Gesture Recognition for Virtual Orchestra Conducting: A Literature Review

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

Over the past decade or so, there have been a number of efforts made in the realm of creating reliable virtual orchestra conducting systems, all with varying degrees of success. Hardware from as early as the Buchla Lightning baton to as recent as the Microsoft Kinect have been used to try to accomplish this task, although there is yet to be a system that is powerful enough to be suitable for performance, whether it be in a live concert setting or in a recording studio. In this paper, 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.

2. EARLIER HARDWARE

In 2006, Lee et al. of the Media Computing Group at the RWTH Aachen University in Germany created a system using modified Buchla Lightning II batons [3]. They use gestural recognition to not only control the tempo, volume, and instrumental emphasis of an orchestral audio recording, but also control the playback speed of the accompanying orchestral video recording 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 is an additional feature on top of the audio control, it will likely be outside the scope of our research. Nevertheless, volume control and instrumental emphasis can provide added realism when implemented into a system that supports it.

In 2012, Han et al. from the Advanced Media Lab at the Samsung Advanced Institute of Technology in South Korea developed their own specialized system for gesture recognition [2]. They used ultrasound to gather 3D positional data, and hidden Markov models are used to process the data and recognize the gestures that would then be used to control tempo, volume, as well as instrumental emphasis. Their approach is more simplistic than the Kinect in regards to the positional data being collected. In their model, the computer only has to recognize the position at one point on the baton rather than at multiple points on the body. This may reduce CPU load significantly, allowing the system to be accessible to more consumer hardware. However, there is no mention to 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 relationship between the amount of latency that can be tolerated and the amount of time needed for a computer to recognize a gesture when it is performed.

In 2014, Pellegrini et al. created their own gesture recognition system using RGB/depth cameras for the specific purpose of soundpainting, composing music through new gestures rather than controlling a set orchestra, in either live or studio environments [4]. Hidden Markov models are also used here to detect a custom set of gestures for variety of different trigger events such as "Play" and "Whole Group." They present an alternative application of gestural recognition as a custom virtual instrument, although if there is a desire for the music to be at a certain tempo, there is no mention of how the computer can adapt to this aspect of the music. Tempo recognition is also important when trying to detect beats in preset music.

3. MICROSOFT KINECT

Since the debut of the Microsoft Kinect in 2010, there has been plenty of interest in the device for research purposes, especially for its motion tracking capabilities. Open-source development around the Kinect has been stagnant, however, ever since Apple bought PrimeSense and OpenNI in 2013 and the original OpenNI website[1] was shut down the following year, although OpenNI 2 binaries and source code are still available for download under the Apache 2.0 license.[2]

In 2014, Sarasúa and Guaus from the Escola Superior de Musica de Catalunya in Spain have developed and tested a computer's beat-detection capabilities using the Kinect [5]. To analyze the input data, their software was built with the

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

[2]Currently, 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

ofxOpenNI module,[3] a wrapper around the OpenNI framework as well as the NITE middleware 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 determining local minima/maxima. Their program, however, only serves only to test the effectiveness of this approach without having the music react to the change in tempo given by the live placement of beats, and further research will be needed to add precisely that.

The following year, Graham-Knight and Tzanetakis from the University of Victoria use the Kinect for a different 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 successfully react to the gestures being played, it 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 latency, but it can be instructive to determine if there is a similar implementation that has a latency small enough for both the performers and the audience not to notice.

4. CONCLUSION

While research prior to the Kinect has determined the different aspects of music that a conductor is able to control and how such control can be achieved, the hardware has struggled to keep up with the demand. Moreover, while the Kinect sounds like an attractive choice for developing a virtual conductor, or for any other application, the proprietary nature surrounding the device means that there has yet to be any robust standard libraries for developing open-source projects around the Kinect. Nevertheless, Sarasúa and Guaus [5] showed that a virtual conductor on the Kinect is possible, and further development is surely needed to realize that possibility.

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] 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.

[4] 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.

[5] 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.

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