Although difficult, it would also be ideal to learn specific production rules for a grammar like ours from data sets in addition to learning the production probabilities. For our grammar, we hand-picked rules using examples of analyzed phrases taken from existing music (namely analyzed examples of Bach) along with our own intuition. It may be possible to automate this process and find phrases that would make good rules by searching through an analyzed data set. GTTM-inspired phrase segmentation similar to the method implemented by Ito et al. [12] may be a useful step in this process.

Finally, the use of chord spaces as a structural aid is one of the most promising features of our implementation, and it could be extended. More complex musical structures, such as the subject, answer, and counter-subject restrictions of fugue form, could be encoded as constraints used by the chord space framework we used in our grammar’s strings, or harmonic phrases, into more elaborate, concrete music. Chord spaces at different levels of abstraction could also be used to alter the style of the music. For example, the Roman numerals generated by a grammar for classical music could be interpreted through a chord space built for jazz to give richer harmonies.

Acknowledgements This research was supported in part by NSF grant CCF-0811665.

7. REFERENCES
that is intended to be used for pitch-tracking or gesture-tracking of the live performance.

Before the performance begins, those staves containing pitch-tracking or gesture-tracking data are passed (though network UDP messages) to a MaxMSP patch that is responsible for tracking the live performance. During performance, one or more streams of scores are being performed. The MaxMSP patch interprets the various input values of various gesture and track time and sends synchronization messages back to NoteAbilityPro. The synchronization messages keep NoteAbilityPro’s playback closely aligned to the live performances so that all the sound and control events embedded in the score are performed at the correct time. The live instruments can also be processed and mixed with other electroacoustic sounds, all of which can be controlled by messages sent to MaxMSP or Pd from the score.

In addition to standard playback methods such as MIDI, NoteAbilityPro has developed a generalized gesture-tracking module in MaxMSP based on the gf [2] and imubu [19] objects developed at IRCAM. Our tracker allows up to six simultaneous channels of gesture data to be recorded and later followed during the performance. In our implementation, a gesture is considered to be any continuous data that can be replicated during performance. Examples of these are musical passages with a reasonable degree of accuracy. Examples of gestures that have been tested are: amplitude and frequency contours of a clarinet performance, viola bow motion, the movements of a dancer, conductor cues and the hand movements of a pianist. These gestures are recorded in rehearsal using amplitude trackers, spectral analysis objects, accelerometers and video tracking methods. They are then followed in performance using the same tracking objects and the data is passed to gf objects which are designed to compare the incoming information to the previously recorded model. The gf object returns the speed of the gesture (relative to the original recorded model) and the current tracking position.

The gestures are recorded as multichannel BPFs at a fixed sampling rate, filtered to remove jitter, placed in the score, and aligned to the other score events by adjusting markers that have been added to the BPF. Before performance begins, all the gestures are sent from the score to the gesture-tracking MaxMSP patch where they are loaded into gf objects. During performance, the score sends network messages to the patch whenever it reaches the beginning of an embedded gesture on a gesture-following staff. The gesture is loaded into the corresponding track in the patch and the object begins tracking the incoming data. The gesture-tracker calculates the score position based on the contour of the gesture and sends synchronization messages back to the score. Since the gf object uses a Hidden Markov Model (HMM) to compare incoming data to the stored model, it tolerates timing deviations easily, which makes it ideal for tracking live performances where tempos might vary dramatically from performance to performance. During rehearsals, it is possible to adjust the tolerance of the gesture-tracker and relative weighting between channels of a multichannel gesture.

In Figure 4, we see a gesture containing the clarinet amplitude (top), pitch (middle) and spectral centroid (bottom) that were recorded and aligned to the score. In the performance, we are able to accurately follow the decrescendo of the last note in the phrase by gesture-tracking the amplitude data taken from the clarinet performance.

In Figure 5, we see the gesture data recorded from the bow motion of a viola using a 3-axis accelerometer in a Texas Instruments EZ-430-Chronos [22] watch on the wrist of the bow hand of the performer. Using this data, the bow motion can be tracked easily, and even if wrong notes are played, the gesture-tracker will follow the performance accurately.
that is intended to be used for pitch-tracking or gesture-tracking of the live performance. Before the performance begins, those staves containing pitch-tracking or gesture-tracking data are passed (though network UDP messages) to a MaxMSP patch that is responsible for tracking the live performance. During performance, one or more streams of score-following data is performed. The MaxMSP patch negotiates between the various pitch and gesture trackers and sends synchronization messages back to NoteAbilityPro. The synchronization messages keep NoteAbilityPro’s playback closely aligned to the live performances so that all the sound and control events embedded in the score are performed at the correct time. The live instruments can also be processed and mixed with other electroacoustic sounds, all of which can be controlled by messages sent to MaxMSP or Pd from the score.

Figure 1. MaxSE system overview

4. PLAYBACK EXTENSIONS

In addition to standard playback methods such as MIDI, internal DLS instruments, Audio Units and sound files, a NoteAbilityPro score can send both discrete and continuous control messages to MaxMSP or Pd during performance. Discrete messages are placed in text boxes and adopt the standard format used in MaxMSP and Pd—each message contains a receive name followed by the arguments to be sent to that receive and ends with a semicolon and a carriage return. In MaxMSP these objects or can be send directly to an editor or they can be generated using the function object in MaxMSP, dumped from the function object as a text file and dragged into the score. Markers can also be added to the BPFs. The name associated with the BPF is the name of the receive in MaxMSP or Pd to which the continuous data is sent. In the case of multi-channel BPFs, each successive channel will connect to a receive with a “_n” extension to the name. The score example in Figure 2 shows both discrete messages and a multichannel BPF for continuous controls. In MaxMSP or Pd, this score will send messages to objects named: [r startup], [r masterVol], [r harm] (the first channel of the BPF) and [r harm_1] (the second channel of the BPF).

5. SCORE-FOLLOWING

5.1. Pitch-tracking

In order to facilitate pitch-tracking of monophonic or polyphonic instruments, NoteAbilityPro can generate files required by antescofo—All the necessary tracking information (including pitches, trills, glissandi, and tempo changes) from all relevant staves is included in this file. Several copies of antescofo can be running simultaneously, each tracking a different instrument. If a high degree of precision between the tracked performance and the MaxMSP or Pd events is required, antescofo—GFWD action messages can also be embedded in the antescofo file. GFWD messages can be entered in the score as text or as notes that have been converted to action messages. In Figure 3, the messages and notes in the second staff are encoded as action messages and the corresponding events will be triggered directly by antescofo during score-following.

Figure 2. Discrete and continuous control messages in NoteAbilityPro

5.2. Gesture-tracking

There are some instances where pitch-tracking is not feasible. Examples of these are musical passages with aleatoric elements, passages using extended instrumental techniques where the pitch content is variable, passages where amplitude or timbre changes on a sustained pitch need to be tracked and complex music for polyphonic instruments. For these situations, gesture-tracking has proven to be a reliable alternative to pitch-tracking. We developed a generalized gesture-tracking module in MaxMSP based on the gf [2] and imbed [19] objects developed at IRCAM. Our tracker allows up to six simultaneous channels of gesture data to be recorded and later followed during the performance.

In our implementation, a gesture is considered to be any continuous data that can be replicated during performance with a reasonable degree of accuracy. Examples of gestures that we have tested are: amplitude and frequency contours of a clarinet performance, viola bow motion, the movements of a dancer, conductor cues and the hand movements of a pianist. These gestures are recorded in real time using multichannel trackers, spectral analysis objects, accelerometers and video tracking methods; they are then followed in performance using the same tracking objects and the data is passed to gf objects which are designed to compare the incoming information to the previously recorded model. The gf object returns the speed of the gesture (relative to the original recorded model) and the current tracking position.

The gestures are recorded as multichannel BPFs at a fixed sampling rate, filtered to remove noise, placed in the score, and aligned to the other score events by adjusting markers that have been added to the BPF. Before the performance begins, all the gestures are sent from the score to the gesture-tracking MaxMSP patch where they are loaded into gf objects. During performance, the score sends network messages to the patch whenever it reaches the beginning of an embedded gesture on a gesture-following staff. The gesture is loaded into the corresponding track in the patch and the gf object begins tracking the incoming data. The gesture-tracker calculates the score position based on the contour of the gesture and sends synchronization messages back to the score. Since the gf object uses a Hidden Markov Model (HMM) to compare incoming data to the stored model, it tolerates timing deviations easily, which makes it ideal for tracking live performances where tempos might vary dramatically from performance to performance. During rehearsals, it is possible to adjust the tolerance of the gesture-tracker and relative weighting between channels of a multichannel gesture. In Figure 4, we see a gesture containing the clarinet amplitude (top), pitch (middle) and spectral centroid (bottom) that were recorded and aligned to the score. In performance, we are able to accurately follow the decrescendo of the last note in the phrase by gesture-tracking the amplitude data taken from the clarinet performance.

Figure 5. BPF Staff containing gesture performance data.

In Figure 5, we see the gesture data recorded from the bow motion of a viola using a 3-axis accelerometer in a Texas Instruments EZ-430-Chronos [22] watch on the wrist of the bow hand of the performer. Using this data, the bow motion can be tracked easily, and even if wrong notes are played, the gesture-tracker will follow the performance accurately.
6. VIDEO TRACKING OF PIANO HANDS

In some cases, it is advantageous to perform both pitch-tracking and gesture-tracking on the same instrument and to use both streams of data to improve the score-following accuracy. The piano, as a polyphonic instrument, is a good candidate for an instrument for which both modes of tracking are useful. Pitch-tracking is sometimes unreliable, especially in dense passages or when the sustain pedal is down. In such situations, we have found that tracking the movements of the performer’s hands (in conjunction with pitch-tracking) provides us with much more reliable score-following. In order to track the movement of the pianist’s hands, a camera is mounted above the piano in such a way that the entire keyboard can be captured. While any webcam can be used, our tests were carried out using a PlayStation® Eye [15] camera and a Logitech® C920 webcam [11]. With other instances of gesture-tracking, the hand movements of a rehearsal are recorded and aligned to the score. The performance is then tracked using approximately the same camera placement and the performance gestures are compared to the recorded model using our gesture-tracking patch.

A Jitter module was designed to analyze the video image from the camera and to produce two channels of continuous data that follow the horizontal movements of the two hands. We are not concerned with individual finger movement, just the relative positions of the hands as they move up and down the keyboard. As in the case of viola bow tracking, wrong notes are tolerated provided that the hands move close to their correct positions at more or less the correct time.

The amount of data produced by a camera per frame is formidable (640x480x3 = 921,600 pixels per frame). To achieve reliable tracking with a minimal footprint on the CPU, several steps are taken to reduce the data as much as possible. The video is cropped so that only the portion of the frame that contains the keyboard is analyzed – this reduces the frame to approximately 640x90x3 (172,800 pixels). As an added benefit, the performer’s arms and body are cut from the frame and this reduces the overall movement within the frame. Next, we eliminate all color information from the video, since color has no impact on the tracking algorithm. The conversion from RGB to luminance reduces the data to 640x90x1 (57,600 pixels per frame) – a reduction of 93.75% from the original data size.

6.1. Mean-Shift / Cam-Shift Overview

The mean-shift algorithm looks for peaks or extrema in the density (i.e. on-pixel distribution) of a specified window of any given frame. The center mass for that window is computed and then the window is centered on that mass. The window is computed again and the window moved to the new center mass location. If the window stops moving a new peak in the frame has been reached and the tracking algorithm is restarted. At any point the algorithm is continued on the next input frame.

The cam-shift algorithm works using the same principle. However, with this approach the search window will adjust in size to accommodate pixel spread (i.e. an object moving closer to or away from the camera).

In order to isolate the hand movements, a background subtraction is performed to eliminate the keyboard from the video image. Several frames of the keyboard alone without the performer present are recorded, and this recording is averaged to remove any minute lighting differences. The result is subtracted from each incoming frame, which results in an image containing only the differences between the current video frame and the subtracted image, i.e. the hands only. An optional last step is to create a binary image by setting and applying a luminance threshold. All pixels above the threshold are set to one pixel and all those below the threshold to zero pixels. By fine-tuning the threshold, further minute fluctuations in the image can be removed.

A “mean-shift” algorithm performs the bulk of the actual tracking. This algorithm is implemented in the Open CV [3] and was ported to MaxMSP/Jitter by Jean-Marc Pelletier [14]. Before the mean-shift algorithm can track the performance, a blob tracking routine is run to determine if there is any major movement within the frame. The minimum blob-size is set to a high value so that any variations in lighting that were not eliminated by the previous filtering methods or other unwanted movements (e.g. shadows cast by the performer while moving the upper body, keyboard shifts during the use of una corda pedal, etc.) are not detected as hands. Once a blob of a certain size is identified, its bounding rectangle can be used to initialize the mean-shift algorithm. The object that performs the mean-shift calculations has two distinct operational modes: 1) mean-shift tracking and 2) continuously adaptive mean-shift tracking (cam-shift).

There is one special case where the system “breaks” in terms of human observable right/left ordering versus computed right/left ordering. If the pianist uses hand crossings, the results observed by a human and the machine are not consistent. For example, a human observer will easily recognize the gesture and identify that the left hand crossed over the right. In contrast, the computer sees that the articulation windows have collided and reorders them so that the hand to the left is always labeled as the left hand. This behaviour is expected, and since the recording of the gesture and the performance of the gesture are consistent, accurate tracking is achieved.

7. SCORE SYNCHRONIZATION MODES

The pitch-trackers and gesture-trackers send synchronization messages to NoteAbilityPro. These messages adjust NoteAbilityPro’s playback tempo in order to align it to the score position requested by the trackers. Depending on the kind of events embedded in the NoteAbilityPro score, the mode of alignment between the score and the tracker can be very tight or relatively loose. Close alignment causes the tempo of the score to be dramatically altered so that events in the score are triggered with very little latency. With a looser following mode, the synchronization between the score and performance is not as tight, but the instrument and rhythmic events embedded in the score are played more musically. The follow mode of NoteAbilityPro can be altered during runtime and requires the use of different modules to support the requirements of different passages in the composition. As well, it should be noted that synchronization messages from pitch-trackers are sent only when new note events are detected, while synchronization messages from gesture-trackers provide continuous score position data.

8. MEDIATION TECHNIQUES

Since IMuSE allows multiple modes of score-following with a single performance, one of the challenges is mediating between the data produced by pitch-trackers and gesture-trackers that are running in parallel. For example, there might be two monophonic instruments being pitched by separate unescofo– modules while piano hand gestures and violin bow motion are tracked by separate gesture-followers. Since the score only want to receive one score position from the analysis layer of IMuSE, we must mediate between the score positions proposed by the different trackers.

As a basic control strategy, one of the trackers is designated as the leader and the others as followers. The leader has the ability to update the followers based on its calculation of the current score position. The tracker designated as the leader can be changed during performance either by sending a message from the score or by analyzing the reliability of the data produced by each tracker. Comparing the deviation between the trackers’ reported score positions and the time we expect those score positions to occur allows us to assess the reliability of each tracking module. The tracker with the smoothest overall trajectory is considered to be the most
6. VIDEO TRACKING OF PIANO HANDS

In some cases, it is advantageous to perform both pitch-tracking and gesture-tracking on the same instrument and to use both streams of data to improve the score-following accuracy. The piano, as a polyphonic instrument, is a good example of an instrument for which both modes of tracking are useful. Pitch-tracking is sometimes unreliable, especially in dense passages or when the sustain pedal is down. In such situations, we have found that tracking the movements of the performer’s hands (in conjunction with pitch-tracking) provides us with much more reliable score-following. In order to track the movement of the pianist’s hands, a camera is mounted above the piano in such a way that the entire keyboard can be captured. While any webcam can be used, our tests were carried out using a Logitech C920 webcam [11]. As with other instances of gesture-tracking, the hand movements of a rehearsal are recorded and aligned to the score. The performance is then tracked using approximately the same camera placement and the performance gestures are compared to the recorded model using our gesture-tracking patch.

A Jitter module was designed to analyze the video image from the camera and to produce two channels of continuous data that follow the horizontal movements of the two hands. We are not concerned with individual finger movement, just the relative positions of the hands as they move up and down the keyboard. As in the case of viola bow tracking, wrong notes are tolerated provided that the hands move close to their correct positions at more or less the correct time.

The amount of data produced by a camera per frame is formidable (640x480x3 = 921,600 pixels per frame). As an added benefit, the video is cropped so that only the portion of the frame that contains the piano keyboard is recorded and aligned to the score. The performance is then tracked using approximately the same camera placement and the performance gestures are compared to the recorded model using our gesture-tracking patch.

The video is cropped to isolate the piano keyboard.

In order to isolate the hand movements, a background subtraction is performed to eliminate the keyboard from the video image. Several frames of the keyboard alone without the performer present are recorded, and this recording is averaged to remove any minute lighting differences. The resulting image is subtracted from each incoming frame, which results in an image containing only the differences between the current video frame and the subtracted image, i.e. the hands only. An optional last step is to create a binary image by setting and applying a luminance threshold. All pixels above the threshold are set to one pixels and all those below the threshold to off-pixels. By fine-tuning the threshold, further minute fluctuations in the image can be removed.

A “mean-shift” algorithm performs the bulk of the actual tracking. This algorithm is implemented in the Open CV library [3] and was ported to MaxMSP/Jitter by Jean-Marc Pelletier [14]. Before the mean-shift algorithm can track the performance, a blob tracking routine is run to determine if there is any major movement within the frame. The minimum blob-size is set to a high value so that any variations in lighting that were not eliminated by the previous filtering methods or other unwanted movements (e.g. shadows cast by the performer while moving the upper body, keyboard shifts during the use of una corda pedal, etc.) are not detected as hands. Once a blob of a certain size is identified, its bounding rectangle is set to the dimensions of the blob algorithm. The object that performs the mean-shift calculations has two distinct operational modes: 1) mean-shift tracking and 2) continuously adaptive mean-shift tracking (cam-shift).

If the center of one window falls within the bounds of another window, the analysis windows have collided and the computer sees that the analysis windows have collided and reorders them so that the hand to the left is always labeled as the left hand. This behavior is expected, and since the recording of the gesture and the performance of the gesture are consistent, accurate tracking is achieved.

7. SCORE SYNCHRONIZATION MODES

The pitch-trackers and gesture-trackers send synchronization messages to NoteAbilityPro. These messages adjust NoteAbilityPro’s playback tempo in order to align it to the score position requested by the trackers. Depending on the kind of events embedded in the NoteAbilityPro score, the mode of alignment between the score and the tracker can be very tight or relatively loose. Close alignment causes the tempo of the score to be dramatically altered so that events in the score are triggered with very little latency. With a looser following mode, the synchronization between the score and performance is not as tight, but the temporal and rhythmic events embedded in the score are played more musically. The follow mode of NoteAbilityPro can be altered during the performance to accommodate the requirements of different passages in the composition. As well, it should be noted that synchronization messages from pitch-trackers are sent only when new note events are detected, while synchronization messages from gesture-trackers provide continuous score position data.

8. MEDIATION TECHNIQUES

Since IMUSE allows multiple modes of score-following with a single performance, one of the challenges is mediating between the data produced by pitch-trackers and gesture-trackers that are running in parallel. For example, there might be two monophonic instruments being pitch-tracked by separate antescofo~ modules while piano hand gestures and violin bow motion are tracked by separate gesture-followers. Since the score only want to receive one score position from the analysis layer of IMUSE, we must mediate between the score positions proposed by the different trackers.

As a basic control strategy, one of the trackers is designated as the leader and the others as followers. The leader has the ability to override the followers based on its calculation of the current score position. The tracker designated as the leader can be changed during a performance either by sending a message from the score or by analyzing the reliability of the data produced by each tracker. Comparing the deviation between the trackers’ reported score positions and the time we expect those score positions to occur allows us to assess the success or failure of the analysis and whether the current analysis data is to be used in the output:

1) Size of the tracking windows
2) Location of the tracking windows
3) Number of tracking windows/blobs

If the tracking windows fall below or beyond an empirically determined size limitation, the tracking is considered void and the cam-shift algorithm is engaged to determine a new and better window size and location.

If two stable windows have been found, the locations of both windows are compared to each other. If the center of one window falls within the bounds of the other, it is assumed that both hands are so close to each other that they can no longer be distinguished.

The number of blobs is constantly monitored during each analysis step. If no blobs are present, a decision is made that the analysis has failed (no hands on the piano) and the last confirmed value is used for data output. If only one blob/hand is detected both analysis windows are set to the same location. This is necessary due to the limitations of the gesture follower, which assumes that there is constant input from two separate, continuous sources. If two blobs are present, the locations are sent to two separate units.

Blobs, as well as mean-shift window locations, are analyzed to obtain information about right and left hand ordering. The center points of both locations are calculated and compared to each other. If the first point is beyond the location of the second, the two are switched and relabeled. This guarantees that the system is aware of which window and which blob is considered right and which is considered left.

There is one special case where the system “breaks” in terms of human observable right/left ordering versus computed right/left ordering. If the pianist uses hand crossings, the results observed by a human and the machine are not consistent. For example, a human observer will easily recognize the gesture and identify that the left hand crossed over the right. In contrast, the computer sees that the analysis windows have collided and reorders them so that the hand to the left is always labeled as the left hand. This behavior is expected, and since the recording of the gesture and the performance of the gesture are consistent, accurate tracking is achieved.
reliable module and can be established as the new leader.

Generally-speaking, pitch-tracking of monophonic instruments using antescofo- provides us with the most reliable score-following data, but the instruments will not be playing at all times, there may be instances when acoustic interference from other instruments become problematic, a section of the composition may be improvisatory, or there may be occasions where mediation between the trackers be as flexible and dependable as possible. As a failsafe measure, we can send manual cues to all trackers to reset their internal data. We are currently investigating new tracking environments. In such cases, the computer operator needs to have the freedom and flexibility to temporarily disable the tracking while other steps (jumping ahead to the next major section; slowing down or speeding up current score speed; etc.) are taken so that the score becomes realigned to the performance. Despite the advantages of automated score-following, our experience has shown us that creating a completely automated and unmonitored score-following environment is currently unrealistic for live performance situations. Even when multiple modes of score-following are used, tracking errors occasionally occur and user-interaction is required in order to maintain the integrity of the performance.

9. EVALUATION

Our evaluation of IMuSe is primarily based on rehearsals and live performances of interactive computer music compositions by all three authors. Our experience using IMuSe is that adding a second mode of tracking increases the reliability of score-following. For example, when we added gesture-tracking to piano performances the score-following accuracy was significantly improved. Gesture-tracking performers in anensemble situation also has the advantage of avoiding the adverse latency, we set NoteAbilityPro to run a fraction of a beat ahead of the actual synchronization positions that are sent to it. As well, controls can be encoded as antescofo- action messages if sound events need to be more precisely linked to performance events.

We plan to continue to refine IMuSe and to test it in a wide variety of performance situations. In particular, we want to gesture-follow cells performances (tracking both the bow hand and left hand), the motion of the trombone slide and the movements of multiple dancers. We want to further refine our piano hand-tracking methods and to compare its accuracy to pitch-tracking in complex contemporary compositions. More work needs to be done in developing robust mediation mechanisms between the layers of incoming data, and in developing self-correction strategies for updating or modifying data when followings start to become inaccurate or unreliable.

More information and video examples of IMuSe tests and performance can be viewed on our website: http://www.opusonemusic.net/muset/imuse.html.

11. ACKNOWLEDGEMENTS

IMuSe is funded by the Canadian Social Sciences and Humanities Research Council (SSHRCC.) Thanks to IMuSe researcher Yota Kobayashi and to performers Sarah Kwok and Christopher Morano who assisted in the development and testing of the system. We are grateful for our on-going collaboration with IRCAM and in particular, to the assistance of Arshia Cont, Frédéric Bevilacqua and Bruno Zamborlin.

12. REFERENCES


reliable module and can be established as the new leader.

<table>
<thead>
<tr>
<th>Pitch-tracker 1</th>
<th>Pitch-tracker 2</th>
<th>Gesture-tracker 1</th>
<th>Gesture-tracker 2</th>
<th>Gesture-tracker 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antescofo</td>
<td>Antescofo</td>
<td>GF / IMuSE</td>
<td>GF / IMuSE</td>
<td>GF / IMuSE</td>
</tr>
<tr>
<td>Instr. 1 Audio</td>
<td>Instr. 2 Audio</td>
<td>Instr. 2 Gesture</td>
<td>Instr. 3 Gesture</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 9. Negotiating between incoming score-following streams.

Generally-speaking, pitch-tracking of monophonic instruments using antescofo—provides us with the most reliable score-following data, but the instruments will not be playing at all times, there may be instances where acoustic interference from other instruments become problematic, a section of the composition may be improvisatory, or there may be occasions where movement (of a dancer, for example) might be a primary controlling feature of the work. It is important, therefore, that mediation between the trackers be as flexible and dependable as possible. As a fail-safe measure, we can send manual cues to all trackers to reset their internal model usually has to be created, analyzed, and aligned to the score for each new performer to account for differences in playing style and performance mannerisms.

Our experiences using IMuSE showed us that mechanisms for ignoring score-following decisions are extremely important in any score tracking environment. When the tracking routines get lost, human intervention is needed to re-align the score to the performance. In most cases, tracking failures, the follower will get stuck waiting for a specific event and no longer moves the score forward in musical time. In such cases, the composer/performer needs to have the freedom and flexibility to temporarily disable the tracking while other steps (jumping ahead to the next major section; slowing down or speeding up current score speed; etc.) are taken so that the score becomes realigned to the performance. Despite the advantages of automated score-following, our experience has shown us that creating a completely automated and unmonitored score-following environment is currently unrealistic for live performance situations. Even when multiple modes of score-following are used, tracking errors occasionally occur and user-interaction is required in order to maintain the integrity of the performance.

9. EVALUATION

Our evaluation of IMuSE is primarily based on rehearsals and live performances of interactive computer music compositions by all three authors. Our experience using IMuSE is that adding a second mode of tracking increases the reliability of score-following. For example, when we added gesture-tracking to piano performances the score-following accuracy was significantly improved. Gesture-tracking performers in an ensemble situation also has the advantage of avoiding the crosstalk of sounds that can occur when pitch-tracking is used. While close-miking techniques can be employed to reduce this problem, gesture-tracking is not susceptible to the sounds or movements of other players. One limitation of gesture-tracking is that it is only viable in situations where the performer’s tracking is close to the recorded model. This means that a new gesture model usually has to be created, analyzed, and aligned to the score for each new performer to account for

10. CONCLUSION AND FUTURE WORK

There are several advantages to creating an interactive computer music composition using IMuSE. The fact that most of the external controls and score-following data are embedded in a single score makes editing easier. Modifications to the score, such as inserting or deleting passages, will alter all aspects of the performance. The embedded messages will simply be sent at a new time and score-following data will be modified to reflect the changes to the score. Having control messages integrated in the score also makes it much easier to rehearse compositions. To start mid-way through a composition, the score sends a playStart message to MaxMSP, in turn, passes the starting beat position to all the followers so they can begin tracking at the correct location. Other embedded controls are sent as the score moves past them during playback.

Archiving a composition is also simplified using IMuSE since almost all of the performance and score-following data, including all the BPFs containing the gestures, are contained in a single document. It is even possible to reference the MaxMSP and Pd files associated with the score and to launch them from NoteAbilityPro.

Latency is always an issue in systems involving inter-application communication. For both pitch-tracking and gesture-tracking, detection requires that the event has already happened, so our analysis of events is always running slightly behind the performance. To overcome this latency, we use NoteAbilityPro to run a fraction of a beat ahead of the actual synchronization positions that are sent to us. As well, controls can be encoded as antescofo—action messages if sound events need to be more precisely linked to performance events.

We plan to continue to refine IMuSE and to test it in a wide variety of performance situations. In particular, we want to gesture-follow cello performances (tracking both the bow hand and left hand), the motion of the trombone slide and the movements of multiple dancers.

We want to further refine our piano hand-tracking methods and to compare its accuracy with existing methods in complex contemporary compositions. More work needs to be done in developing robust mediation mechanisms between the layers of incoming data, and in developing self-correction strategies for updating or modifying score data that followers start to become inaccurate or unreliable. More information and video examples of IMuSE tests and performances can be viewed on our website: http://www.opusonemusic.net/muset/imuse.html.

11. ACKNOWLEDGEMENTS

IMuSE is funded by the Canadian Social Sciences and Humanities Research Council (SSHRC.) Thanks to IMuSE researcher Yota Kobayashi and to performers Sarah Kwok and Christopher Moran who assisted in the development and testing of the system. We are grateful for our on-going collaboration with IRCAM and in particular, to the assistance of Arshia Cont, Frédéric Bevilacqua and Bruno Zamborlin.

12. REFERENCES


1. INTRODUCTION

With the growth of computer networks and their specific uses in networked audio and music, the desire for sharing real-time audio has inspired research by both musicians and network engineers.

To measure the performance of real-time network streaming systems, some parameters, such as bandwidth, latency, jitter and packet loss, can be used.

Latency corresponds to how long a message takes to travel from one end of a network path to the other [12], and in the case of audio streaming, latency is perceived as delay. Jitter is defined as variation in latency over time and it is calculated as the standard deviation of latency measurement. Packet loss is the percentage of packets lost in transmission, received corrupted or received out of order and therefore rejected.

A point raised frequently in the computer networking literature [5], [19]) is that multimedia streaming requires high bandwidth and low latency. It is also a common view in the same literature that data loss is not a crucial impairment to audio streaming, because lost packets might only result in small glitches in the resulting audio/video streams, without compromising intelligibility.

Different audio application scenarios may not agree with the aforementioned views, and according to specific requirements they may allow the trade-off between latency and packet loss to tilt to one side or the other. For instance, in a distributed music rehearsal one would not worry so much about packet loss but would with latency, whereas in a distributed musical recording there should be no packet loss, but high latency values might be overlooked [14].

From a low-level networking point of view, latency and packet loss are features directly related to the protocol implementation and to the concept of protocol reliability. Thus, a reliable network protocol uses techniques to achieve reliability that would normally increase the delivery time of a network packet, resulting in increased latency. Since an unreliable protocol like UDP is usually faster than a reliable protocol, UDP is suggested by many computer networking textbooks as the best protocol choice for audio streaming. In fact we do see the use of the UDP protocol in many existing real-time audio streaming applications, such as NetJack [3], SoundJack [4] and JackTrip [2].

As we examine different usage scenarios, like rehearsal and recording, we are invited to review the way audio is distributed over computer networks and to consider other transport protocols besides UDP [13]. TCP [10], for instance, is a reliable protocol and its use should have less packet loss rates as compared to UDP. Other protocols that will be investigated in this paper are SCTP [9], which is reliable and has been used in VoIP, and DCCP [6], which is an unreliable protocol with congestion control.

In this paper we have considered the specific case of audio streaming within local computer networks, as a first testbed for future discussion of audio streaming over wide-area networks. The remainder of this paper is organized as follows: Section 2 presents the network architecture and the four transport layer protocols we studied, Section 3 presents the implementation of a tool for multi-channel audio streaming using these protocols, and Section 4 presents the experimental setup and numerical results. Conclusions and future steps are presented in Section 5.

2. BACKGROUND

This section presents some features about the TCP/IP protocol stack, the IP protocol and the four transport proto-