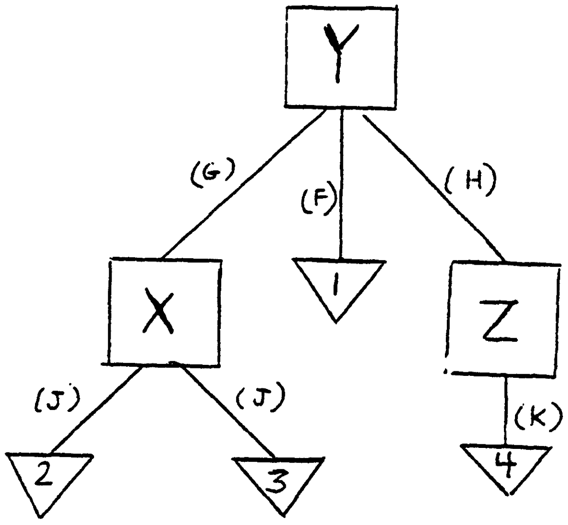

Figure 5 - An example tree scheduler in which four voices are mapped through four maps. Squares are mergers. Triangles are voices. Time mappings are shown in parenthesis.