

Ensemble Timing in Computer Music by David Jaffe

Presented by
Brian Highfill & Balamurali Ramasamy Govindaraju

USC ISE 575 / EE 675

April 27, 2010

Introduction

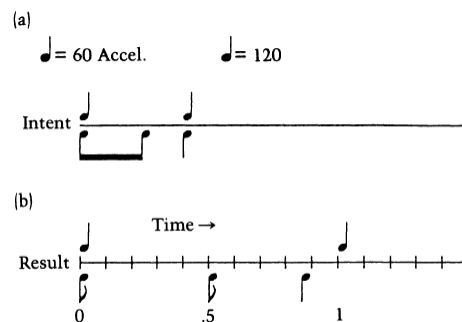
- Simulation of expressive playing
 - Requires deviation in time trajectories of voices
 - Makes possible compositions that are too difficult or impractical for humans to perform
- <http://www.youtube.com/watch?v=p4DGE5t3x0>
- Previous works before this study
 - Lacked clarity in timing issues
 - Had trouble with inter and intra- voice timing and phrasing even in the case of traditional music performances.

Ensemble timing in live and computer music

- In live performances, each performer occasionally deviates from the 'basic time'
- The performers synchronize with each other at certain events using feedback from listening
- These events anchor a pulse pattern called 'clock time'
- In general events can range from individual beats to complex pattern
- A voice is a process that produces sequence of events
- Ensemble timing consists of
 - Coordination at anchor points
 - Freedom to deviate from the basic time

Simple approaches to Timing

- Clock time : All voices are played at their own original basic time
 - Always results in rigid and mechanistic rendering of the performance
- Tempo perturbation: Adding small random values to the event durations
 - Causes accumulated vertical misalignment between different voices



Running Onset Tally

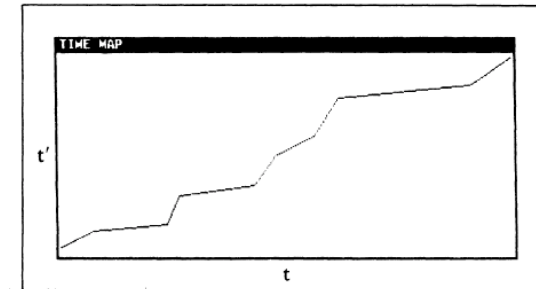
- To each of the original pre perturbed onsets we add small random values to obtain the current onset times
- The duration of the perturbations should be adjusted so that there is no gap or overlap between the perturbed onsets.
- This is analogous to human musical performance.

Pseudo code

```
float preperturbed Onset[number of events]; // for known source
for (i = 1; i < number of events ; i++)
{
    perturbed onset [i] = preperturbed onset [i] + random
    number; // |random number| < duration
    duration [i - 1] = preperturbed Onset[i] – preperturbed
    Onset[i - 1];
}
```

Time Maps

- Strictly increasing function which maps from one time scale to another time scale.



```
prePerturbedOnset :=  
  prePerturbedEndTime;  
onset := TimeMap -  
  (prePerturbedOnset);  
prePerturbedEndTime :=  
  prePerturbedCurrentDuration +  
  prePerturbedOnset;  
duration :=  
  TimeMap(prePerturbedEndTime) -  
  onset;
```

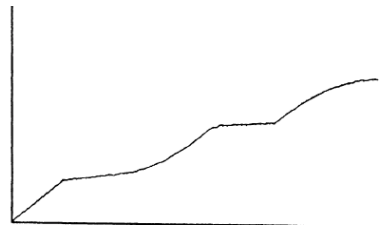
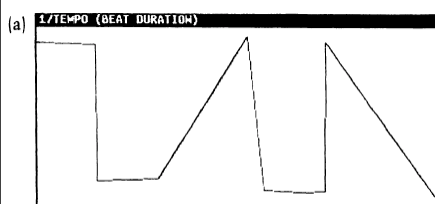
Creating Time maps

- Each segment of the time map can be described by an ordered pair(T,f)
 - T- preperturbed offset time of the segment
 - f – Function that defines the segment

```
( (10 (function (lambda (x) x)))
  (15 (function (lambda (x) (exp (/ x 2.5) 2))))
  (9999 (function (lambda (x) (* x 2))))
)
```

TEMPO ED

- A graphical tempo editor allows users to specify the tempo function interactively
- The editor automatically creates a corresponding time map on a window.



Combining Maps

- Each voice has its own independent time map.
- Time maps of multiple voices can be combined using functional composition
- Let $g(t)$, $f(t)$ be two time maps, where $g(t)$ is defined over the interval $[f(0), f(T)]$, where T is the duration of the piece .
- The composite function is give by $g(f(t))$
- The tempo at time t is give by the derivative of $g(f(t))$
- $|g(f(t))|' = g'(f(t)) * f'(t);$

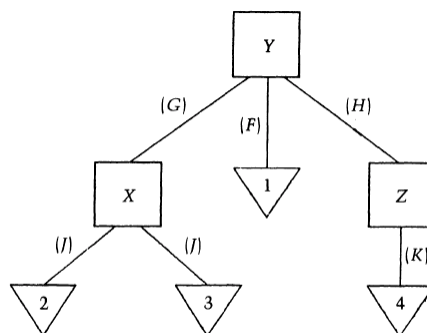
Implementation Constraints

- **Explicit Maps – Computed in advance of the performance**
 - At runtime the final composite map can be used as a lookup table
 - Can be used in real time.
- **Implicit Maps – Computed at run time**
 - First map is accessed at pre mapped time to produce a intermediate time
 - This intermediate time is accessed in the next map to produce the post mapped time.

Recursive Scheduler of Hierarchical Timing

- Combining multiple timing maps results in complex temporal dependencies and new relationships in ensemble timing.
- Requires a system that supports
 - Parallelism
 - Individual time maps for each voice
 - Ability to merge each of the maps to single voice.

Scheduler Tree



Scheduler Tree

- The scheduler is tree of process objects
- There are two types of objects
 - Voices
 - Mergers
- The terminal nodes are voices that produce events , non terminal nodes are the Mergers.
- Merges handle one or more events with their time maps.


Local warping

- Local warping is usually employed during 'given and take' style rubato
- All transformations are defined over the interval $[0, 1]$ and $f(0) = 0, f(1) = 1$
- The transformations are scaled according to the length of the segment
- Example of a Local transformation

$$f(t) = t + a \sin(bt)$$

a- controls the amount of rubato

b-controls the frequency of the deviation



The image displays a musical score for four voices: HIR, HIL, LOUW, and LOHL. The score is arranged in two systems of four staves each. The first system shows the initial notes for each voice, with HIR starting on a high note and LOHL on a lower note. The second system shows the continuation of the music, with various notes and rests for each voice. The notation includes treble clefs, a key signature of one sharp (F#), and a common time signature (C). The notes are primarily quarter and eighth notes, with some rests. The voices are labeled on the left side of each staff.

Conclusion

- Time maps provide a way to model warped time from clock time.
- It provides a system for hierarchical time warping
- Inverse time mapping may be used to find the score from the live performance