Alan Tishk

Culminating Experience

Music Technology Innovation - Class of 2014

Virtual Conducting Experience -

One Half of ROBATON

Introduction

Robaton is a combination of two separate projects - the Virtual Conducting Experience (referred to in this paper as “VCE”), developed by myself; and Curiosibot, a robot developed by Pierluigi Barberis. We started these projects separately, and fused them into Robaton in February, 2014.

This report will focus on the VCE, and will explain its role as half of Robaton.

The VCE is a combination of hardware and software allowing the user to experience conducting a virtual orchestra through hand gestures.

Description of the Culminating Experience Project (the WORK)

The initial plan for the VCE was to build a training program for students learning conducting - the system would utilize the user’s gestures to control playback of a virtual ensemble, analyze the user’s conducting gestures, and give real-time feedback about his/her performance.

Several ideas were brainstormed. Some were scrapped, and some ultimately made it into the final project. Some of the scrapped ideas include:

- A portable version of the VCE, developed in Pure Data, running on a Raspberry Pi computer, utilizing accelerometers and speakers embedded in a piece of clothing, and Google Glass for displaying data
- Multiple pieces of music with varying difficulty levels, allowing the user to “level up” in the software
- Alternate gesture tracking technology, such as the Leap Motion, Hot Hands, or a DIY version of Imogen Heap’s Mi.Mu Gloves

The Kinect sensor was chosen due to its ability to track large hand gestures (as opposed to the Leap Motion, which has a considerably smaller range), its strong adoption by the Maker and hacker communities, and its price point ($40 USD on the used market, with several used Kinect Model 1414 units available at video game stores around the USA).

The VCE ultimately evolved into a project utilizing the Kinect sensor and these pieces of software: Synapse for Kinect, Ableton Live 9 Suite, Max 6, Max for Live, and Kontakt. In February, 2014, it was combined with Pierluigi Barberis’s “Curiosibot” project to form our joint CE project called Robaton, and the conducting “trainer” idea was put on the back burner.

Below is a description of how the system works:

A piece of music is loaded into Ableton Live. This music can consist of audio, warped to follow the conductor's timing using Ableton's audio warping algorithms; MIDI-sequenced parts being played back through synthesizers or samplers; or a hybrid of both audio and MIDI data.

An open-source[1] application called Synapse runs in the background and collects information from the Kinect, forwarding this information, as OSC data, to Max via UDP. Synapse tracks the position of the user's hands.

To the right is a screenshot from the Synapse software, showing the software bound to the user's skeleton.

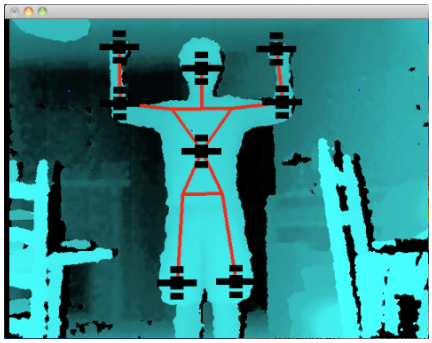

Fig. 1 - Synapse for Kinect, showing the software bound to the user's skeleton

A custom Max for Live patcher, programmed by myself, is loaded onto a MIDI track in Live. This patcher can be loaded onto any MIDI track, but for the purpose of keeping things easy to compartmentalize, I created a dedicated track for this patcher and titled the track "tempo change."

Screenshots of the VCE follow below:

Fig. 2 - Presentation View of VCE patcher, shown as Max for Live patcher

Opening this patcher reveals the contents, shown below:

[1] http://synapsekinect.tumblr.com/post/6305362427/source

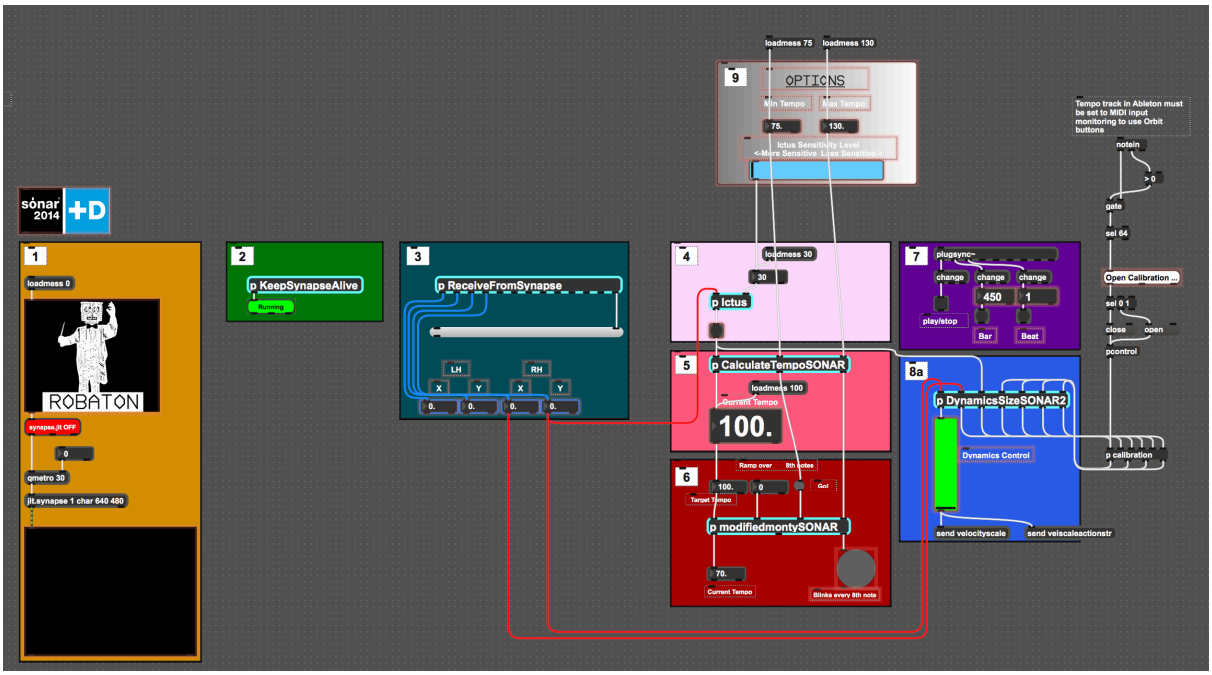

Fig. 3 - Patching View of VCE patcher (top layer)

Color panels have been overlaid on each section of the patcher to highlight the main chunks of the program. As signal flow inside Max is not linear, the numbers at the top left part of each chunk do not exist to demonstrate signal flow, rather they are there to label each chunk for explanation in this paper.

Below are synopses of the function of these chunks.

Chunks 1, 2, 3

These three chunks deal with information from the Synapse software running in the background. Chunk 1 is the Jitter object available as part of Synapse, which, in this project, simply displays a monochrome image from the Kinect's infrared camera. In the future, I may implement functionality to change the color of the image when Synapse locks onto the user's skeleton.

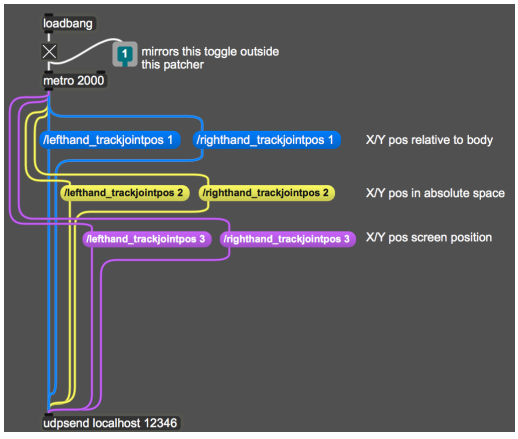

Fig. 4 - Keepalive

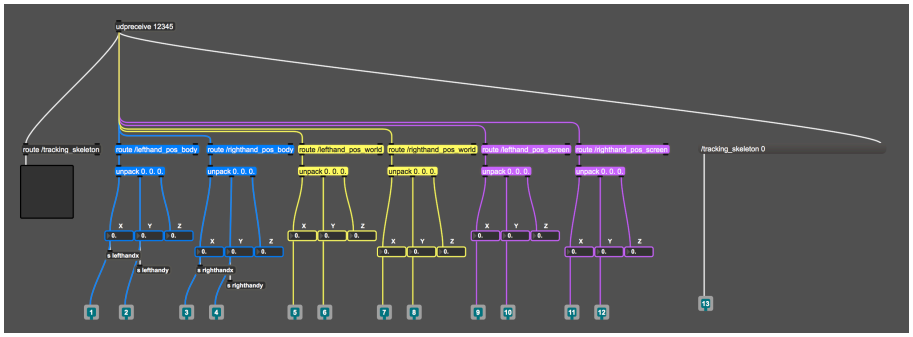

Fig. 5 - Receive from Synapse

Chunk 2 is the “keepalive” for Synapse. Synapse requires us to request the information we need (position of hands, elbows, etc.) every 2 seconds. Chunk 3 receives the requested information from Synapse via UDP.

Figures 4 and 5 show the inside of the “KeepSynapseAlive” subpatcher and the “ReceiveFromSynapse” subpatcher, respectively, which are being used in this particular situation to track the left and right hands.

Chunk 4 - The Ictus

The ictus is the moment at which a beat occurs. Programming a computer to determine the moment of ictus by interpreting a gesture was a point of contention among some classmates and instructors. Everyone agreed on one thing - the ictus is not defined by an exact set of X/Y coordinates. An orchestra does not refuse to play a beat if the conductor’s baton is a few inches off of dead center - rather, the players use several different types of cues, including eye contact, the movement of the conductor’s hands, and even the conductor’s breathing pattern.

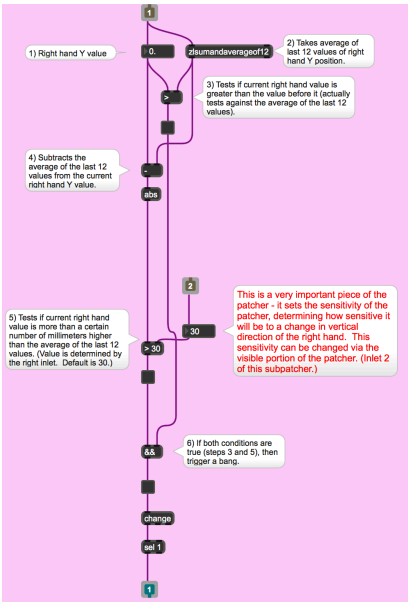

Fig. 6 - Ictus

As the Kinect sensor cannot discern eye contact, and programming Max to interpret breathing patterns would be too challenging for the scope of this project, I decided, with the advice of Ben Houge, to determine the ictus by a change in the vertical movement of the conductor’s hand. In other words, when the conductor’s hand changes from moving downward to moving upward, the software interprets that moment as the ictus and it triggers an event.

Figure 6 shows a screenshot of the “Ictus” subpatcher. This subpatcher determines when the vertical position of the right hand changes by more than 30 mm upward since the last downward movement.

Chunk 5 - Tempo Calculation

This chunk of the VCE can be likened to a tap tempo, but it would be more accurate to compare it to the engine of a car. While driving, you accelerate by pressing on the accelerator. However, when you let your foot off of the gas, the car does not keep going at the same speed - it slows down. The “CalculateTempo”

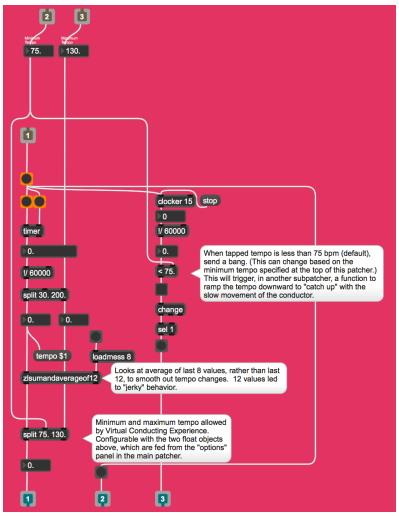

Fig. 7 - Tempo Calculation Subpatcher

subpatcher acts to “rev” the master tempo of Ableton Live - rather than acting as a traditional tap tempo, this patcher will slow the tempo down over time, allowing the system to slow down with the conductor when he/she slows down, and speed up when he/she speeds up. Figure 7 shows the inside of this subpatcher.

Chunk 6 - Tempo Ramping

Simply changing the tempo of Ableton based on each tempo tap produced unnatural results. A method of ramping tempo to the desired tempo was needed. Rather than reinventing the wheel, an internet search for Ableton tempo ramping led to a patcher called “Live Tempo Automator 1.0” by Monty McMont.[2] I heavily modified this patcher to make it usable (and easier to follow) in the VCE, and incorporated it into the project. As the bulk of it is from Monty’s patch, this modified patcher is called “ModifiedMonty.” Figure 8 is a screenshot of this patcher.

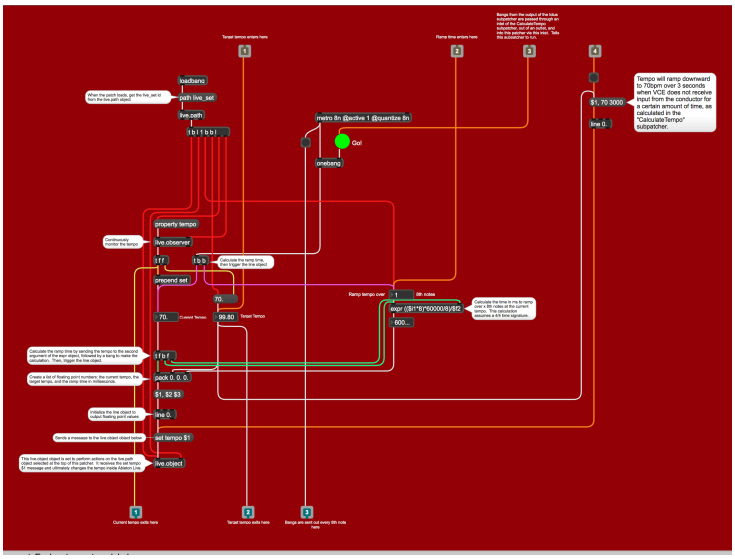

Fig. 8 - Tempo Ramping

Chunk 7 - Bar/Beat Display

This portion of the patch exists to display the bar/beat counter in larger numbers than appear in Ableton’s transport bar.

Chunk 8 - Dynamics Scaling

A huge piece of conducting is controlling the dynamics of the music. Conductors often use the size of their hand gestures to communicate how loudly or softly to play a piece of music. This portion of the VCE exists in two separate pieces, shown in Figures 9 and 10.

The “DynamicsSize” subpatcher measures the vertical size of each gesture in millimeters, and scales the value to values between 0. and 1. This information is sent remotely to the 8b objects inserted onto each MIDI track. This information is then scaled to MIDI CC values to control the velocity of each note on the track. At the time of

---

[2] http://www.maxforlive.com/library/device/386/live-tempo-automator

CE completion, a method of measuring horizontal gestures was in progress, but was not completed in time. This will be implemented in later versions of the VCE.

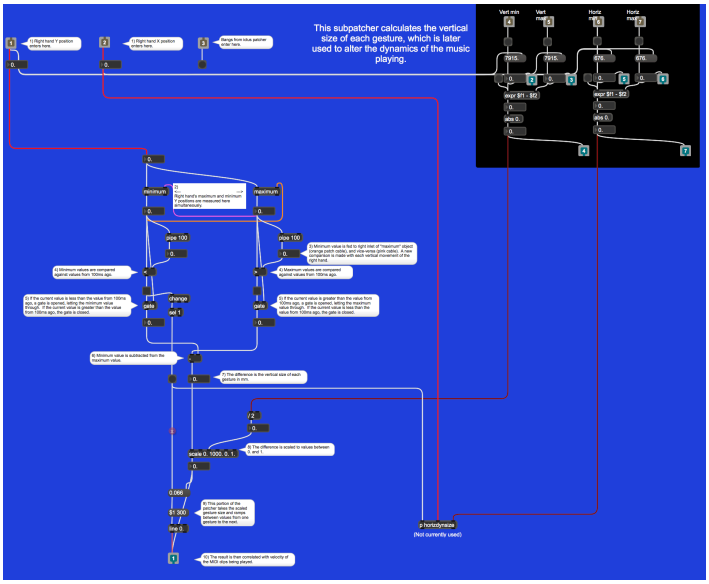

Fig. 9 - Size of gesture correlates with dynamics

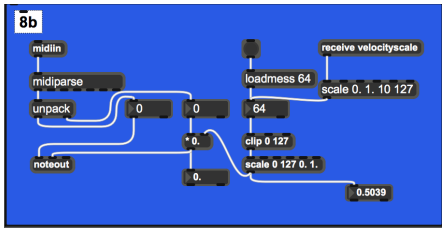

Fig. 10 - This is inserted on each MIDI instrument track

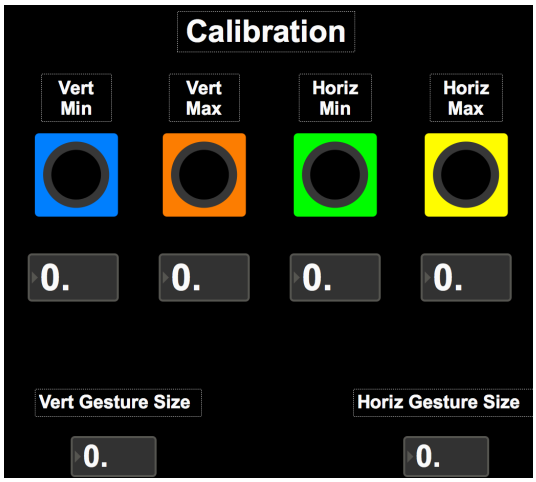

Fig. 11 - Calibration Subpatcher

The “DynamicsSize” subpatcher also receives information from a Calibration subpatcher (Figure 11), which gathers information about the size of the conductor’s right arm by measuring its distance at four distinct points in its travel. The system is calibrated each time a new conductor steps in front of the VCE.

Chunk 9 - Options Panel

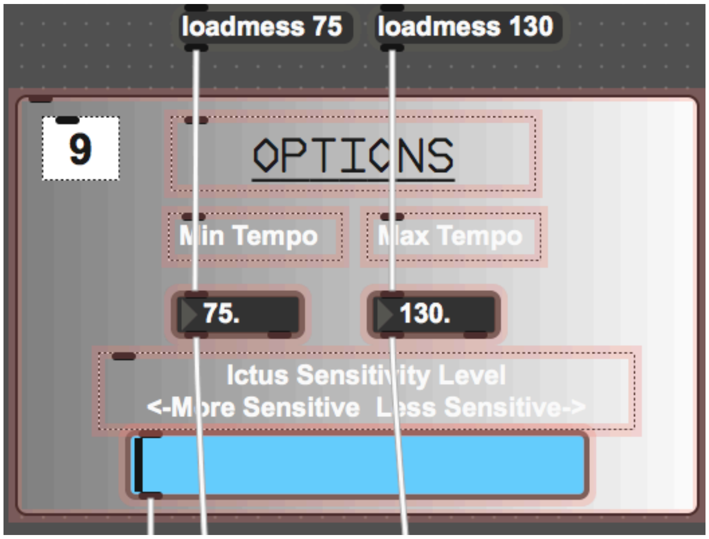

Fig. 12 - Options Panel

The options panel allows the user to set preferences for minimum and maximum tempo and sensitivity of the Ictus patcher. The default minimum and maximum tempo objects are set to 75 and 130. The Ictus Sensitivity Level slider changes the amount of vertical movement needed to trigger an ictus. The default is 30mm, but it can be set as high as 130mm with this slider.

As mentioned above, a piece of music is loaded into Ableton Live, using Ableton's built-in sequencer.

The outputs of the MIDI instrument tracks in Ableton are forwarded to the MIDI inputs of one instance of Kontakt, which is loaded onto one MIDI track. Stock sampler instruments, provided with Berklee's software bundle, are used almost exclusively, with the only exception being CineHarp Pluck samples, which are free. I opted to use these samples because I could not afford better ones, and, as the system needs to be able to run on my laptop, I could not rely on the sample libraries on the lab or studio computers.

Fig. 12 - Kontakt window for instruments in the Curiosibot Concerto

I wrote the plugin shown in Figure 10 and instantiated it on each MIDI instrument track. This plugin intercepts the MIDI notes playing on each channel and overwrites new velocity information, allowing for real-time manipulation of velocity by the conductor.

Audio tracks are played back (using Ableton's audio warping algorithms) at the tempo set by the conductor. A method of allowing the conductor to affect the volume of audio tracks in real time will be added in later versions of the VCE.

To summarize the VCE, a conductor steps in front of a Kinect sensor. Information about his/her body position is captured and forwarded to Max. Sequence playback is started. The VCE software interprets the conductor's movements and uses them to determine tempo and dynamics of the music.

When the VCE is combined with Curiosibot (Pierluigi Barberis's project), the robot follows tempo changes from the VCE, becoming a part of the orchestra.

Innovative Aspects of the WORK

While this was not the first attempt ever made at a virtual conducting system, I approached it as a conductor first and a technologist second. I focused on making a system that would be responsive to the ways conductors actually move, rather than trying to shoehorn a conductor's movements into a narrow space. The VCE does not require a conductor to hold anything in his/her hand, and the Kinect allows the conductor to make large movements without being restricted to a tiny space - previous attempts at this kind of system required a conductor to hold something or be confined to a small space. Also, as this project became one half of Robaton, it is innovative in that it allows the user to control a robot with conducting gestures.

New Skills Acquired

Through my work on this project, I learned a great deal about conducting, software development, systems integration, and gesture control systems. I was forced to think like a computer does: for example, how exactly would a computer interpret a change in a hand's vertical movement? A person watching someone's hand moving can make that distinction, but how do you make a computer recognize it? I acquired skills on how to approach gestures in a way that computers can understand them, which is something I plan on pursuing long after my time at Berklee Valencia is complete.

I acquired the ability to decipher and adapt existing software, as I had to do with the "ModifiedMonty" patcher shown in Figure 8. I gained the ability to communicate with composers to write music, as I had never commissioned a piece of music before.

As Pierluigi Barberis and I worked together on the Robaton project, I gained skills in working with someone as equal partners on a major project, and learned how to program software to interface with a robot. I acquired skills in website design, in pitching ideas concisely and effectively, and preparing and presenting a project at a trade show.

I also learned a great deal about video editing and honed my presentation skills. Additionally, I learned about time management and expectation management from this project.

Challenges, both Anticipated and Unexpected

This project was riddled with challenges of both types. The Kinect sensor acts finicky in different lighting conditions, which is a problem I did not expect at first, but grew to expect problems each time I set up the system. I encountered a challenge with Max for Live, in that when you need to change something, Max for Live forces you to open a patch, make your changes, save, and quit Max before the change will be applied, making it act more like a traditional programming language that needs to compile code.

Some anticipated challenges were getting the computer to react like an orchestra would, changing one little thing which would then "break" the program, and chasing bugs in inadequately documented or poorly supported software.

Future Ramifications and/or Plans for the Work

I hope to use the knowledge and skills I acquired during this process and apply it to further exploration of gesture control technology in music. Gesture control has become my "pet cause" this year, and even though sometimes it's scary and frustrating to wade through documentation of technology in its infancy, it's exciting to me and I've never wanted to shy away from it or give it up.

I feel that gesture control in music performance and composition can help people express themselves in ways that they haven't been able to before. 2014 is an exciting year for gesture control technology - the Kinect, Myo Armband, Hot Hands, Leap Motion, etc., are all very accessible, inexpensive, and incredibly powerful with just a bit of programming chops. Lots of people talk with their hands - it helps them express their feelings in a way that doesn't come across solely with their words. I hope to continue my work in gesture control for music applications to help composers and performers express themselves more freely and naturally.

Conclusion

The Virtual Conducting Experience works well both as a standalone project and as one half of Robaton. Throughout the course of this project, I picked up skills in several disciplines, and I will apply these skills to future projects. While using gestures to control music is nothing new, advances in technology and innovative uses for existing technology will continue to allow musicians to push the envelope of expressive performance.