Automated Accompaniment of Musical Ensembles

Lorin Grubb and Roger B. Dannenberg

School of Computer Science, Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
lgrubb@cs.cmu.edu and rbd@cs.cmu.edu

Abstract[1]

This paper describes a computer accompaniment system capable of providing musical accompaniment for an ensemble of performers. The system tracks the performance of each musician in the ensemble to determine current score location and tempo of the ensemble. "Missing parts" in the composition (i.e., the accompaniment) are synthesized and synchronized to the ensemble. The paper presents an overview of the component problems of automated musical accompaniment and discusses solutions and their implementation. The system has been tested with solo performers as well as ensembles having as many as three performers.

Introduction

Musical performance in ensembles requires more than just mastery of an instrument. Ensemble performers must be able to listen to one another and react to changes in the performance. These include changes in tempo and loudness, for example. Even following the flow of performance can be difficult as instruments drop out and re-enter. Each performer must synchronize with the ensemble, that is, play at the appropriate time and with the appropriate tempo. The computer accompaniment problem is to track, in real-time, a performance by a solo musician or a group of musicians and to produce the "missing" voices of the composition in synchrony with the other performers. It is assumed that all performers, including the computer, play from a score that specifies the pitches and their musical timing. An ensemble accompaniment system is a flexible alternative to playing along with a recording when a full ensemble is not available. It also enables new compositions combining human and machine performers.

We describe a system developed to provide accompaniment for an ensemble. Several computer accompaniment systems for following *solo* performers have been described (Dannenberg 1984, Vercoe 1984, Baird et. al. 1993). In (Dannenberg 1984) the accompaniment problem is partitioned into three distinct subproblems: 1) detecting what the soloist has performed, 2) determining the score position of the soloist from the detected performance, and 3) producing an accompaniment in synchrony with the detected performance. A solution for each subproblem and a method for its implementation is also provided.

The system presented here extends the capabilities of previous accompaniment systems by performing with multiple musicians. To follow an ensemble, the solutions to the first and second subproblems in the solo accompaniment system must be simultaneously applied to multiple performers. Before taking actions to control an accompaniment (the solution to the third subproblem), the ensemble system must examine and combine the score position and tempo suggestions produced through tracking multiple performers. Note that this may require resolution of conflicting information. This paper details specifics of the problem of tracking multiple performers and combining the results in order to produce an accompaniment. A solution to the problem and a brief description of its implementation are provided. We conclude with a discussion of some qualitative results of actually using the system to accompany multiple performers.

Problem Description

The performance-monitoring component of a computer accompaniment system must process a digital representation of the performance in order to extract required parameters. These parameters might include fundamental pitch, note duration, dynamic (relative loudness), and articulation. The representation of the performance, in the simplest case, might be MIDI (Musical Instrument Digital Interface) messages (Rothstein 1992) sent from an electronic keyboard. Since most of the required parameters are explicitly represented in MIDI messages, extracting the needed information is simple. In a more difficult case, the representation is an audio signal from a microphone. This is often the representation obtained from wind instruments, for example. Extracting musical parameters from this type of input requires more analysis than does processing MIDI messages. Fortunately, there exist commercially available devices for converting analog input into MIDI-encoded pitch and amplitude information. The most difficult case is multiple instruments recorded with a single microphone. We do not address this case.

Once basic parameters (like pitch) have been extracted from the performance, the next task for accompaniment systems is to estimate the performer's location in the score.

1. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship

This involves comparing the sequence of extracted performance parameters to the expected sequence of parameters (the given score), attempting to find the best match. A robust pattern matcher is required because a perfect performance cannot guaranteed.

As successive score locations are identified and time stamped, tempo is estimated by comparing the actual time difference between performed events and the expected time difference between the corresponding score events. Since it is well-known that performers alter durations of particular beats within a measure for expressive purposes (Desain & Honing 1992), the accompaniment system must average successive time difference comparisons to avoid sudden drastic tempo changes.

Accompaniment systems also produce the actual accompaniment. Generally, an accompaniment must be continuous and aesthetically acceptable, yet reactive to the performer's omissions, errors, and tempo changes. If the performer increases or decreases the tempo slightly for interpretive reasons, the accompaniment should do likewise. If the performer pauses or jumps ahead in the score, the accompaniment should follow as much as possible, but the accompaniment should always sound "musical" rather than "mechanical". Thus, a considerable number of decisions must be made by the accompaniment generation component in response to tempo and location information.

The task of estimating score location and tempo in an ensemble accompaniment system is complicated by the fact that multiple performers must be tracked and their individual score locations and tempi combined and resolved. Several considerations affect this resolution process. For instance, if one performer's tracking system is having difficulty tracking the performance, possibly because of signal processing difficulties or because the performer is making mistakes, then the estimates from that tracking system should not strongly affect the final estimation. Also, performers who become inactive for a relatively long duration (i.e., have a rest or a sustained pitch) should affect the final estimations less than recently active voices, which are more likely to indicate the current tempo and score position. Additionally, a performer whose score position is relatively distant from the majority of the ensemble, presumably indicating that this performer is lost or has fallen behind, should be ignored. A combination of the tracking systems' estimates must satisfy these considerations as much as possible in order to produce an accurate, unified ensemble score location and tempo.

Approach

To identify the score location of a single performer, we use a modified version of the dynamic programming algorithm for identifying the longest common subsequence of two strings (Cormen et. al. 1990). Regarding the performance as a sequence of pitches, the objective is to delete a minimal number of notes from performance and score sequences to obtain an exact match. In practice, a prefix (the performance) is matched against a complete string (the score), and only a portion of the score is examined in order to save time.

The matching algorithm is applied on every recognized note in the performance. The objective is to find the "best" match between performance and score according to the evaluation function:

$$evaluation = a \times matched\ notes - b \times omissions - c \times extra\ notes$$

Although the number of ways the performed pitches can be matched against the score is exponential in the number of performed notes, dynamic programming allows us to compute the best match in time that is linear in the length of the score, and which gives a result after each performed note. By using a "window" centered around the expected score location, the work per performed note is further reduced to a constant. A more detailed presentation of this algorithm can be found in (Bloch & Dannenberg 1985), which also shows how to modify this algorithm to handle polyphonic performance input (e.g., chords played on a keyboard).

If a new score location has been posited, it is placed in a buffer along with a timestamp indicating the "real time" when that location was reached by the performer. If one views these buffer entries as points in a graph mapping real time of the performance on the abscissa to "score time" (the position in the score) on the ordinate, the tempo of the performance at any instant is given by the slope of the graph (since tempo is the amount of score traversed in a unit of real time). Figure 1 graphs a tracked performance. Since performers are noticeably inconsistent within a tempo, it is necessary to apply some form of averaging in order to avoid occasional drastic tempo change estimates. Although many averaging techniques are available, we have elected to simply take the slope of the line between the first and last points in the location buffer. Since the buffer size is limited and relatively small, with older entries discarded one at a time once the buffer's capacity is exceeded, this tempo estimation is still responsive to actual changes in tempo but less "jerky" than estimates based solely on the two most recent entries in the buffer. If the matching algorithm detects a performance error, the buffer is emptied and no tempo or score position estimates are possible until several correct notes are played.

For an ensemble accompaniment system which must track multiple performers simultaneously, separate instances of the match state and the score location buffer must be maintained. Notes from different performers are identified by different MIDI channels, and the appropriate state is updated. Since score location information for each performer is available, it is possible to estimate each performer's current score location at any time. For example, consider Figure 2. If at time t_1, the performer is at score location s1 and maintaining an estimated tempo of 0.5; then at time t_1+2, the performer's expected score location would be s_1+1 (if no intervening input is received from the performer).

The various estimates must eventually be consolidated into a single ensemble score position and tempo. The accompaniment system estimates an ensemble score position and tempo on every input from every performer. To accomplish this in accordance with the criteria presented in

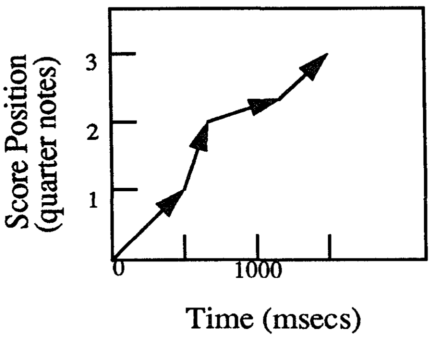

Figure 1,
Example performance graph.

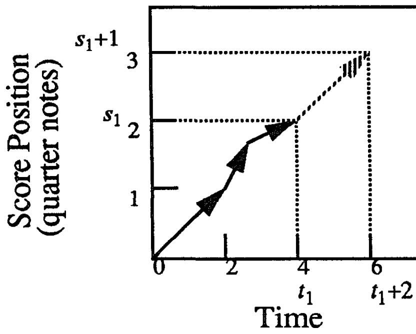

Figure 2,
Estimated score position.

section two, each pair of estimations from each tracking system is rated, and a weighted average is computed from both score location and tempo estimates. The ratings are constructed so that an estimate is given more weight when it is more recent, and less weight when it does not cluster with other estimates. Figure 3 presents the rating function. The final rating (FR) used for the weighted average is the product of the squares of two independent ratings—a recency rating (RR) and a clustering rating (CR).

The recency rating for each tracking system (as given in Figure 3) decays from a value of one to zero during a three-second interval. If the score-position buffer of the tracking system is empty, then the recency rating is zero. This value is squared in the final rating product, causing the final rating to decay more rapidly (in a quasi-exponential fashion) over the three-second interval. The rating is designed to give preference to the most recently active performers, thereby making the accompaniment performance more reactive. The clustering rating (also given in Figure 3) characterizes the relative separation of voices. It is the ratio of the summed distance of the i'th voice from all other voices, divided by the maximum possible summed distance at the time of rating. The rating decays from a value of one for the best case (i.e., all voices and the accompaniment are at identical score positions) to a value of zero in the worst case (i.e., all voices except the i'th voice are at the same position). This rating, like the recency rating, is squared in the final rating so as to give an even stronger preference to voices which are "relatively" tightly clustered. The final rating is a product of the squares of the other ratings so that it is guaranteed to be less than or equal to the minimum of the individual squares (since each square ranges from zero to one). Thus, as the criteria characterized by the component ratings fail to be satisfied, the final rating decreases.

The ensemble score position and tempo are calculated as a weighted average of the tracking system estimates. Each estimate is weighted by its final rating, and thus affects the overall estimate according to its "relative" satisfaction of the specified criteria compared to the estimates of the other tracking systems. For example, consider a performance of the section of score presented in Figure 4. As the first performer proceeds, the recency rating of the other voices will decay. The tempo and score position estimated by the first performer's tracking system will quickly dominate the ensemble average, causing the accompaniment to more closely follow that performer.

Once the ensemble score position and tempo estimates have been calculated, they are used to make adjustments to the accompaniment performance according to a set of accompaniment rules. These rules are based upon studies of how real accompanists react to similar situations encountered during a performance (Mecca 1993). First, the time difference between the ensemble score position and the accompaniment score position is calculated. If the time difference is less than a pre-determined "noise" threshold, then the accompaniment tempo is simply set to the ensemble tempo. The noise threshold is used to prevent excessive jumping and tempo alterations, since performers do make subtle alterations in note placement (Bilmes 1992). This threshold is adjustable but is generally set to around 100 msecs. If the performer is ahead of the accompaniment by a difference at least as great as the noise threshold, but less than the "jump" threshold, then the accompaniment tempo is increased to an abnormally fast tempo (even faster than the actual ensemble performance) so that the accompaniment will catch-up to the ensemble. If the accompaniment catches the ensemble prior to calculating another estimate, the tempo is reset to the previous tempo estimate. The jump threshold indicates how large a score position difference is too large to bother trying to perform in order to catch the ensemble. If the time difference is at least as great as this threshold, then the accompaniment system will skip to the estimated ensemble score position and start using the estimated ensemble tempo. Finally, if the performer is behind the accompaniment by a time difference at least as great as the noise threshold, then the accompaniment will pause until the performer reaches the accompaniment's current position. To prevent the accompaniment from continuing too far ahead of the performers, it is necessary to maintain an input expectation point. If this point is passed without additional input from any performer, then the accompaniment system pauses until additional input is received. Once input is received, the score position and tempo are estimated, necessary alterations to the performance parameters are implemented as just described, and the accompaniment is restarted.

$$\text{Ensemble Score Position} = \frac{\sum_{i=1}^{n} FR(i) \times pos(i)}{\sum_{i=1}^{n} FR(i)}$$

$$FR(i) = (RR(i))^2 \times (CR(i))^2 + c$$

FR(i) = Final rating for estimate from tracking system i
RR(i) = Recency rating for estimate from tracking system i
CR(i) = Clustering rating for estimate from tracking system i
c = Very small constant to prevent FR from reaching zero

$$RR(i) = \begin{cases} 1 - \dfrac{(rtime - ltime(i))}{3} & (rtime - ltime(i)) \le 3 \\ 0 \quad \text{if} & (rtime - ltime(i)) > 3 \end{cases}$$

rtime = Current time for which estimates are made
ltime(i) = Time of last match made by tracking system i

$$CR(i) = 1 - \frac{\left(\sum_{j=1}^{n} |pos(i) - pos(j)|\right) - |acc - pos(i)|}{n \times (pos(max) - pos(min))}$$

n = Number of active tracking systems
pos(i) = Score position for tracking system i
pos(j) = Score position for tracking system j
acc = Score position for accompaniment
pos(max) = Maximum of all pos(i), pos(j), and acc
pos(min) = Minimum of all pos(i), pos(j), and acc but NOT pos(max)

Figure 3, Function to calculate ensemble score position.

Implementation

The ensemble accompaniment system is constructed using the CMU MIDI Toolkit (Dannenberg 1993) which provides MIDI message handling, real-time scheduling, and performance of MIDI sequences. It is possible to adjust the position and tempo of a sequence (score) performance on-the-fly as part of processing input or generating output. The system is written in C and runs on DOS-based PCs, Macintoshes, Amigas, and Unix workstations. The ensemble accompaniment system has been tested on both a DOS-based system and an Amiga.
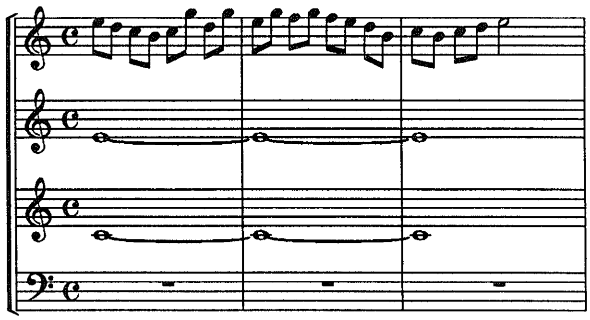
Figure 4, Score excerpt.

To obtain performance input, both MIDI keyboards and pitch-to-MIDI converters have been used. The keyboards themselves generate MIDI messages which can be directly sent to the ensemble accompaniment system. The pitch-to-MIDI converter is an IVL Pitchrider, designed for use primarily with wind instruments. It takes input directly from a microphone, analyzes the input to identify fundamental pitch and attack, and generates MIDI messages that can be sent to the ensemble accompaniment system. For wind instruments, the data from this device can be used by the accompaniment system without modification. A software preprocessor has been developed to further analyze the data sent from this device when receiving vocal input (singing).

The ensemble system consists of four software components: a matcher, which receives input from a performance and uses dynamic programming to determine score location; an estimator, which maintains the score location buffer, calculates tempi, and generates estimates on request; a voter, which rates and combines multiple score location and tempo estimates into an ensemble location and tempo; and a scheduler, which uses the ensemble estimates to change the accompaniment performance according to the rules described in section three. A matcher-estimator combination forms a single performance tracking system. The accompaniment system may instantiate multiple tracking systems at initialization depending on user-supplied specifications. Only one voter and one scheduler are ever present. Figure 5 diagrams the interaction between these components for the case when the system is to accompany an ensemble of two performers.

When MIDI messages are received as input, the appropriate matcher is invoked according to the channel number of the message. (Each performer's input must be sent on a separate MIDI channel.) The matcher applies the dynamic programming algorithm to determine the score location of the performer. The success or failure of this process is passed along to the estimator. If the match was successful then the estimator will pass its score position and tempo estimates to the voter. The voter will then request similar estimates from each of the other active tracking systems. The ratings are then generated for each set of estimates, as previously described. When an ensemble score position

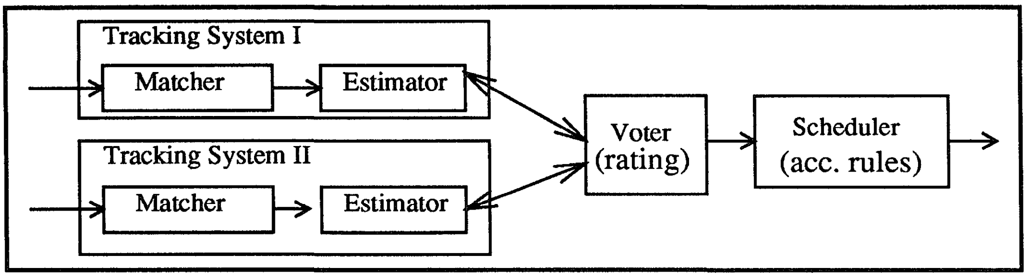

Figure 5, Components of the accompaniment system.

and tempo have been generated, they are passed to the scheduler which determines any necessary and appropriate modifications to the accompaniment performance, according to the rules presented in the previous section. Our toolkit automatically manages the activation of input handling routines and MIDI message generation for performance of the accompaniment. The MIDI output can then be sent to a synthesizer for sound production.

Results

The ensemble accompaniment system has performed with ensembles of one, two, and three players consisting of both MIDI keyboards and acoustic wind instruments. The pieces used for testing range in difficulty from a simple canon on "Row, row, row your boat" to excerpts from Handel's *Water Music*. In the case of a single performer, the system functions exactly the same as the solo accompaniment system previously constructed. It is highly reactive to tempo changes of the soloist and tolerant of omitted notes, wrong notes, and extra notes. If too many wrong notes or extra notes appear in the performance (as in a heavily embellished rendition), the matcher becomes unable to recognize the part, but the accompaniment will continue according to the last tempo estimate. The occasional mistake from a competent performer does not present a problem. In the case of omitted notes (such as when the performer jumps ahead), the system will ignore the performer until enough notes are correctly matched so that the matcher's score overcomes the penalty imposed by the skipped notes. The farther the performer jumps, the larger is the penalty and the corresponding delay. Note that this penalty increases only for notes skipped by the performer—if the soloist omits a rest during which time the accompaniment plays, no penalty is generated and the accompaniment almost immediately re-synchronizes with the performer.

In the case of an ensemble of two performers, the system is able to simultaneously track both performers. If either performer drops out, the system continues to accompany the other. Tempo changes of the latter performer are recognized by the system—readily so, once the silent performer's recency rating has decayed. Also, since the score position of the accompaniment affects the clustering rating for each performer, if one performer should skip ahead or fall behind, the system will continue to synchronize with the other performer—ignoring the "lost" performer until he or she rejoins the ensemble or until the first performer stops or becomes lost. If both performers skip ahead or change tempo in synchrony, then the accompaniment does likewise according to the accompaniment rules described in section three. The system acts similarly in the case of three performers. In addition, should two of the three performers jump ahead or fall behind to the same position in the score, leaving only one performer in synchrony with the accompaniment, the system will quickly re-synchronize with the two performers since they represent the majority of the ensemble. Note also that when tracking multiple performers, the accompaniment is less affected by a single performer playing wrong notes or omitting notes, providing the other performers are accurate and in synchrony.

While the system works well with small ensembles, several problems must be addressed in order for the system to perform with larger ensembles. Two problems at the input level are that MIDI only permits sixteen logical channels and that individual microphones in large ensembles will experience cross-talk, making pitch estimation more difficult. There is no reason to believe that either of these problems is insurmountable.

Compute time is another consideration. Processing an input requires time linear in the ensemble size (since estimates from all tracking systems must be re-rated). In the worst case, if all parts simultaneously play a note, the amount of work to be completed before the next note in the score is performed is quadratic in the ensemble size. On the slowest PC used, handling a single input requires 1.4 msec. for an ensemble of one. The expense of recomputing one rating (for larger ensembles) is 0.3 msec. Based on these numbers, a conservative estimate indicates that we can process 16 inputs in 100 msec. A sixteenth note of 100 msec. duration implies a tempo of 150 quarter notes per minute. This is a fast tempo. If we were to update the voter once every 100ms instead of on each input, we could handle hundreds of instruments in real time with current processor technology. For large acoustic ensembles, the computation will be dominated by signal processing of acoustic input.

Conclusions

Developing this accompaniment system has helped to define important criteria and considerations relevant to ensemble accompaniment. When generating score location and tempo estimates for an ensemble, it is useful to consider both the recency of the input from individual performers and the clustering, or relative proximity, among the performer's score positions. This information assists in distinguishing recent and reliable performer input from that which has come from a lost or resting performer, or one who is not following the score.

Construction and testing of this system has demonstrated there exists a trade-off between reactivity and stability of an accompaniment. As previously indicated, the ensemble accompaniment system currently attempts to be reactive to the performers. For example, in the case of three performers where two performers have jumped ahead in the score but one has remained with the accompaniment, the system will quickly jump to the score location of the ensemble majority. This reactivity could be questioned, since in some cases maintaining a stable accompaniment that does not skip ahead with the majority might be preferred. A more stable accompaniment might also be desired if the majority of the ensemble is consistently dragging the tempo, as opposed to changing tempo for expressive purposes. This trade-off must be considered during construction of both the rating functions used to calculate the ensemble position and tempo, and the rules used to determine when to change tempo and score position of the accompaniment.

Although the ensemble accompaniment system generally reacts to performance situations in a reasonable manner, there remain some questionable actions which might be improved. Some of these are related to the reactivity-stability trade-off just mentioned. One example is the placement of expectation points used to pause the accompaniment if no performer input is received. The more frequently these points are placed, the more reactive to tempo reductions and missed entrances the system becomes. The more sparse their placement, the more stable the accompaniment and the more performers are forced to compensate for their own mistakes. The use of knowledge-based rules to better define the relative rating of each performer's score location and tempo is also a consideration. If it is clear from the score that a particular performer should be inactive at present, then perhaps that performer's estimates should be ignored. This might make the system more immediately reactive to the contrapuntally active performers, as opposed to waiting for the inactive performer's recency rating to decay. It is hoped that further experimentation with the present system will help to define a more comprehensive understanding of these trade-offs and alternatives, leading to a more versatile accompaniment system.

Additionally, pre-performance analysis of the score might help develop useful performance expectations. Annotations in scores can provide useful performance hints to the scheduler (Dannenberg & Bookstein 1991). Automating this annotation process would, however, require significant musical knowledge pertinent to interpreting scores. Alternatively, we are also interested in experimenting with learning through rehearsal, possibly by using techniques similar to those presented in (Vercoe & Puckette 1985). Ideally, an accompaniment system should be able to improve by practicing a piece with an ensemble and noting where to expect consistent tempo changes, embellishment, or performer error. This could be done with a single ensemble and single score, as well as with a single score performed by multiple ensembles. Since even two expressive performances by the same ensemble may vary greatly, the challenge will be to extract reliable characterizations of multiple ensemble performances and use them to enhance the accompaniment in successive performances, beyond what the naive and score-independent expectations allow. Using our current system as a starting point, techniques for effectively learning performance nuances can now be explored.

References

Baird, B., Blevins, D., and Zahler, N. 1993. Artificial intelligence and music: implementing an interactive computer performer. *Computer Music Journal* 17(2): 73-9.

Bilmes, J. 1992. A model for musical rhythm. In Proceedings of the 1992 International Computer Music Conference, 207-10.

Bloch, J. and Dannenberg, R. 1985. Real-time computer accompaniment of keyboard performances. In Proceedings of the 1985 International Computer Music Conference, 279-90.

Cormen, T., Leiserson, C., and Rivest, R. 1990. *Introduction to Algorithms*, 314-19. New York: McGraw-Hill Book Co.

Dannenberg, R. 1984. An on-line algorithm for real-time accompaniment. In Proceedings of the 1984 International Computer Music Conference, 193-8.

Dannenberg, R., and Bookstein, K. 1991. Conducting. In Proceedings of the 1991 International Computer Music Conference, 537-40.

Dannenberg, R. 1993. *The CMU MIDI Toolkit*. Pittsburgh: Carnegie Mellon University.

Desain, P., and Honing, H. 1992. Tempo curves considered harmful. In *Music, Mind, and Machine: Studies in Computer Music, Music Cognition, and Artificial Intelligence*, 25-40. Amsterdam, Thesis Publishers.

Mecca, M. 1993. Tempo following behavior in musical accompaniment. Master's thesis, Carnegie Mellon University.

Rothstein, J. 1992. *MIDI: A Comprehensive Introduction*. Madison: A-R Editions.

Vercoe, B. 1984. The synthetic performer in the context of live performance. In Proceedings of the 1984 International Computer Music Conference, 199-200.

Vercoe, B and Puckette, M. 1985. Synthetic rehearsal: training the synthetic performer. In Proceedings of the 1985 International Computer Music Conference, 275-78.