To the Graduate Council:

I am submitting herewith a thesis written by Andrew Horner entitled "Fractal Timing in Computer Music: A MIDI Implementation." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

David Straight, Major Professor

We have read this thesis
and recommend its acceptance:

David C. Mutchler

Donald Pederson

Accepted for the Council:

Vice Provost
and Dean of the Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature Andrew Horne

Date July 20, 1989

FRACTAL TIMING

IN COMPUTER MUSIC:

A MIDI IMPLEMENTATION

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Andrew Horner

August 1989

ACKNOWLEDGMENTS

A number of scholars and friends have assisted me generously in the development of this paper. In particular, I am indebted to Professors David Straight, David Mutchler, and Donald Peterson, the members of my committee. Their support and input made possible this inherently interdisciplinary study, and is deeply appreciated.

ABSTRACT

Human musical performance is filled with subtle nuances and intricate timing relationships, which makes its simulation a complex task for the computer. Various efforts in realizing dynamic tempos and phrasings have applied a process, termed "time warping", utilizing simple linear or polynomial functions. This paper introduces the use of fractal functions in this time warping process.

By utilizing the Macintosh computer and a specialized file format, standard among MIDI software and hardware, a time warping application was developed which included both conventional functions and those generated by a fractal process. With this implementation in place, examples were tried, and the results gathered.

These results demonstrated that the fractal process was effective in generating aesthetically pleasing musical transitions. In addition, the fractal results were wide-ranging in character and thus suited to various musical contexts. The author's composition, Clockwork, provided an opportunity to use the process in a real musical piece. Thus, fractal warping was determined to be effective in the creation of computer music nuances and phrasings.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION AND MUSICAL PERSPECTIVE

Musical performance practice, which has evolved continuously through the centuries, constitutes a largely unwritten source of information of an often subtle nature. Among the areas it encompasses is the knowledge of musical time. Teachers spend years impressing their students with the "feel" of music: when to push ahead or hold back in a composition, and how to push ahead or hold back. Composers traditionally make such an indication with few or no words in the musical score, trusting the musical instinct of their performers and the musical precedent of their training.

Some modern composers, however, deliberately attempt to override traditional musical conditioning by specifying every subtle nuance for each note of the piece. This is an approach necessary with computers, but is invariably demanding on an instrumentalist; it is very difficult for a performer to play no more or no less than what is indicated. Other composers, such as Steve Reich, have investigated new timing relationships in their specification of long-term divergence and re-convergence of intensely pulse-oriented music. These too, are often very difficult pieces to perform well, and require a retraining of the performers involved.

When performers play together, they typically follow the same musical pulse, yet their independent lines exactly

correspond only at key moments in the music. These moments may occur every beat, at the start of a new phrase, or only at very special moments in some modern compositions. The coordinated deviations between such moments add a rich and animated character to the music, which is lost when a computer vertically aligns all the notes in time. It is hard to imagine Haydn's Clock Symphony sounding very interesting if it were in fact, played with each subdivision perfectly even. It is even harder to imagine music such as free-flowing jazz having any life at all under these circumstances.

Thus, an interesting area of computer music research has been the "humanization" of musical time. The integration of the musical keyboard into computer music, and the development of other "real-time" electronic musical instruments, has contributed to the realization of this ideal. However, some music is not instrumental in nature, and in other cases it simply may not be realizable in real-time.

These cases point to the need for a system of timing manipulation that can produce the intricacies of a human performance, as well as develop interesting non-traditional timing relationships. David Jaffe's work in this area has laid the foundation for what he calls "ensemble timing", or alternatively as a process, "time warping". Jaffe defines these terms to suggest the "interaction between two or more "voices", each with its own view of time" (Jaffe 1985). A voice might be a French Horn player in a brass quintet.

However, the voice might not be a performer at all, but rather a complex series of events stored in a computer. In any case, the "give and take" inherent in each voice, and the relative proportion of each, are the fundamental elements of ensemble timing in music.

CHAPTER II

BACKGROUND

The current paper will draw on information from three diverse subjects, each of which has a well-developed jargon of its own. These subjects are: time warping, fractals, and the Musical Instruments Digital Interface (MIDI). Time warping is a concept delineated by computer music, though long observed in musical performance practice. The second of the group, fractals, were introduced as a new class of mathematical models by Benoit Mandelbrot, and have since been applied to a number of quite diverse projects. And finally, MIDI is a communications standard which insures the compatibility between various electronic musical instruments.

Since each of these topics plays an important role in the development of this paper, the following background section presents some of the fundamentals of each. The information is in no way complete, but it does detail the topics to the extent that they are used within the context of this paper.

Time Warping

The basic approach to timing adopted in the current implementation consists of the application of a tempo warping function to a section of the music under consideration. The warping function may be thought of as a distinct musical

gesture such as an accelerando or a retard, or in the case of more subtly varying functions, an expressive rubato.

The warping function consists of four primary parameters: the warp starting and ending times, and the initial and ending tempo scalers. The starting and ending times simply delineate which section of the music is to be altered by the warp. The tempo scalers determine the ratio by which time will be adjusted. A tempo scaler of one indicates that time will remain unchanged. A tempo scaler of 1/2 indicates that the instant in time will pass twice as fast as originally, and conversely a tempo scaler of two will pass twice as slowly.

The warping function's only restriction is that it be strictly positive. Sections left undefined by the warping function are implicitly valued at one, which leaves time unchanged. Discontinuities in the function are allowed, and represent instantaneous changes in tempo. The altered time of the warp function may be determined by integrating each segment of the warp function. Thus, modifications in a given time "x" may be computed through the algorithm described in the pseudo-C code of Figure 1.

As an example, consider the single-segment warping function in Figure 2. When this warping function is applied to the original integer times in Table 1, the resulting warped times are produced which reflect the acceleration in tempo that occurs between original times five through ten, and the subsequent displacement of ensuing times by 1.25 seconds.

If (x < wf->st)
/* x in pre-warping region: no change */
y = x

Else if (x < wf->end) /* x in warping region */

$$y = wf->st + \int_{wf->st}^{x} f(t)\ dt$$

Else /* x in post-warping region: displace time */

$$y = x - warplen + \int_{wf->st}^{wf->end} f(t)\ dt$$

where:
old time = x >= 0
new time = y
wf->st = warp function start time
wf->end = warp function end time
warplen = wf->end - wf->st
and f() is tempo warping function

Figure 1. Modifications on time during warping.

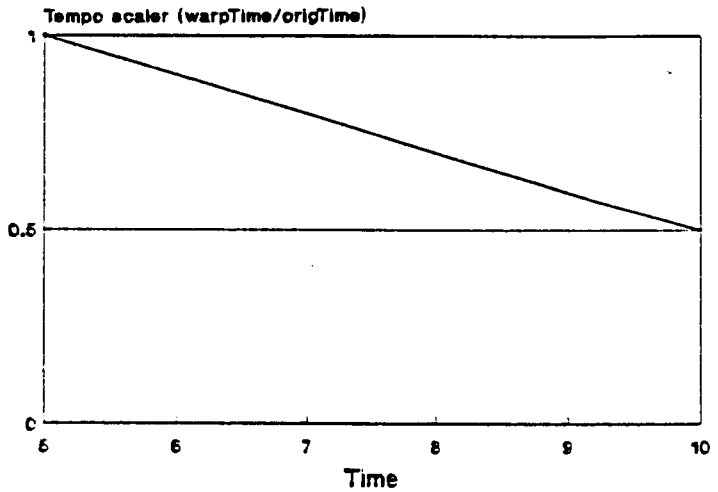


Figure 2. Example warping function.

Table 1. Results of Figure 2 warping function applied to integer times.

	Original Time	Warped Time
pre-warped region	0.00	0.00
	1.00	1.00
	2.00	2.00
	3.00	3.00
	4.00	4.00
warping region	5.00	5.00
	6.00	5.95
	7.00	6.80
	8.00	7.55
	9.00	8.20
post-warping region	10.00	8.75
	11.00	9.75
	12.00	10.75
	13.00	11.75
	14.00	12.75
	15.00	13.75

Fractals

Mandelbrot's introduction of fractals has resulted in a flurry of activity in a variety of fields. In addition to their widespread use in computer graphics, several musical applications have been published. Composers such as Charles Dodge (Dodge 1988), Charles Wuorinen (Dodge 1985), and Gary Nelson (Nelson 1988) have integrated fractals into various parts of their work. In addition, physicists Richard Voss and John Clarke (Voss and Clarke 1978) have demonstrated that a phenomenon closely related to geometric fractals is reflected in the long-term spectral distribution of various traditional Western musics.

The striking feature that fractals possess is their self-affinity. Self-affinity implies that each small portion of a fractal shape, when magnified, can be exactly reproduced on a larger portion (the horizontal and vertical scales are not necessarily equally magnified). This property can be easily implemented through the use of a recursive algorithm. As a simple example, given the initial two-segment shape in Figure 3, each segment is subsequently replaced by a scaled version of the initial two-segment shape. This process is continued through generations of ever shortening segments, resulting in a far more sophisticated shape than the original.

By substituting the fractal function produced by the recursive process for the tempo warping function,

sophisticated timing interactions can be created through the specification of comparatively simple functions. The implementation of this concept and its results are among the primary subjects of this paper.

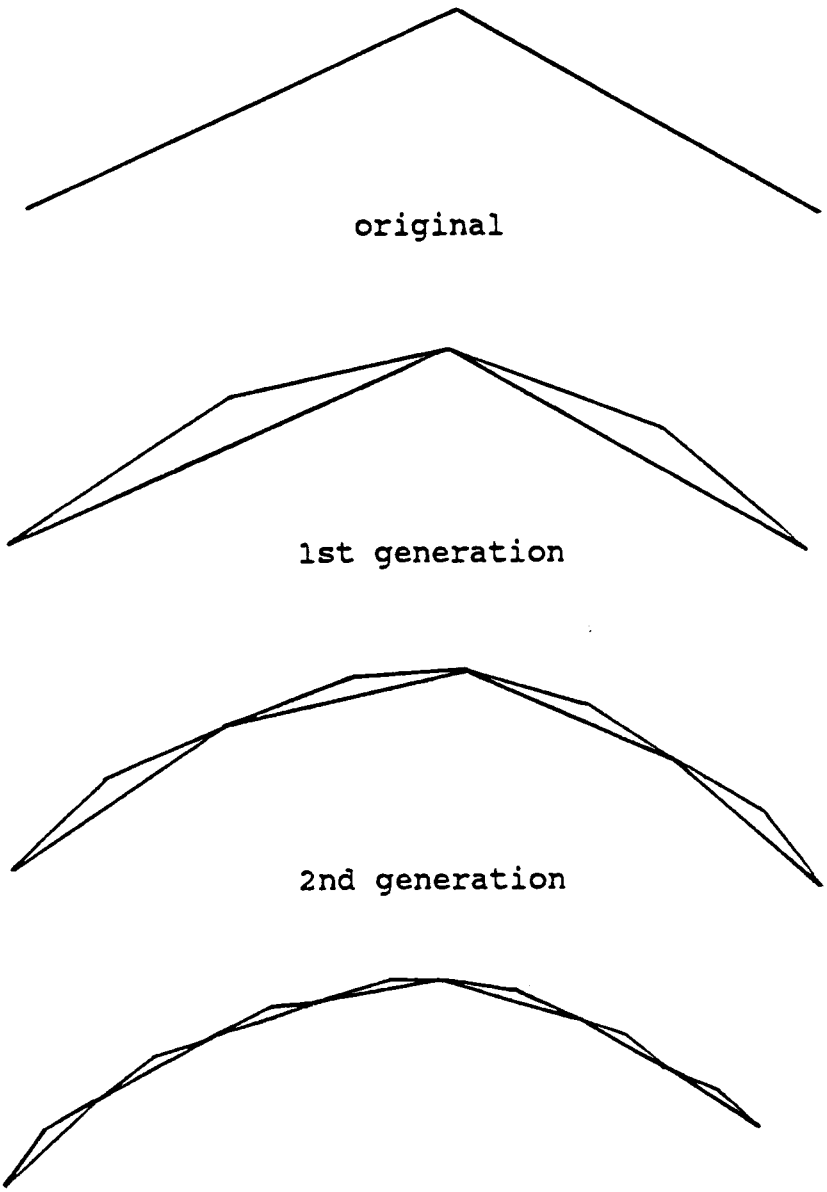


Figure 3. Simple function during early generations of fractal process.

MIDI and Standard MIDI Files

MIDI was developed as a communication standard for electronic musical instruments so that compatibility between equipment of various manufactures could be achieved. The information it carries takes the form of "events" which describe discrete components of a musical composition or performance. These components include such things as key presses and releases, control changes (this might be information sent by a modulation wheel, joystick, pedal, etc), and program changes (what type of sound the synthesizer uses; i.e. piano, brass, chain saws).

MIDI events are typically transmitted in real-time at the moment when they are to take effect. An event consists of a status byte which indicates the type of event to be executed, and in most cases, a series of data bytes which contain information specific to the event to be executed. For example, a "key on" event is described by a status byte of value 80 hexadecimal and two data bytes which describe which of the keyboard's notes should be activated, and with what velocity.

All MIDI-compatible musical instruments have built-in ports which can be connected directly via MIDI cables. However, connecting a Macintosh (the computer used in this paper's implementation) to a MIDI instrument requires a hardware Mac/MIDI interface unit. The interface connects to one of the Macintosh's two serial ports, and has MIDI ports

which may be cabled to various MIDI instruments. The interface for the Macintosh may run at one of three clock rates: .5 MHz, 1 MHz, and 2 MHz. The 1 MHz clock rate is the most common. The interface provides a buffer to help facilitate communication between the Mac and the MIDI instrument.

MIDI transmission itself is standardized at a rate of 31,250 bits per second. With eight data bits and two stop bits per byte, the transfer rate is roughly three bytes per millisecond. Thus, a three-byte "key on" event will take about a millisecond to be transmitted. However, not all events are three bytes in length, in fact some, such as "system exclusive" events which contain manufacturer-specific synthesizer information, are theoretically unlimited in their length.

This points out the fact that a data-intensive transmission could overwhelm the MIDI data rate and even overflow the interface buffer. This is a potential problem for MIDI, and must be kept in mind, especially in instances where timing is crucial, such as time warping.

While MIDI defines a standard for communicating data in real-time, standard MIDI files provide a means of storing MIDI events in a standardized format in data files on disk. This allows the exchange of information between various software packages and hardware. For instance, a musical piece may be recorded by one software package (usually called a "sequencer"), saved to a standard MIDI file, and then read in

and modified by an entirely different application. The piece may then be again saved to a standard MIDI file for future processing or use.

This procedure is in fact, the one adopted by the current time warping implementation. This convenient and flexible method of information exchange allows the user to make use of the sophisticated features of a commercially available sequencer, and adopt time warping as a special purpose technique in the musical process.

CHAPTER III

IMPLEMENTATION

The Macintosh computer was chosen as the environment most suitable for the exploration of time warping. The computer is reasonably fast, and currently represents the most familiar approach to computer music for most musicians. Since experimentation on an interactive level was determined to be the most suitable approach to time warping, the inherently non-real time method of direct synthesis was ruled out for this project. Additionally, the increasing compatibility among computers using MIDI has become such that a non-MIDI implementation would have virtually no chance of future use by other composers and musicians.

User Interface

The time warping user interface was designed to encourage experimentation. Numerous features were included to facilitate creation of the warping function, and to allow for immediate feedback of results

The main screen for time warping (see Figure 4) displays three lists: the "note list" which contains the active MIDI event list, the "warping function" in which the user may specify the basic time warping function, and the "region to be warped" which defines where the basic function will be applied in time. Thus, these last two lists work together to

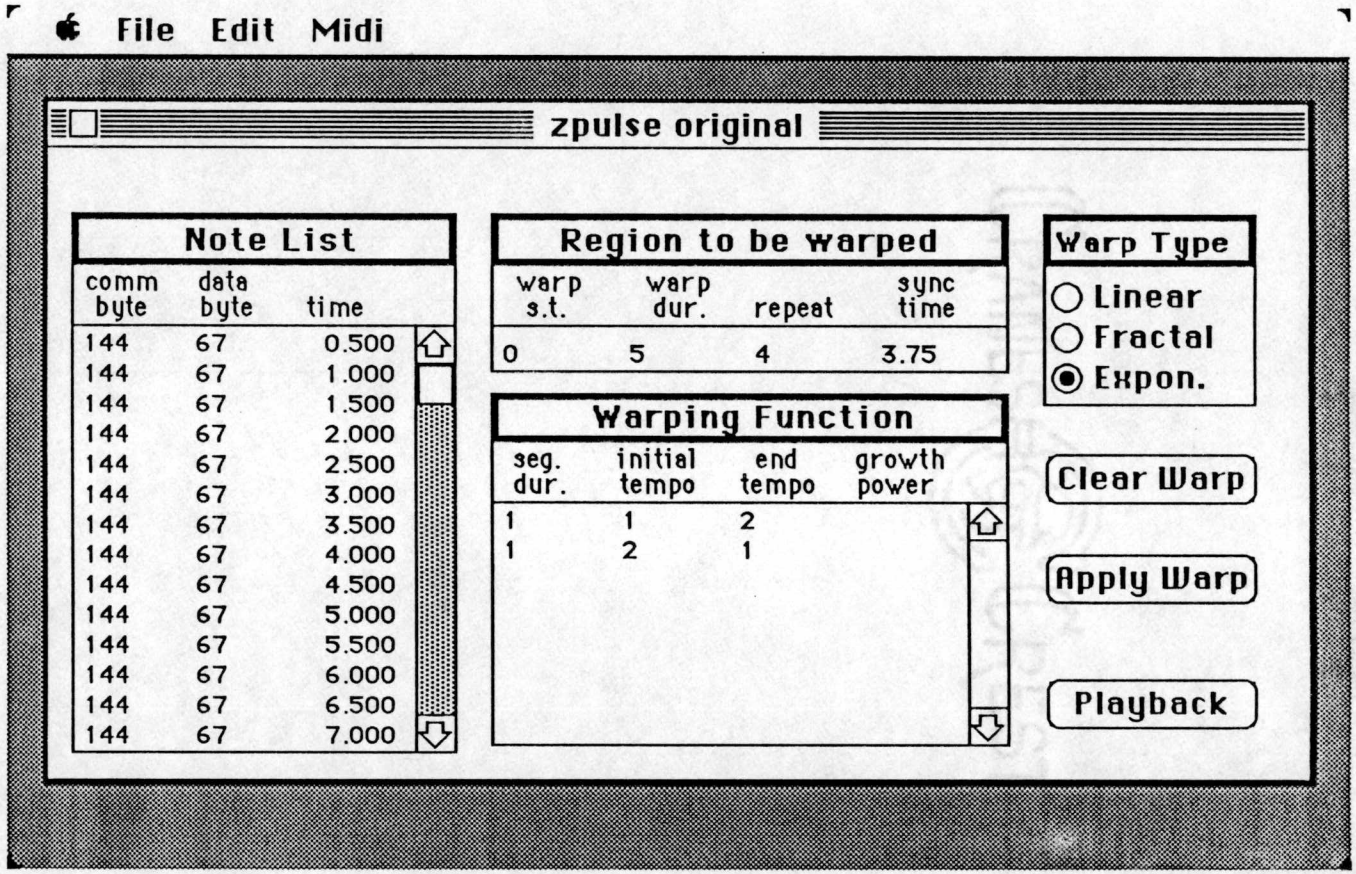


Figure 4. Time warping main screen.

define the complete warping function.

Additionally, the "warp type" choices specify whether the function will be comprised of line segments (linear), curves (exponential), or a fractal shape generated from the basic function. If an exponential warp type is selected, an extra parameter, termed the "growth power", is needed to determine the direction and the depth of the curves. This parameter is located in the last column of the "region" list when exponential warping is selected. Additionally, an optional parameter for exponential warping called the "sync time" (located in the last column of the warping function list) allows the user to fit the warped region to a specified length by automatically adjusting the depth of the curves. With a fractal warp type, an extra parameter called the "fractal ratio" (located in the same position as the growth power) is needed to dictate the scaling factor used in producing each subsequent generation in the fractal process.

The "repeat" parameter in the warping region is a time-saving feature which allows the warp to be applied to the regions immediately following the one specified. This is an especially important feature in measured musics where the warping repeatedly cycles through a collection of measures. In fractal and exponential warping, this reduces computation considerably.

The main screen also contains a collection of buttons which help the warping process along. The "apply warp" button

serves as the signal to implement the specified warping function with respect to the note list. As soon as warping is completed, the "time" parameter in the note list is updated for inspection. The warping function specified by the user prior to the application of the warp is left intact for possible revision. If the user is dissatisfied with the results, the original note list can be quickly restored by a "revert to saved" command located in the "File" menu, and revision can be made to the warping function. The warp function may be completely erased by the selection of the "clear warp" button.

At any point in the process, the user may audition the current note list by hitting the "playback" button. Playback may be aborted, which is often desired in the cases of long note lists, by hitting any key on the Macintosh keyboard. It is important that the correct clock rate and serial port be selected in the "Midi" menu before playback is attempted. Playback provides the single most important method of evaluating the results of the warping process.

Data Structures

Time warping is a very data-intensive procedure by its very nature. Likewise, music often consists of tens of thousands of notes and nuances. Thus, the representation of this information must be concise and efficient. The current implementation uses two primary, and several auxiliary data

structures for internal storage and processing.

The first structure contains the sequence of MIDI events and their timing information. This sequence is coded as a linked list of 100-event blocks. Each event includes a starting time for that event (in milliseconds), its status byte, and up to either four data bytes or a pointer to a linked list of data bytes if more than four. Events are always ordered in increasing fashion by their starting times. In addition, each event also maintains the last stored starting time for quick processing when the user chooses "revert to saved" from the "File" menu. Each block also maintains the number of events currently in that block. The total block size runs about 1300 bytes in length. This block-configuration was selected for its efficiency in accessing events sequentially, and for its ability to store long musical sequences which might exceed 50,000 events in length.

The other primary data structure and auxiliary structures are used to store the warping function. The information contained in the primary structure includes the starting and ending times of the region to be warped, the initial and ending tempo scalers used to effect the warp, and a pointer so that the function may be represented as a linked list where a multiple-segment warp is indicated. The auxiliary structures are also linked lists which correspond to the primary structure. They are used to store the growth powers and sync times which might be needed to supplement the information

contained in the primary structure.

Playback

Since time warping often involves the subtle shifting of time on an order as small as a millisecond, a similar resolution is essential in the playback mechanism. The Macintosh contains a standard Rockwell-compatible 6522 Versatile Interface Adapter (VIA) chip, which has two timers, one of which may be programmed to interrupt periodically. In the current implementation, this timer is set up to interrupt at one millisecond intervals. A handler routine increments the number of milliseconds passed, which is maintained in a global variable, since the user hit the playback button.

While the timer runs on the interrupt level, a playback loop is entered which is responsible for sending MIDI events to the interface. Each time through the loop, the time of the next pending event in the MIDI event list (in milliseconds) is compared with the number of milliseconds passed since the start of playback. If the event's time is less than or equal to the current time, the event is transmitted via the selected serial port to the MIDI interface. The user may terminate the playback sequence and reset the timer at any point by hitting any key on the Macintosh keyboard. An "all notes off" control change is consequently sent out to insure that notes left ringing indefinitely due to the cancelation of their "note off" events are silenced.

This routine allows for precision timing which is maintained internally even when the MIDI interface unit gets bogged down in a data-intensive passage. Since the interface typically buffers its input, the playback loop is able to process events faster than the millisecond interrupt time in most cases. This is aided by the use of "running status", a MIDI shorthand, which reduces the number of bytes needed in transmission. What this all implies is that events will be sent out from the Macintosh on time, and hopefully will make their way from the MIDI interface with a minimal delay. As pointed out earlier, the interface buffer can be filled in extreme cases. When this happens, it is the interface's responsibility to handle the situation. Thus, the limitations of MIDI itself far exceed those of the playback mechanism, so an understanding of MIDI's limitations will allow the user to know what to expect in the way of output.

Linear and Exponential Warping

Linear and exponential warping provide an effective means of representing a wide array of functions. In fact, with sufficient segments, any function can be closely approximated using either of these warp types (see Figure 5). Thus, linear and exponential warping represent a conventional building-block approach to time warping. The following paragraphs detail this approach.

In the implementation presented here, linear warping is

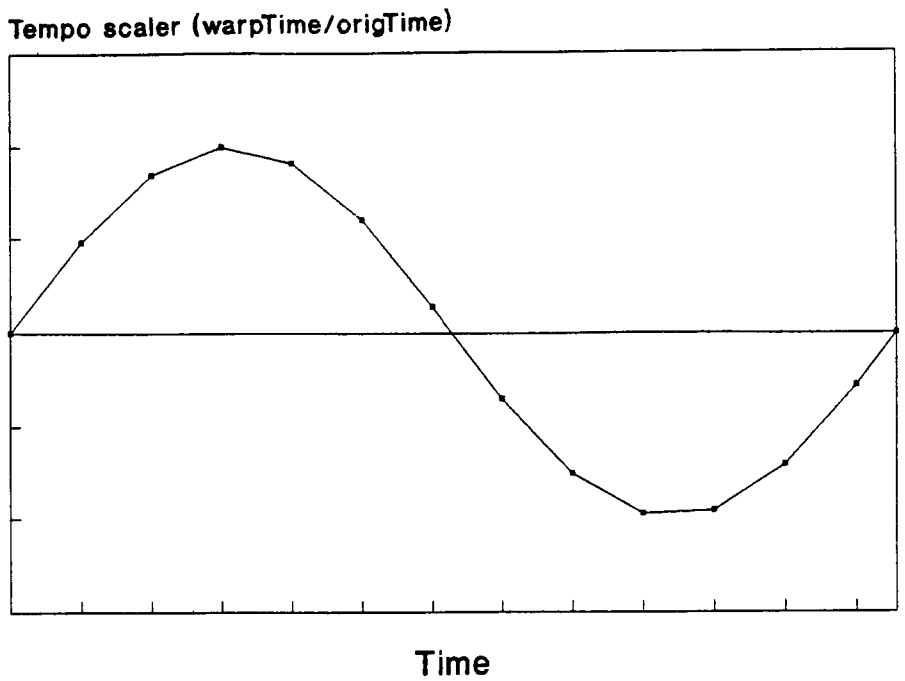


Figure 5. Discretized sine wave produced with a linear warping function.

simply a special case of exponential warping. The equation in Figure 6 is the tempo function which utilizes the warping parameters that the user specifies. Given a time "y" between zero and the warp length, this equation returns the tempo at that moment in time. This tempo function is integrated in the time warping process to calculate warped times. Note that for linear warping, the growth power is simply set to one, which simplifies the equation. For an exponential warp, the growth power may be any value greater than zero. Thus, a linear warp is simply a special case of exponential warping.

$$\text{Current Tempo}(y) = wf\text{->}et - tempodiff * ((warplen-y)/warplen)^{(1/wf\text{->}gp)}$$

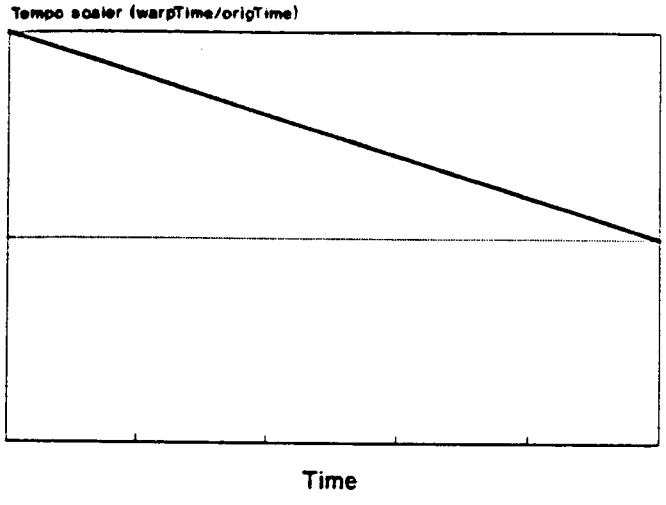
where:

- wf->st = warp function start time
- wf->end = warp function end time
- wf->it = warp function initial tempo
- wf->et = warp function end tempo
- wf->gp = warp function growth power
- warplen = wf->end - wf->st
- tempodiff = wf->et - wf->it
- time y is some number such that 0 <= y <= warplen

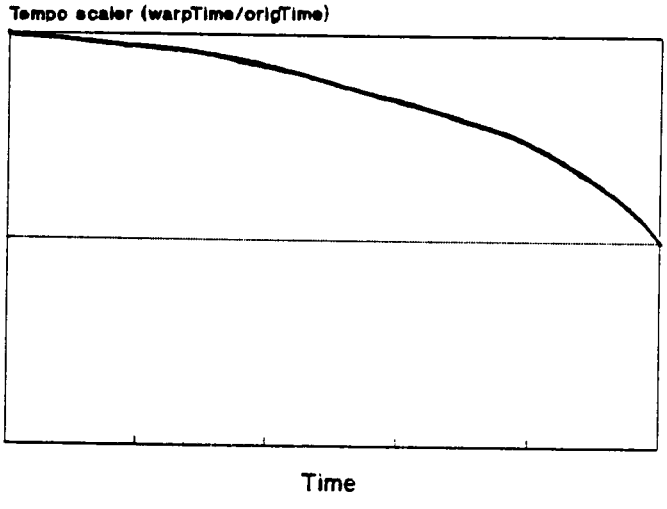
Figure 6. Tempo function.

As noted earlier, the growth power is used to effect the curved transitions between the initial and ending tempo scalers. While a growth power of one yields a linear warp, values greater than one will result in a slower initial growth toward the end tempo, and conversely a growth power less than one will cause a faster initial growth toward the end tempo (see Figure 7). The growth power must, however, always be greater than zero.

gp = 1.0



gp > 1.0



gp < 1.0

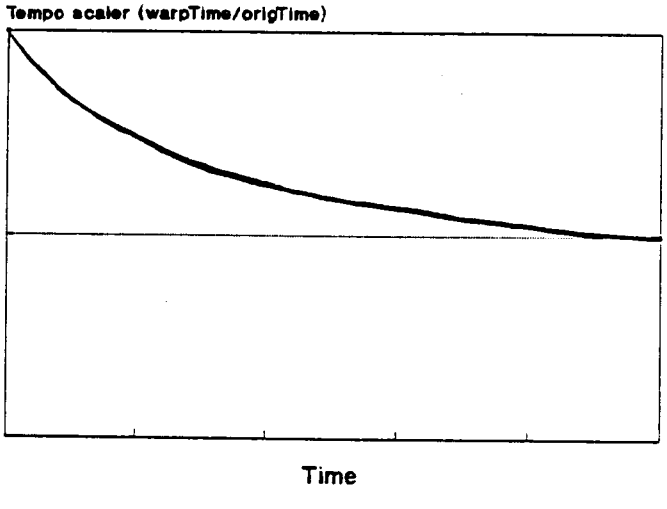


Figure 7. Various growth power (gp) tempo transitions.

Syncing may be specified as an option for exponential warps. Syncing fits the warped region to a specified length by automatically adjusting the growth power parameter.

Though the result of exponential synchronization may happen to be a linear warp, syncing cannot be applied to linear warping. This is due to the fact that synchronization occurs through the adjustment of the growth power parameter (which by definition is always one for a linear warp) to make the length of the warped region match the sync time.

As an example, if we are given the warp function in Figure 8 and a sync time of 1.667, then we are looking for the growth power which makes the equation of Figure 9 true. By moving the constant sync time to the left side of the equation and using a numerical root-finding technique, such as the secant method (modified so that a negative growth power is never considered in evaluation), we can determine that a growth power of 0.5 satisfies this particular example. This is the process, expanded to work with multiple-segment warp functions, which the current implementation follows.

In the current implementation, the integral part of the equation of figure 9 is computed through symbolic integration of the tempo function. The equation that results from symbolic integration is as follows:

```
new time  =  warplen * (wf->et + wf->gp*wf->it)/(1 + wf->gp)
```

where the constants are the same as in Figure 6.

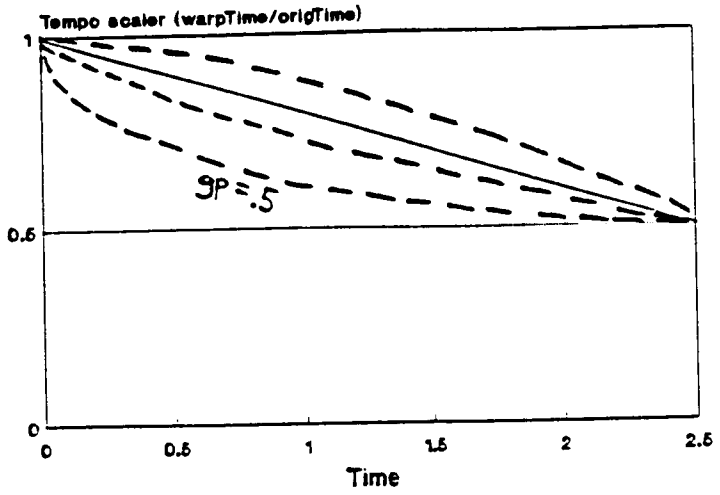


Figure 8. Example warping function.

$$\int_0^{2.5} \mathcal{F}(t)\, dt = 1.667$$

where:
$\mathcal{F}(t)$ is the tempo warping Function

Figure 9. Syncing equation.

There are further restrictions which must be placed on the sync time. Given the two-segment function of Figure 10, note that the depth of the curves are restricted by the bounds of the endpoints. In this example the sync times must be restricted by the range of 2.5 to 5.0. The user can avoid this restriction by specifying additional segments with endpoints that expand the bounds.

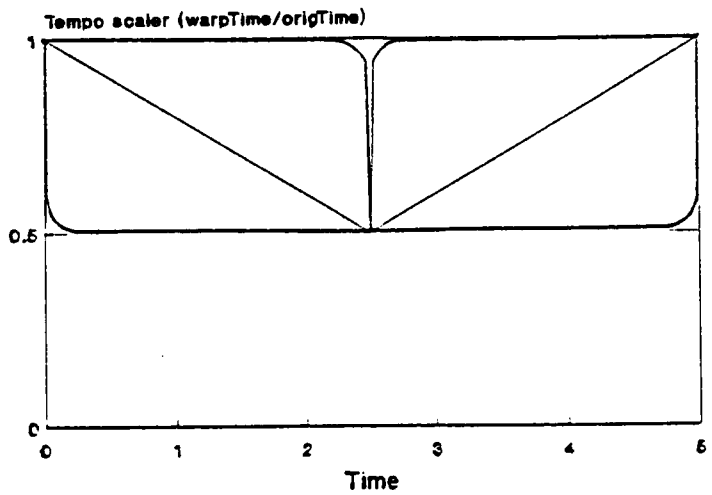


Figure 10. Two-segment warp demonstrating sync range.

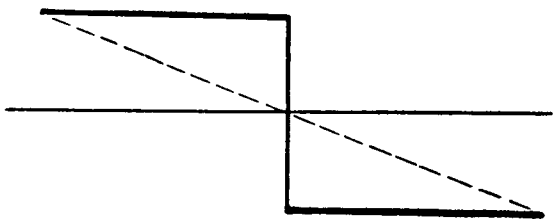
Fractal Warping

Fractal warping is a clear departure from the approach of conventional linear and exponential warping. In fractal warping the user simply supplies a basic function, which is then reproduced in miniature over each segment in each subsequent generation of the function. This process results in a myriad of nuances and intricate timing interactions when applied in time warping. The implementation of this recursive process, however is in many ways related to that of linear and exponential warping.

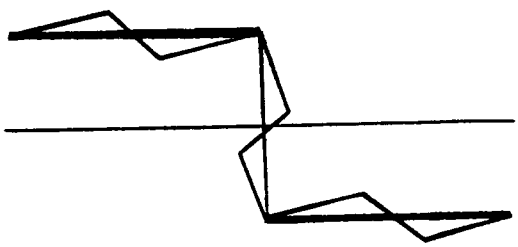
The fractal function is comprised of numerous short linear segments, and in that sense, can be classed as a linear warp with the property of self-affinity. The reproduction of the function on each segment of the previous generation is managed through careful proportioning of the original function. This reproduction continues on each segment until all subsegments have time durations of less than 0.1 seconds.

Several factors must be accounted for in the implementation of the fractal process. First of all, a rotated copy of the warping function must be made in which the endpoints lie on the horizontal axis. This rotation allows the function to be reproduced in a scaled form over each of the subsegments of the warping function. Furthermore, while the horizontal dimensions of the reproduction can be simply rescaled from the original, vertical fractal scaling must be

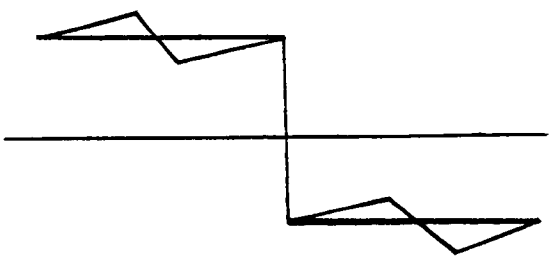
modified so that invalid functions are not generated. By scaling the vertical dimension of the reproduction to be proportional to the time the segment under consideration spans, rather than simply rescaling the original function, only valid functions are produced (see Figure 11).



original warping function
(dotted lined represents horizontal
axis when function is rotated)



simple fractal scaling (produces invalid function)



acceleration-sensitive fractal scaling

Figure 11. Basic fractal scaling.

The effect of this scaling process is to minimize fractal changes over sections where the tempo is already rapidly varying, while maximizing deviations in static or nearly constant tempos. This is an important quality which reflects the inherent relationship of nuance to large-scale gesture in music. When a performer makes a transition from a dramatic to a more expressive passage, the forthright emotion and energy of the former are transformed into a more personal form characterized by subtle intimacies in the phrasing.

The vertical scaling process can be further aided by the introduction of an additional scaling factor as a variable. This variable, when restricted to the range negative one to positive one, provides a means of fine tuning the fractal function in a manner akin to exponential warping's growth power. This variable is termed the fractal ratio (see Figure 12), and is specified by the user as one of the "warping region" parameters.

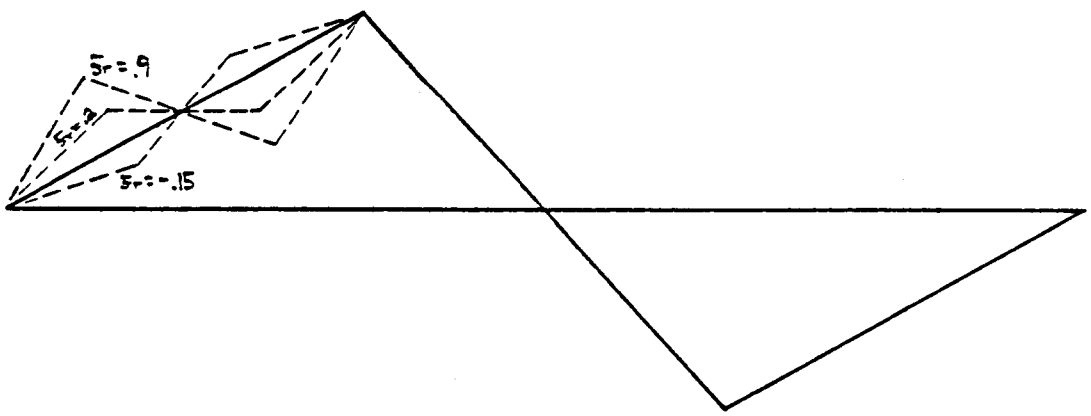


Figure 12. Various fractal ratios.

The algorithm for the fractalizing process is given in Figure 13. It uses the warp function as an input, and modifies that function so that it becomes a fractal warping function.

The current implementation does not include syncing as an option for fractal warping. Some warping functions can only be synchronized over a small range or even a single point, regardless of the fractal ratio. For instance, when Figure 12's endpoint tempo ratios are one, and the maximum and minimum are inverses of one another, the function will be synchronizable only over a single point. In order to keep the implementation as general-purpose as possible, the user is restricted to specifying the fractal ratio. Fine tuning of this parameter through trial and error allows the user to manually sync the warp where possible, and modify the warping function where special cases are encountered.

```
Fractalizing process

-set the "current segment" to the first segment of the
 warp function

-while (the "current segment is not NULL)

     -set the "current subsegment" to the first segment
      of the rotated warping function

     -while (the "current subsegment" is not NULL)

          -replace an appropriate portion of the "current
           segment" with a scaled version of the "current
           subsegment"

          -if the "current subsegment" spans a time
           duration of 0.1 or more, call the fractalizing
           process recursively to further fractalize the
           "current subsegment"

          -set the "current subsegment" to the next
           segment of the rotated warping function, or
           NULL if no more segments


     -set the "current segment" to the next segment of
      the warp function, or NULL if no more segments

-end
```

Figure 13. Fractalizing algorithm.

CHAPTER IV

EXAMPLES AND RESULTS

Analysis of a Simple Example

Returning to the simple example warping function of Figure 10 on page 25, we investigate the effect of a range of fractal ratios and sync times using fractal and exponential warping. This was done through the warping of a constant pulse whose rate was two beats per second (corresponding to the "original time" in Table 2). The warp was repeated five times so that its cyclic "feel" could be discerned. Various fractal ratios through the range negative one to positive one were tried, and then the corresponding sync time was applied to exponential warping so that their warp times could be compared. These results are contained in Table 2. The process allows for up to a two millisecond error, which accounts for the slight deviations in the specified exponential sync time verses the resultant warped time.

By comparing the resultant warped times of Table 2, we can determine some of the general characteristics which differentiate fractal from exponential warping. The sync times which most closely match that of a linear warp (an almost flat curve in exponential warping and a small value for the fractal ratio in fractal warping) also demonstrate the smallest differences between the respective exponential and fractal warped times. However, sync times which tend toward the

Table 2. Simple example using various corresponding fractal ratios (fr) and sync times (st).

	Original Time										
	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
fr=-1.0	0.000	0.557	1.132	1.661	2.140	2.502	2.864	3.343	3.872	4.448	5.004
st=5.000	0.000	0.500	1.000	1.500	2.000	2.500	3.000	3.500	4.000	4.500	5.000
fr=-0.75	0.000	0.536	1.073	1.564	2.003	2.342	2.681	3.121	3.611	4.149	4.684
st=4.684	0.000	0.496	0.983	1.459	1.918	2.341	2.711	3.186	3.683	4.184	4.685
fr=-0.5	0.000	0.515	1.015	1.467	1.868	2.185	2.502	2.903	3.354	3.855	4.370
st=4.370	0.000	0.491	0.962	1.410	1.826	2.184	2.499	2.911	3.378	3.870	4.370
fr=-0.325	0.000	0.501	0.975	1.400	1.774	2.076	2.378	2.752	3.177	3.651	4.152
st=4.152	0.000	0.486	0.944	1.369	1.753	2.075	2.371	2.746	3.180	3.655	4.152
fr=-0.1	0.000	0.483	0.923	1.314	1.654	1.937	2.220	2.560	2.950	2.390	3.873
st=3.873	0.000	0.479	0.916	1.307	1.651	1.936	2.216	2.554	2.947	2.389	3.874
fr=0.0/ st=3.75	0.000	0.475	0.900	1.275	1.600	1.875	2.150	2.475	2.850	3.275	3.750

Table 2. (Continued)

	Original Time										
	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
fr=0.0/ st=3.75	0.000	0.475	0.900	1.275	1.600	1.875	2.150	2.475	2.850	3.275	3.750
fr=0.1	0.000	0.467	0.877	1.236	1.546	1.813	2.080	2.390	2.750	3.160	3.626
st=3.626	0.000	0.469	0.881	1.238	1.546	1.812	2.083	2.396	2.754	3.161	3.627
fr=0.325	0.000	0.448	0.824	1.149	1.425	1.674	1.923	2.199	2.524	2.900	3.348
st=3.348	0.000	0.454	0.829	1.145	1.420	1.674	1.927	2.229	2.554	2.920	3.350
fr=0.5	0.000	0.433	0.782	1.080	1.330	1.565	1.800	2.050	2.348	2.679	3.130
st=3.130	0.000	0.435	0.773	1.056	1.314	1.565	1.824	2.102	2.405	2.739	3.132
fr=0.75	0.000	0.411	0.722	0.981	1.194	1.408	1.622	1.834	2.094	2.405	2.815
st=2.815	0.000	0.380	0.654	0.907	1.157	1.407	1.661	1.924	2.198	2.488	2.816
fr=1.0	0.000	0.388	0.659	0.880	1.054	1.248	1.441	1.615	1.836	2.108	2.495
st=2.5	0.000	0.250	0.500	0.750	1.000	1.250	1.500	1.750	2.000	2.250	2.500

extremes of the syncing range show ever-widening differences in the two warp types (considerably more than a tenth of a second). These differences can be summarized as follows: as the sync time tends toward the endpoints of the syncing range, fractal warping's initial acceleration is slower than that of exponential warping, but this slow start is made up for with a recovery which allows the fractal pulse to catch and pass the other. In fact, the fractal pulse moves ahead by so much that it has to rapidly retard near the end to come out even. The non-extreme sync times also demonstrate this seemingly quirky series of accelerations and retardations, though not nearly so obviously.

Extensive audio comparison of the various experiments was conducted to determine the perceptible characteristics of both warp types. Exponential warps were most successful in areas closest to that of a linear warp. Specifically, the sync times of 3.13 and 4.37 seemed to be the outer bounds for the experiments in which the tempo transition was perceptible. In sync times which were smaller than 3.13, the tendency was for the result to be heard at a constant pulse of four beats per second with a pause at the beginning of each repeat. Correspondingly, sync times greater than about 4.37 resulted in a constant pulse at two beats per second with a sudden surge in the middle of each cycle. However, the linear experiment (produced with a sync time of 3.75) was perhaps surprisingly not the most "musical" of the various warps. The

linear transition was somewhat predictable in nature, and had the quality of an inexperienced performer. Rather, the sync times between the linear warp and the respective bounds 3.13 and 4.37 demonstrated the most musically useful transitions, though all were lacking in exceptional musicality.

The corresponding fractal warps, however, consistently demonstrated an inherent animation, and something like musical drive. Unlike the exponential type, the entire range of fractal syncing clearly accentuated the tempo transition. The times closest to a linear warp had a satisfying sense of rhythmic growth and resolution in their nuances, while those at the extremes had a Mahleresque lilting or swelling quality in their feel. Each individual experiment also had a peculiar quality of its own, which could allow the user to pick from a field of useful possibilities the one most suited to the musical context of a piece. Some were almost waltz-like, while others had a determined drive in their feel. The common characteristic among them all was an aggressiveness, very much like that brought out by a performer as rhythmic intensity.

A Musical Example

The author's composition, Clockwork, is an example of how fractal time warping can be incorporated into the musical process. Clockwork utilizes a single basic warping function (see Figure 14) stretched over the length of the entire piece. By setting the fractal ratio to a relatively high value of 0.618, deviations in the original function are created which lend the music a dynamic sense of movement. The overall effect is a pulse which is musically convincing on both macro and micro levels.

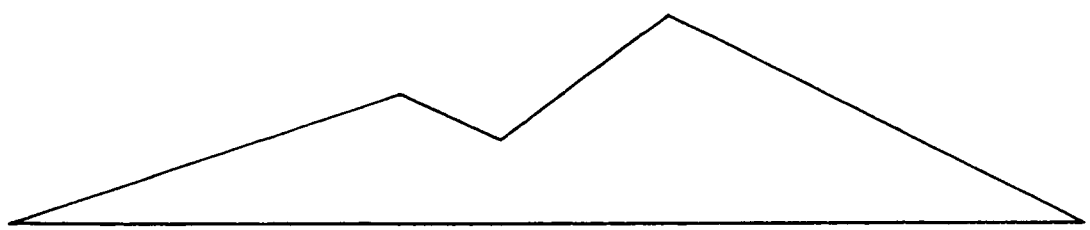


Figure 14. Clockwork warping function.

CHAPTER V

EXTENSION OF THE MODEL

An extension to the basic fractal model is the use of fractals which are not exactly self-affine, but rather statistically self-affine. Statistical self-affinity occurs when each small section looks like, though not exactly like, a larger portion of the function.

A preliminary implementation of this feature was tested which used a normal distribution where the mean was set to the user-specified fractal ratio. The standard deviation was found to be optimal at a value of 0.25. This deviation allowed a maximal variation within the gestural character of the specified fractal ratio.

This extension of the basic model provided a range of different usable transitions to choose from, while maintaining a common character. Thus, time warping was brought a step closer to the human performance model, where a musical passage is played the same and yet differently in each individual performance.

CHAPTER VI

APPLICATIONS

The application of fractal warping offers many exciting possibilities. These applications extend beyond the simple synthesis of a musical pulse presented in earlier in this paper. A typical use of fractal warping would be the modification of a recorded performance to give it a new and different phrasing. The ability to dynamically alter musical phrasings to taste clears the way for creative adaptations of compositions to dance and film, as well as other multi-media productions.

Since fractal warping is a general model of time modification, anything that is described as a series of discrete events may be subject to the process. Prime candidates that come to mind include choreography simulation, computer animation, and holography.

CHAPTER VII

FUTURE WORK

As with any interesting project, there are always improvements and new avenues to explore. The are a number of features which might be added to improve the user interface and make time warping appear more accessible to the musician. Additionally, other equally valid and diverse approaches to the warping function are possible.

A useful feature for future addition is the independence of musical tracks. Under the current implementation when the MIDI file is read in, all events are collected in a sequential string, and warping is applied to this as a whole. Some standard MIDI files separate tracks of events, so that the event list of each part is maintained apart from the others (i.e. all the events for the violin are followed by all the events for the viola, etc.). By maintaining this independence, warping could be directed at any one or group of these parts, and leave the others unaffected. Under the current implementation, this can be done through some extra work on the part of the user. By saving each part as a separate standard MIDI file and warping them as needed, the MIDI files can be easily recombined in most commercial sequencers. However, more immediate feedback could be gained by having this process built in to the time warping implementation.

Yet another useful addition would be the introduction of

graphic entry for the warping function. This would help musicians in visualizing the function and its effect. The reduction of numerical parameters is always desirable in the implementation of arts-oriented programs. This was in fact included in Jaffe's Lisp implementation of time warping. The process could be further developed by the inclusion of graphically-represented "preset" functions which might be chosen rather than specifying one from scratch. More research is needed to determine just which functions are versatile enough to serve as presets though.

CHAPTER VIII

CONCLUSIONS

Fractal time warping offers a new a possibly more intuitive process for ensemble timing in computer music. While allowing the user to specify a basic gesture, the fractal process works to fill out the intricate details of nuances and phrasings in a meaningful way. Additionally, fractal warping appears to provide a range of such nuances and phrasings, thus allowing the user to choose an expressiveness appropriate to the musical context under consideration. Whether or not this range fulfills every need is still open to question, however.

Nonetheless, the technique has provided encouraging results which warrant further investigation. The qualitative differences between the fractal results and those from exponential warping were striking, and highlighted the comparative limitations of the later. Thus, the application of fractal warping to computer music provides an additional tool for the simulation of human performance interactions, and encompasses a group of new possibilities for exploring innovative conceptions of musical time.

Bibliography

Bibliography

Apple Computer. 1986. Inside Macintosh Vol I-V. New York: Addison-Wesley Publishing.

Apple Computer. 1988. Macintosh Family Hardware Reference. New York: Addison-Wesley Publishing.

Burden, R., and J. Faires. 1985. Numerical Analysis. Boston: Prindle, Weber & Schmidt.

Chernicoff, S. 1987. Macintosh Revealed Vol. I and II. Indianapolis, Indiana: Hayden Books.

Dannenberg, R. 1989. "The Canon Score Language." Computer Music Journal 13(1): 47-56.

Dannenberg, R. 1986. The CMU MIDI Toolkit. Pittsburgh, Pennsylvania: Carnegie Mellon University.

Dannenberg, R., P. McAvinney, and D. Rubine. 1986. "Arctic: A Functional Language for Real-Time Systems." Computer Music Journal 10(4): 67-78.

DeFuria, S., and J. Seacciaferro. 1988. MIDI Programming for the Macintosh. Redwood City, California: M&T Publishing.

Dodge, C. 1988. "Profile: A Musical Fractal." Computer Music Journal 12(3): 10-14.

Dodge, C., and T. Jerse. 1985. Computer Music. New York: Schirmer Books.

Gardner, M. 1978. "White and Brown Music, Fractal Curves and 1/f Fluctuations." Scientific American 238(4): 16-31.

International MIDI Association. 1988. Standard MIDI Files 1.0. Los Angeles: International MIDI Association.

Jaffe, D. 1985. "Ensemble Timing in Computer Music." Computer Music Journal 9(4): 38-48.

Loy, G. 1985. "Musicians Make a Standard: The MIDI Phenomenon." Computer Music Journal 9(4): 8-26.

Moore, F. 1988. "The Dysfunctions of MIDI." Computer Music Journal 12(1): 19-28.

Nelson, G. 1988. Fractal Mountains. Oberlin, Ohio: Oberlin Conservatory of Music.

Nelson, G. 1988. Real Time Interactive Composition. Oberlin, Ohio: Oberlin Conservatory of Music.

Peitgen, H.-O., and P. Richter. 1986. The Beauty of Fractals. Berlin: Springer-Verlag.

Peitgen, H.-O., et al. 1988. The Science of Fractal Images. Berlin: Springer-Verlag.

Rockwell International. 1987. Controller Products Data Book. Newport Beach, California: Rockwell Semiconductor Products Division.

THINK Technologies. 1986. LightspeedC User's Manual. Bedford, Massachusetts: THINK Technologies.

Vercoe, B. 1986. Csound Manual. Cambridge, Massachusetts: Media Lab, M.I.T.

Vercoe, B., and M. Puckette. 1985. "The Synthetic Rehearsal: Training the Synthetic Performer." Proceedings of the 1985 International Computer Music Conference. Paris: IRCAM.

Voss, R., and J. Clarke. 1978. "1/f Noise in Music: Music from 1/f Noise." Journal of the Acoustical Society of America 63(1): 285-263.

Zicarelli, D. 1987. "M and Jam Factory." Computer Music Journal 11(4): 13-29.

Appendix

The cassette tape enclosed with this paper contains six representative sound examples of simple time warping, and a recording of the author's composition Clockwork. The sound examples were produced using warp functions corresponding to those depicted in Figure 15. Each warp is applied to five seconds worth of pulses. In examples two through six the warp is repeated for perceptual clarity. The sound examples demonstrate the following conclusions pointed out in the paper:

- the mechanical and unmusical effect of linear warping.
- mild exponential and fractal warps are nearly indistinguishable, and musically acceptable.
- extreme exponential warps fail miserably at even the most basic level.
- extreme fractal warps are still musically useful in appropriate contexts.

Clockwork is a relatively straightforward piece which utilizes fractal warping to help stretch a minimal number of musical elements to their limit. Other parameters of the composition (i.e. dynamics, timbre, etc.) make use of a variation of the same function used in the time warping process.

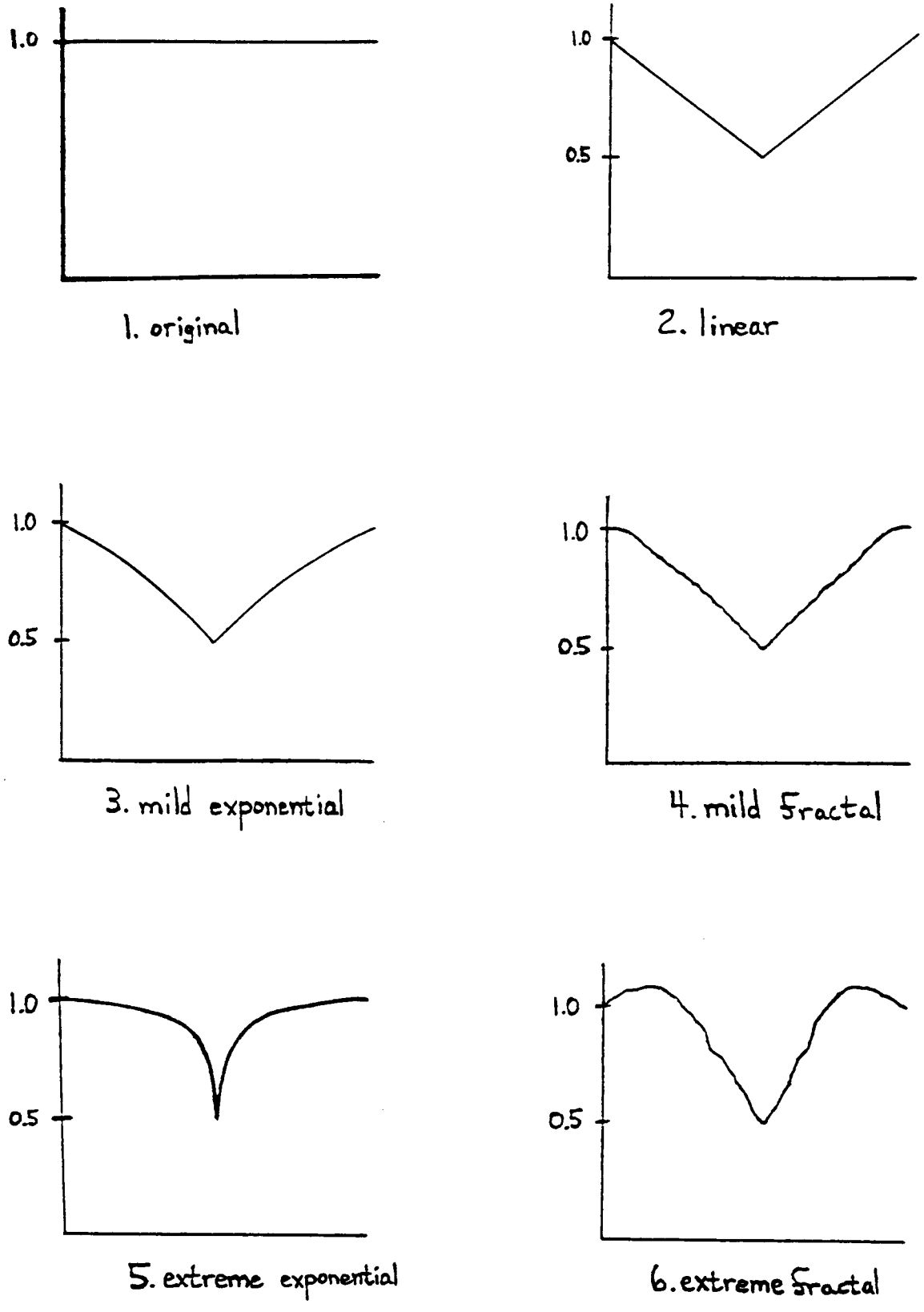


Figure 15. Sound examples 1-6.

VITA

Andrew Horner was born in San Rafael, California on November 18, 1964. He attended elementary schools in Big Rapids, Michigan, and graduated from North Allegheny High School in Wexford, Pennsylvania in June 1983. During high school he spent two years studying at the National Music Camp in Interlochen, Michigan.

In the fall 1983 he entered the Boston University School of Music as a Trustee Scholar. During the course of his undergraduate work, he spent two years studying digital synthesis in the M.I.T. Experimental Music Studio. He received a Bachelor of Music degree in May 1987.

The following September he entered the Computer Science Department at the University of Tennessee, Knoxville. During the course of his work at UTK, the author was employed by the Graduate School of Planning and the Department of Music.