Gesture Recognition for Virtual Orchestra Conducting Using the Kinect

Edward Ly
Earlham College
801 National Road West
Richmond, Indiana 47374
esly14@earlham.edu

Keywords

gesture recognition, conducting gestures, beat pattern gestures, music interfaces

1. INTRODUCTION

The idea that one could conduct a virtual orchestra is nothing new. Indeed, since the debut of the Buchla Lightning in 1991, the possibility has been ever present, but the limits of existing technology make such a possibility impractical in a mass setting. Nevertheless, the problem of virtual conducting in computer music research remains to be developed. While there have been a number of efforts made in this realm using a wide variety of available hardware, there is yet to be a virtual conductor that is powerful enough to be suitable for live performance, whether it be in a concert hall or in a recording studio. The Microsoft Kinect, which debuted in 2010, is one recent piece of hardware that we believe paves the way forward for further development into this problem.

There has been plenty of interest in and research into the Kinect from the academic and open-source communities, with its motion tracking capabilities having particular interest for virtual conducting applications. OpenNI has been one such framework for accomplishing this task until around 2013, when Apple bought PrimeSense and the OpenNI framework. The Kinect for Xbox One was also released that year, and as compatability issues have been raised between the two versions of the Kinect, open-source development around the Kinect has been stagnant as well. In the following year, the original OpenNI website[1] was shut down, and many other associated libraries, such as the NITE middleware that provides skeleton tracking for OpenNI, are no longer available for download. While the OpenNI 2 binaries and source code are still currently available,[2] NITE has not. For this reason, an open-source alternative for OpenNI and NITE must be established.

In the next section, we will highlight and compare several papers detailing the different implementations that have been attempted while listing some of the benefits and drawbacks of each system. Afterwards, we will propose our own system that addresses some of the prior concerns while using one possible alternative for OpenNI and NITE, and then outline the tasks to be done in order to accomplish building this system.

2. PREVIOUS RESEARCH

2.1 Earlier Hardware

In 2006, Lee et al. used modified Buchla Lightning II batons to create a system that controls the tempo, volume, and instrumental emphasis of an orchestral audio recording [4]. Additional control of the playback speed of the accompanying orchestral video recording has been implemented as well. The gesture recognition itself uses a framework called Conducting Gesture Analysis (CONGA) to detect and track beats, while a variation of a phase vocoder algorithm with multi-resolution peak-picking is used to render real-time audio playback. While video control will likely be outside the scope of our research, volume control and instrumental emphasis can provide an added sense of realism on top of what the Kinect already provides. In the ten years since then, however, the Buchla Lightning has been discontinued, making the CONGA framework obsolete as well. In addition, the framework itself, while touting a recognition rate close to 100 percent, has a latency described as "acceptable for non-professionals, [but] professionals will find the latency much more disturbing" [3]. Indeed, the system has made only one public appearance in a children's museum that same year.

In 2012, Han et al. developed their own virtual conductor system that relied on ultrasound to gather 3D positional data [2]. Hidden Markov models were implemented to process the data and recognize the gestures that would then control tempo, volume, as well as instrumental emphasis. Compared to the Kinect, their model is more simplistic in that the computer only has to recognize the position at one point on a baton rather than at multiple points on the body. We hypothesize that this may reduce CPU load significantly, allowing the system to be accessible to more consumer hardware. However, there is no mention of the amount of latency that is involved at any stage in the system when the gesture recognition is touted to be reliable with about 95 percent accuracy. Further research will need to determine the amount

[1] `http://www.openni.org/`

[2] now `http://structure.io/openni` for the binaries and `https://github.com/occipital/openni2` for the source

ACM ISBN 978-1-4503-2138-9.
DOI: 10.475/123_4

of latency of this system and whether or not this system remains viable for use in a live performance.

In 2014, Pellegrini et al. proposed yet another gesture recognition system using RGB/depth cameras, but they use this system for the specific purpose of soundpainting, composing music through gestures in either live or studio environments [6]. Hidden Markov models are also used here to detect a custom set of gestures and to trigger a variety of different musical events. Gesture recognition for the purpose of creating custom virtual instruments is a problem in computer music research that bears a large resemblance to the problem of conducting a virtual orchestra, as both attempt to manipulate sound in some shape or form. However, tempo is not as much of a factor when playing a musical instrument, and when a song is already predetermined, such as in a live orchestral performance, being able to detect beats and maintain tempo is crucial.

2.2 Microsoft Kinect

One of the first known uses of the Kinect for the purpose of virtual conducting was in 2014, when Sarasúa and Guaus developed and tested a computer's beat-detection capabilities using the Kinect [7]. The application was built with the ofxOpenNI module,[3] a wrapper around OpenNI as well as NITE and the SensorKinect module. Human participants were also involved to contribute to the computer's learning capabilities, even though human error and time deviations had to be taken into consideration. This approach is especially useful as a starting point given that the beat-detection algorithm is about as simple as calculating the current amount of vertical acceleration and finding local minima or maxima. Their application, however, only serves to test the effectiveness of the algorithm and not yet have the music react to the change in tempo given by the live placement of beats. Our project aims at the very least to include the ability for the music to react to said beats.

The following year, Graham-Knight and Tzanetakis use the Kinect for yet another purpose, namely for creating touchless musical instruments for people with disabilities [1]. Here, the positional data from the Kinect is sent to the visual programming language Max/MSP through the Open Sound Control (OSC) Protocol for analysis and playback. While the system does react to the gestures being made, the application usually requires more than one attempt for the gesture to be recognized. Moreover, a bigger limiting factor of this system is the 857 ms average latency, which is too large to be practical for live performances. It is unclear which part of the system contributes the most to latency, whether it be the Max/MSP language or the Kinect itself. Nevertheless, our project aims to develop an application with a latency small enough that both the performers and the audience would not notice.

3. OUR DESIGN

Our goal is to build a virtual conductor application for the Kinect that requires nothing but open-source, cross-platform libraries. To achieve this, we will use the XKin library[4] developed by Pedersoli et al. as an alternative to NITE for its gesture recognition and learning capabilities [5]. This library is built on top of the libfreenect library,[5] which is an open-source alternative to OpenNI and continues to be updated by the OpenKinect community even today. OpenGL will also be a requirement in order to provide a GUI for the user as well as to verify and debug the input stream from the Kinect.

Once a gesture has been performed and recognized, there are two possible ways to output audio. The first is to directly specify the sound that is to be played. Libraries such as PortAudio have high-level APIs for generating and outputting audio through ALSA. The second is use JACK to route the gesture trigger message to an external audio application, specifically to a digital audio workstation (DAW) such as LMMS,[6] for ease of composing an orchestral piece as well as ease of manipulation of tempo for the received messages. We will start with the first method to ensure working audio interaction, and then proceed with the second method as time permits.

Regardless, time will only permit the application to be tested and run under Ubuntu 14.04, but the cross-platform nature of all libraries used will allow for future work to add compatibility for Windows and Mac computers as well.

4. TIMELINE

Our plan is to develop the following parts of the application with the following deadlines:

- October 26: Develop a preliminary test build for the application by learning a simple gesture and controlling the playback of a sawtooth wave.
- November 2: Add more complex gestures, particularly conductor gestures, and add more control over the sawtooth wave accordingly.
- November 9: Integrate JACK for routing gesture messages to LMMS for integration with VST instruments and synthesizers.
- November 16: Complete the first draft of the paper.
- November 23: Continue working on application. Complete outline for the poster.
- November 30 or December 4: Presentation
- December 12: Complete the second draft of the paper.
- December 16: Complete the final draft of the paper.

5. REFERENCES

[1] K. Graham-Knight and G. Tzanetakis. Adaptive music technology using the kinect. In *Proceedings of the 8th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '15, pages 32:1–32:4, New York, NY, USA, 2015. ACM.

[2] S. Han, J.-B. Kim, and J. D. Kim. Follow-me!: Conducting a virtual concert. In *Adjunct Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST Adjunct Proceedings '12, pages 65–66, New York, NY, USA, 2012. ACM.

[3] `https://github.com/gameoverhack/ofxOpenNI`

[4] `https://github.com/fpeder/XKin`

[5] `https://openkinect.org/wiki/Main_Page`

[6] `https://lmms.io/`

[3] E. Lee, I. Grüll, H. Kiel, and J. Borchers. Conga: A framework for adaptive conducting gesture analysis. In *Proceedings of the 2006 Conference on New Interfaces for Musical Expression*, NIME '06, pages 260–265, Paris, France, France, 2006. IRCAM — Centre Pompidou.

[4] E. Lee, H. Kiel, S. Dedenbach, I. Grüll, T. Karrer, M. Wolf, and J. Borchers. isymphony: An adaptive interactive orchestral conducting system for digital audio and video streams. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 259–262, New York, NY, USA, 2006. ACM.

[5] F. Pedersoli, N. Adami, S. Benini, and R. Leonardi. Xkin -: Extendable hand pose and gesture recognition library for kinect. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, pages 1465–1468, New York, NY, USA, 2012. ACM.

[6] T. Pellegrini, P. Guyot, B. Angles, C. Mollaret, and C. Mangou. Towards soundpainting gesture recognition. In *Proceedings of the 9th Audio Mostly: A Conference on Interaction With Sound*, AM '14, pages 18:1–18:6, New York, NY, USA, 2014. ACM.

[7] A. Sarasúa and E. Guaus. Beat tracking from conducting gestural data: A multi-subject study. In *Proceedings of the 2014 International Workshop on Movement and Computing*, MOCO '14, pages 118:118–118:123, New York, NY, USA, 2014. ACM.