INTEGRATED MULTIMODAL SCORE-FOLLOWING ENVIRONMENT

Martin Ritter
School of Music
University of British Columbia
Vancouver, Canada

Keith Hamel
School of Music
University of British Columbia
Vancouver, Canada

Bob Pritchard
School of Music
University of British Columbia
Vancouver, Canada

ABSTRACT

The Integrated Multimodal Score-following Environment (IMuSE) [5] is a software project aimed at developing a system for the creation, rehearsal and performance of score-based interactive computer music compositions. An enhanced version of the NoteAbilityPro [7] music notation software is the central controller in IMuSE. The score contains conventional notes and performance indications as well as discrete and continuous control messages that can be sent to other applications such as MaxMSP [15] or Pure data (Pd) [11] during performance. As well, multiple modes of score-following [2] can be used to synchronize the live performance to the score. Score-following strategies include pitch-tracking of monophonic instruments, pitch-tracking and amplitude-tracking of polyphonic instruments and gesture-tracking of performers' hand movements.

In this paper, we present an overview of the IMuSE system, with a focus on its abilities to monitor and coordinate multiple pitch and gesture trackers. As an example of gesture-tracking, we show how video analysis is used to follow piano hand movements. We discuss strategies for negotiating between multiple streams of score-following data and we assess the strengths, limitations and future research directions of the IMuSE system.

1. BACKGROUND

Score-following has a relatively long history, with the first research being done in the early 1980s by Dannenberg [5, 6] and Vercoe [24, 25]. This research was aimed at automated accompaniment where the computer would generate midi or audio tracks that were synchronized to a live musical performance. In the following decade, Puckette [17, 18] developed the *explode* Max object that could be used for monophonic score-following of pitch-tracked or midi data. In the early 2000s, score-following systems such as suivi~ [20, 21] and antescofo~ [4] were developed using Hidden Markov Models [1]; these systems prove to be far more robust in performance situations where machine perception and performer mistakes need to be accommodated. More recently, audio analysis has been used for music retrieval systems such as Query-by-Humming [12] and intelligent audio editors [7]. Our focus is on score-following in live interactive computer music compositions involving several performers and where complex audio interactions between the live instruments and the computer generated sounds are required.

2. INTRODUCTION

One of the challenges of creating score-based interactive computer music compositions is that there is often no connection between the notated score and the programming environment used to produce the electroacoustic layers of the composition or to process the live instruments. As a result, composers must develop strategies for synchronizing the live and electroacoustic components of their composition using manual cues, foot pedals or rigid timing controls. As well, the fact that several disconnected software applications are being used in an interactive performance can make rehearsing difficult and the editing of the composition can be awkward since it often means altering several different software components.

The Integrated Multimodal Score-following Environment (IMuSE) has been developed to alleviate some of the issues associated with creating, rehearsing and performing score-based interactive computer music compositions. While this system could be used in some improvisatory performances, it is primarily designed for compositions that have a well-defined score and a where a high degree of synchronization between the live performance and the electroacoustic processing is needed.

3. SYSTEM OVERVIEW

The central software component in IMuSE is the notation program NoteAbilityPro. This application has been enhanced to allow MaxMSP messages, multichannel breakpoint functions (BPFs), extended notes, conTimbre [9] controls and antescofo~ action messages to be added to the score and aligned to other score events. As well, NoteAbilityPro can be externally controlled through network UDP messages so that synchronization information from pitch-tracking and gesture-tracking of a live performance can be used to alter the playback of the score. In order to accommodate conventional playback methods, remote messaging and synchronization with the live performance, music staves in a NoteAbilityPro score can have different functions: they can contain music notation that is intended for performers to play, notes and sound files that are intended to be played as audio events directly from the score, messages that are intended to control external applications, or notation and continuous graphical data

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 are 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.

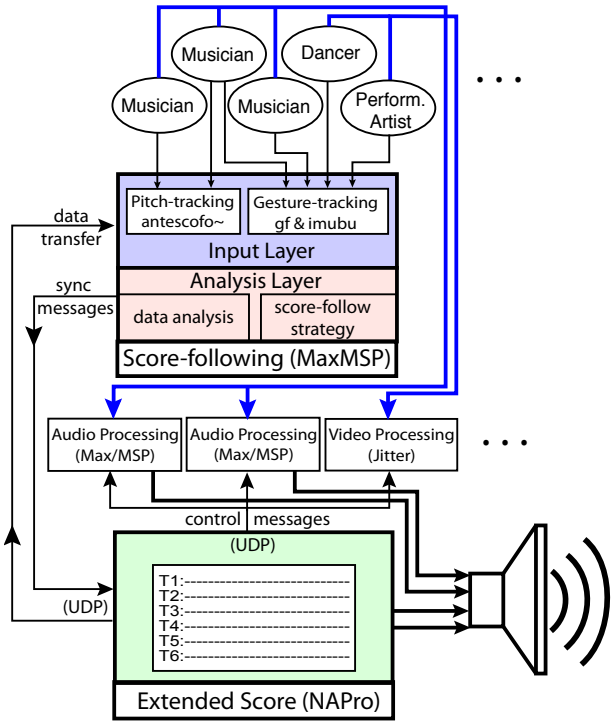

Figure 1. IMuSE system overview

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 messages can be parsed through an *OSC-route* [23] object or can be send directly to *receive* objects anywhere in the patch.

Continuous messages are represented in the score as single- or multi-channel BPFs. BPFs can be created in NoteAbilityPro using 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 masterVolL*], [*r harm*] (the first channel of the BPF) and [*r harm_1*] (the second channel of the BPF).

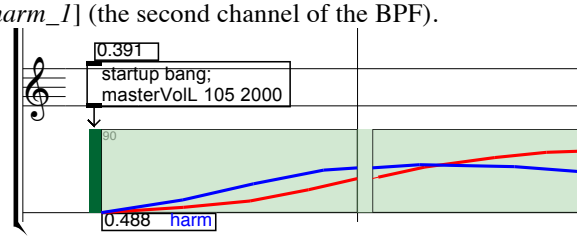

Figure 2. Discrete and continuous control messages in NoteAbilityPro

5. SCORE-FOLLOWING

5.1. Pitch-tracking

In order to facilitate pitch-tracking of monophonic or polyphphonic 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.

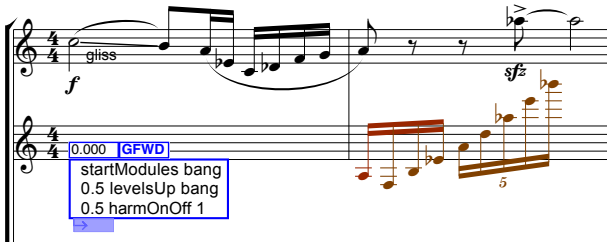

Figure 3. Staves used by *antescofo~* for pitch-tracking and action messages.

The *antescofo~* file generated from this score contains both the note information needed to track the performance (the notes on the first staff) and the action messages embedded in the second staff. These action messages are sent directly to *receives* in MaxMSP or Pd. Pitch content in this file is specified in midicents so that 7200 is equivalent to midi pitch 72. In Table 1, action

messages are shown in bold and the pitches to be tracked by *antescofo~* in italics.

```
BPM 90.0
; ---------- measure 1 beat 0.00 ---------
GFWD event0
{
  startModules bang
  0.5 levelsUp bang
  0.5 harmOnOff 1
}
MULTI ( 7200->7100 ) 2.0000 Measure1
NOTE 7100 0.5000
NOTE 6900 0.2500
NOTE 6300 0.2500
NOTE 6000 0.2500
NOTE 6100 0.2500
NOTE 6500 0.2500
NOTE 6700 0.2500
; ---------- measure 2 beat 4.00 ---------
NOTE 6900 0.5000 Measure2
GFWD gEvent84
{
  0.000 pafNote 57 80 3000
  0.250 pafNote 53 70 3000
  0.250 pafNote 59 60 3000
  0.250 pafNote 63 50 3000
  0.250 pafNote 69 40 3000
  0.200 pafNote 74 35 3000
  0.200 pafNote 80 30 3000
  0.200 pafNote 88 28 3000
  0.200 pafNote 94 25 3000
}
NOTE 0 1.0000
NOTE 8000 2.5000
```

Table 1. *Antescofo~* data generated from the score.

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 *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 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 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 *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.

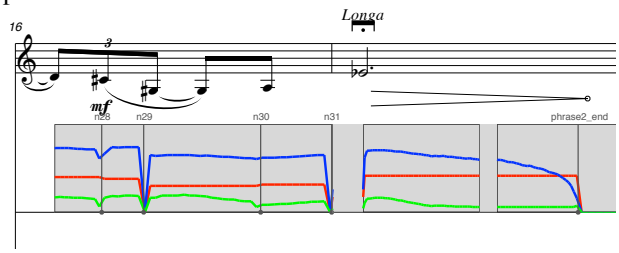

Figure 4. BPF Staff containing performance data.

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.

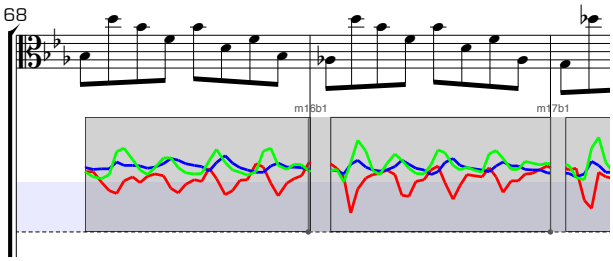

Figure 5. BPF Staff containing viola bow motion.

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 PlayStation® Eye [15] camera and 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.

Figure 6. The location of the video camera above the piano keyboard

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 colour information from the video, since colour has no impact on the tracking algorithm. The conversion from RGB to luminescence reduces the data to 640x90x1 (57,600 pixels per frame) – a reduction of 93.75% from the original data size.

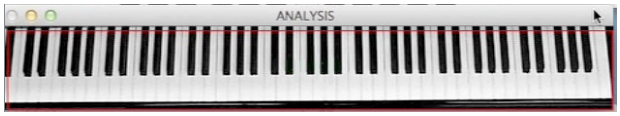

Figure 7. The video is cropped to isolate the piano keyboard.

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 luminence threshold. All pixels above the threshold are set to on-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 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).

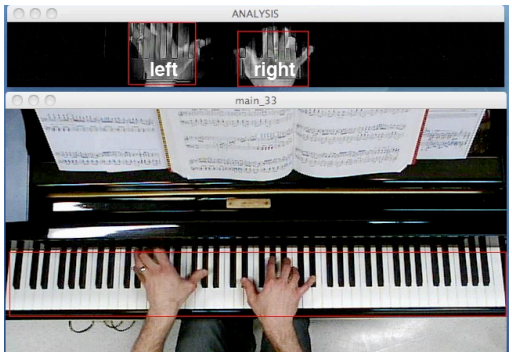

Figure 8. Tracking the movements of the two hands in a piano performance

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 point. The center mass 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 found and convergence has been reached. At this 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 itself in size to accommodate pixel spread (i.e. an object moving closer to or away from the camera).

Initially, the bounding box of the blob-tracking stage of the analysis is used to set the window function of the cam-shift algorithm. The algorithm is run for approximately two frames worth of data. During this analysis the bounding rectangle is set to the dimensions that the cam-shift algorithm perceives as being the appropriate size for the hands. Once a suitable size is established, the cam-shift tracking is turned off, and mean-shift tracking is engaged.

6.2. Error Handling

There are three error handling routines that determine 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 re-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 the two cam-shift 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 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 timed and rhythmic events embedded in the score are played more musically. The follow mode of NoteAbilityPro can be altered during the performance based on 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 update 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 reliability of each tracking module. The tracker with the smoothest overall trajectory is considered to be the most

reliable module and can be established as the new *leader*.

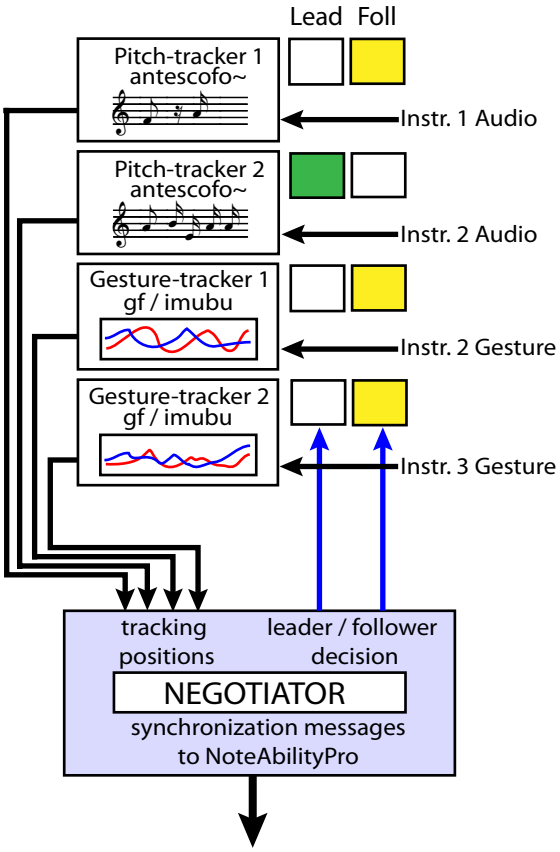

Figure 9. Negotiating between incoming score-following streams.

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 when 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 failsafe measure, we can send manual cues to all trackers to reset their internal score positions. We are currently investigating new methods for moving smoothly between multiple trackers as they come in and out of prominence in complex chamber music and multimedia performance situations.

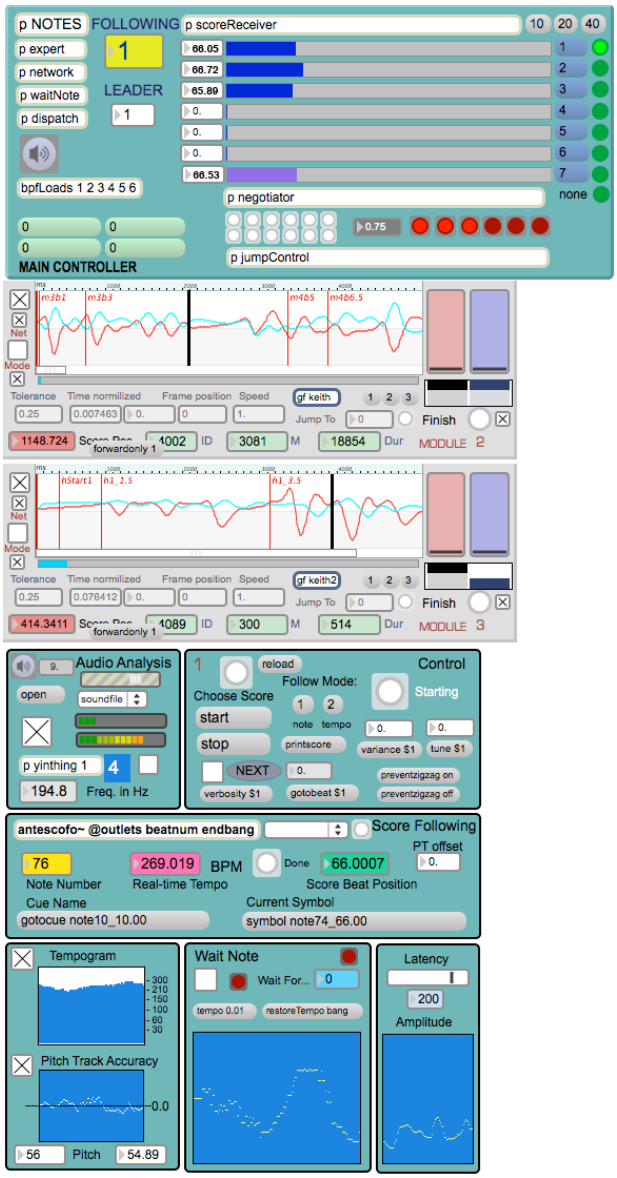

Figure 10. The IMuSE score-follower with two gesture-trackers and one pitch-tracker active.

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 cross bleeding 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

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 of 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 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.

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 that, 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 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 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 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 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 Kobayahsi 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

[1] Baum, L. and Petrie, T. “Statistical inference for probabilistic functions of finite state markov chains.” The Annals of Mathematical Statistics, 37(6):1554–1563. 1966.

[2] Bevilacqua, F., Schnell, N., Rasamimanana, N., Zamborlin, B., Guédy, F., “Online Gesture Analysis and Control of Audio Processing, Musical Robots and Interactive Multimodal Systems.” Jorge Solis and Kia C. Ng. *Musical Robots and Interactive Multimodal Systems*. Springer Tracts in Advanced Robotics Vol 74. Springer Verlag. 2011, p. 127-142.

[3] Bradski, G, and Kaehler, A. *Learning OpenCV*. O'Reilly, 2008.

[4] Cont, A. "ANTESCOFO: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music", *Proceedings of International Computer Music Conference*, Belfast, Ireland. August, 2008.

[5] Dannenberg, R, & Mukaino, H. “New Techniques for enhanced quality of computer accompaniment.” *Proc of ICMC*. 1998. pp. 243-249.

[6] Dannenberg, R. An On-Line Algorithm for Real-Time Accompaniment. Proc of ICMC 1984. pp 193-198.

[7] Dannenberg, R. & Hu, N. “Polyphonic audio matching for score-following and intelligent audio editors.” In Proc. of ICMC. 2003. pp. 27-34.

[8] Hamel, K. “Integrated Multimedia Score-following Environment.”

<http://opusonemusic.net/muset/imuse.html.> February 15, 2012.

[9] Hummel, T. “ConTimbre – Die Instrumentaltechnik der Neuen Musik.” <http://www.contimbre.com> February 15, 2012.

[10] Litke, D & Hamel, K., “A Score-based Interface for Interactive Computer Music,” *Proceedings of the International Computer Music Conference*. Copenhagen, Denmark. August 2007.

[11] Logitech C920 Website. “Logitech HD Pro Webcam C920. http://www.logitech.com/en-ca/product/hd-pro-webcam-c920. Feb 22, 2013

[12] McNab, R., Smith, L., Witten, I., Henderson, C. and Cunningham, S. “Towards the digital music library: tune retrieval from acoustic input,” in *DL ’96: Proceedings of the first ACM international conference on Digital libraries*. New York, NY, USA: ACM, 1996, pp. 11–18.

[13] Orio, N., Lemouton, S., Schwarz, D. & Schnell, N., “Score-following: State of the Art and New Developments”, *Proceedings of New Interfaces for Musical Expression*. Montreal, Canada. 2003.

[14] Pelletier, J. “cv.jit – “Computer Vision for Jitter.” <http://jmpelletier.com/cvjit/> February 15, 2013.

[15] PlayStation Eye Website. “PlayStation Eye Camera for the PS3 System.” <http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html> February 19, 2013.

[16] Puckette, M. "Pure Data." *Proceedings of International Computer Music Conference*. Hong Kong, China. August, 1996.

[17] Puckette, M. & Lippe, C. Score Following in Practice,” *Proc. of ICMC*. 1992. Pp. 182-185.

[18] Puckette, M. “Score following using the sung voice.” *Proc. of ICMC*, 1995. pp. 175–178.

[19] Schnell, N., Röbel, A., Schwarz, D., Peeters, G., Borghesi, R., “MuBu & Friends: Assembling Tools for Content Based Real-Time Interactive Audio Processing in Max/MSP.” *Proceedings of International Computer Music Conference*. Montreal, Canada. August, 2009.

[20] Schwartz, D. Orio, N, & Schnell, N. “Robust Polyphonic Midi Score Following with Hidden Markov Models.” Proc. of ICMC 2004.

[21] Schwartz, D. Cont, A., & Schnell, N. From Boulez to Ballads: Training Ircam’s Score Follower”. Proc. of ICMC 2005.

[22] Texas Instruments Website. Chronos:Wireless development tool in a watch-TI.com.” <http://www.ti.com/tool/ez430-chronos> February 19, 2012.

[23] Wright, A., Freed, A. & Momeni, A. “OpenSound Control: State of the Art 2003,” *Proceedings of New Interfaces for Musical Expression*. Montreal, Canada. 2003

[24] Vercoe, B. “The synthetic performer in the context of live performance.” *Proc. of ICMC*. 1984. pp. 199-200

[25] Vercoe, B. L. and Puckette, M. S. Synthetic Rehearsal: Training the Synthetic Performer. *Proceedings of the International Computer Music Conference*. 1985.

[26] Zicarelli, D. Max/MSP Software. San Francisco: Cycling ‘74. 1997.