Practical Aspects of a Midi Conducting Program

Roger B. Dannenberg
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 USA
Email: dannenberg@cs.cmu.edu

Kenneth Bookstein
University of California, San Diego
La Jolla, CA 92093

ABSTRACT

A MIDI-based conducting program was implemented to allow a conductor to control the tempo of a MIDI performance that accompanies a live performer. The tempo is controlled by tapping beats on a keyboard. A number of features were added in the process of preparing for a large-scale performance, a concerto for live piano and MIDI orchestra and chorus. This experience led to a number of practical suggestions.

1. Introduction

From approximately December 1990 through March 1991, the authors collaborated closely on the development of conducting software which was used for a concert at Alice Tully Hall in New York. This concert featured the New York premiere of Ronn Yedidia's Concerto for Piano, Electronic Instruments, Choir, and Orchestra. Since neither an orchestra nor a choir were available for this concert, the plan was to perform the piece with piano and electronic instruments only. Several synthesizers and samplers, under the control of a live conductor, recreated the complete orchestral score live, including the choir and all orchestral and electronic parts.

The software, modeled after the Conductor Program of Max Mathews [Mathews 89] and accompaniment systems by the first author [Dannenberg 84], evolved to meet the requirements of a this demanding performance, and we have learned that conducting is not as simple as it seems. As we found with music synthesizers [Kaplan 81], accompaniment systems [Dannenberg 88a] and other computer music systems, the step from the lab to the concert stage is a giant one. This is due in part to the fact that researchers often "idealize" a problem or focus on only the obvious hard parts.

We do not claim that we have developed anything fundamentally new. The value of this work is the fact that it results from practical experience with professional musicians, and (to our knowledge) no detailed descriptions of the techniques we used have been published. In addition to this publication, the complete source code of our conducting system is available from the first author and will be included in the next release of the CMU MIDI Toolkit [Dannenberg 88b].

2. The Conducting Problem

The basic problem of Midi conducting is quite simple. Midi data (including timing information) is loaded into computer. A human conductor produces "beats" by tapping on a keyboard or some other device that generates Midi note messages. The messages are used to compute a tempo and a score position. The stored Midi data is output accordingly.

Perhaps it is more important to state what we do *not* consider. We do not address the problem of the physical conducting interface, that is, how to map human gestures to beat times. Others have proposed sonar sensors [Haflich 83], sensors in batons [Keane 89], computer vision [Matsushima 89], and drums [Mathews 80]. We assume only that a Midi message is delivered from the gesture-sensor to the conducting software to indicate beats, and we used a keyboard as our "gesture sensor." Secondly, we are concerned only with controlling the timing of a performance. Our system does not attempt to integrate control of dynamics or timbre with its control over tempo.

3. System Overview

Our system is structured like a computer accompaniment system [Dannenberg 89] in several respects. There are two parts to the score that is loaded into the computer. As you would expect, one part contains the music to be performed. The other part indicates where conducting beats are expected. Depending on the music, it may be desirable to conduct every quarter note, every measure, or not at all. The conductor's part tells our software when to expect beats and how to interpret them.

> ***Hint 1: the score must contain a carefully constructed conductor's part (do not assume a conducted beat on every quarter note.)***

In our system, a designated Midi channel holds the conductor's part, and all other channels hold the music to be performed[1]. Another advantage of this approach is that the score may contain parts with different tempi. Although we did not use this feature, the conductor part can be designed to correspond to any one (or none!) of these parts.

The other aspect of the system that is based on computer accompaniment is the separation between

- the "matcher," that associates incoming beat messages with the conductor's part in the score, and
- the "performer," that schedules the output of Midi data.

This separation allows the system to have a very flexible coupling between the timing *intent* expressed by the conductor, and the *realization* of that intent in terms of when notes are actually performed. Figure 1 illustrates the structure of our system.

[1]Our Amiga implementation is thus limited to 15 channels. We worked to have the Amiga send synchronization messages to a "slave" machine which would play another 16 channels, but it turned out to be faster to squeeze the original 32 channels into 15, so the "slave" system was never completed.

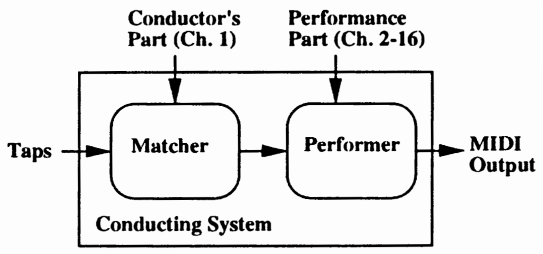

Figure 1: The conducting system consists of a Matcher that associates incoming beats with the conductor's part, and a Performer that schedules output messages.

4. Findings

Our methodology was simple: one of us developed all the code and the other developed all the Midi scores and used the system. When the system was found to be inadequate, we implemented a solution. This section reports on our findings, the solutions we found necessary to optain a musically useful system.

One set of findings is essentially that there can exist too much of a good thing. The original system assumed that the conductor would always be in control, but we found a conflict between the necessity for fine control of tempo in the slow, "expressive" passages, and the necessity in very rhythmically stable passages for the computer to be less sensitive. In these very rhythmic passages, it is extremely difficult to conduct a secure steady beat. In our system, conducting is disabled simply by omitting beats from the conductor's part so that the system does not expect any beats.

> ***Hint 2: Conducting should be disabled in some sections.***

This raises the question of what happens if the conductor forgets and beats time anyway. In our system, the default action is to jump ahead to the next downbeat in the conductor's part. This would be disasterous.

> ***Hint 3: Humans make mistakes.***

We added a simple switch to the conductor's part. Normally, the switch is on and all input is processed. When the switch is off, incoming beat messages are ignored. The switch is only turned off when no beats are expected[2].

In sections where conducting is disabled, the conductor can increase or decrease the tempo with a keystroke. Each keystroke changes the tempo by a few percent, providing a form of tempo control that is suitable for very rhythmic passages.

Hint 4: Provide tempo increment and decrement functions.

This form of tempo control can also be used in "sequencer mode," where the score plays without conducting.

Hint 5: Provide a sequencer mode that plays the score without conducting input.

It is often important to obey the tempo markings in the score. This finding has several ramifications. First, our tempo changes are "compiled" into the score so that if a tempo change is marked on a beat, the effects of that change happen immediately. This anticipates the new tempo that will be apparent when the next beat arrives from the conductor[3].

Hint 6: Anticipate the conductor by compiling tempo changes.

It is also the case that when entering a new section, it might be desirable to assume the original marked tempo rather than make the tempo relative to the previous section. One approach to this problem is to have the conducting system gradually "pull" the tempo toward the one marked in the score. In the absence of conducting input, the tempo gradually adjusts to the "right" one. This feature was disabled in the performance, however.

An alternative is to change the tempo to the "right" one immediately. Our conducting software allows special commands to be embedded in the performance part that set the tempo to nominal tempo as marked in the score.

Hint 7: Implement a command (within the score) that sets the tempo to a predefined value.

This was used at the beginning of a section where conducting is disabled to make sure that the tempo would be as planned.

CMU MIDI Toolkit scores can actually set program variables and call routines, so special commands and switches are supported by the score language. In our conducting system implementation, we also reserve note-on commands with pitch 0 and low velocity numbers to implement special controls in the score. This allows conventional sequencers with nice graphical interfaces to generate and edit conducting scores.

Hint 8: Maintain compatibility with standard formats.

There are various possibilities for determining the tempo from incoming beat messages. The simplest option is to use the difference between the last 2 beats to estimate the tempo. This does not work well because a small timing error translates to a significant tempo change. As with previous accompaniment software, we use a history buffer and take the cummulative tempo over the last several beats.

Hint 9: Tempo averaging is important.

Furthermore, large changes in the tempo estimate from the matcher are ignored or limited by the performer section.

Hint 10: Limit the sensitivity of the system to conducted tempo change.

In general, the performer can get ahead of the conductor, particularly when the conducted tempo slows down. We limit how far ahead of the conductor the performer can be (when this point is reached, the performer is stopped completely).

Hint 11: Limit how far the performance can lead the conductor.

In practice, we used an allowable lead of zero; that is, the computer waits if the expected beat does not arrive.

On the other hand, it is important to also regulate the situation in which the performer falls behind the conductor. In our current system, there is an upper bound on how fast the performer can race to catch up with the conductor. We used a "catch-up" rate of 8 times real-time, and "catch-up" mode occured when the performer fell more than

[2]In the implementation, the switch-on and switch-off commands are actually imbedded in the performed part so that conducting is switched on and off in synchrony with the performed music.

[3]We implement this by compiling the score times into millisecond time units. Performing the score at a constant rate in terms of milliseconds will then create the effect of tempo changes. An alternate approach would be to keep the score in terms of beats, but adjust the performance tempo according to the ratio of tempo changes that are encountered. This approach gets complicated if there are multiple simultaneous tempi in effect.

10 ms behind the conductor. This indicates our conductor received very quick response, but had to tap very accurately.

Hint 12: Various recovery strategies can avoid unnatural performances.

Finally, it is essential to support rehearsals. Our conductor can be started at an arbitrary time point. An interface that could jump to a particular labelled rehearsal point would be even better.

Hint 13: Facilities to support rehearsal are critical.

5. Summary and Conclusions

We implemented and tested a conducting program with a number of features that have not been reported in the literature. Most of our features make the system either more flexible or more robust. In the flexibility category, we include a conductor's part (rather than assuming every quarter note is conducted), we annotate the score with tempo changes, support rehearsal, use standard file formats, and have tempo increment and decrement controls. For robustness, we perform tempo averaging, limit the sensitivity of the system, and have recovery strategies to avoid unnatural performances.

The basic functions of a conducting program are not difficult to implement. We found that much more work is required to meet the needs of professional musicians, and we spent months developing and refining the extensions reported above. The final system works well. By comparison, we found only one commercial product that supports conducting, and it was unuseable. One area we did not investigate thouroughly is the best way to handle tempo changes. For example, Otehru reports the use of a second order model that anticipates acceleration. We did not find this necessary, but it would be interesting to perform a comparison of first-order and higher-order tempo tracking.

6. Acknowledgments

The authors would like to thank Commodore-Amiga and Carnegie Mellon for their partial support of this project.

References

[Dannenberg 84] Dannenberg, R. B. An On-Line Algorithm for Real-Time Accompaniment. In *Proceedings of the 1984 International Computer Music Conference*, pages 193-198. Computer Music Association, 1984.

[Dannenberg 88a] Dannenberg, R. B. and H. Mukaino. New Techniques for Enhanced Quality of Computer Accompaniment. In *Proceedings of the 1988 International Computer Music Conference*, pages 243-249. Computer Music Association, San Francisco, 1988.

[Dannenberg 88b] Dannenberg, R. B. *The CMU MIDI Toolkit* Carnegie Mellon University, 1988. Published by the CMU Studio for Creative Inquiry.

[Dannenberg 89] Dannenberg, R. B. Real-Time Scheduling and Computer Accompaniment. *System Development Foundation Benchmark Series. Current Directions in Computer Music Research.* In Mathews, Max V. and Pierce, John R., MIT Press, 1989, pages 225-262.

[Haflich 83] Haflich, S. M. and Burns, M. A. Following a Conductor: The Engineering of an Input Device. In *1983 International Computer Music Conference Proceedings.* Computer Music Association, 1983.

[Kaplan 81] Kaplan, J. S. Developing a Commercial Digital Sound Synthesizer. *Computer Music Journal* 5(3):62-73, Fall, 1981.

[Keane 89] Keane, D. and P. Gross. The MIDI Baton. In *Proceedings of the 1989 International Computer Music Conference*, pages 151-154. Computer Music Association, 1989.

[Mathews 80] Mathews, Max V. and Abbot, Curtis. The Sequential Drum. *Computer Music Journal* 4(4):45-59, Winter, 1980.

[Mathews 89] Mathews, M. V. The Conductor Program and Mechanical Baton. *System Development Foundation Benchmark Series. Current Directions in Computer Music Research.* In Mathews, Max V. and Pierce, John R., MIT Press, 1989, pages 263-281.

[Matsushima 89] Matsushima, T., S. Ohteru, S. Hashimoto. An Integrated Music Information Processing System: PSB-er. In *Proceedings of the 1989 International Computer Music Conference*, pages 191-198. Computer Music Association, 1989.