EG UK Computer Graphics & Visual Computing (2019)
G. K. L. Tam and J. C. Roberts (Editors)

Comparing Gestural Interfaces using Kinect and OpenPose

Aminur Rahman, Louis G. Clift and Adrian F. Clark

Computer Science and Electronic Engineering
University of Essex, Colchester CO4 3SQ, UK

Abstract

We describe the implementation of a gesture recognition facility for navigating through virtual reality applications in a shared VR facility. An implementation based around the Microsoft Kinect is described and the fruits of several years' experience are summarized. An alternative implementation based around the OpenPose library is then presented and the two are compared.

CCS Concepts

• ***Human-centered computing*** → ***User interface programming;*** *Virtual reality; Collaborative and social computing systems and tools;*

1. Introduction

One way for a group of people to experience a virtual environment together is for them to interact with it on a large display. The question then arises as to the best way to allow them to navigate through that environment. One attractive choice is to use gestures: they avoid the user needing to carry a peripheral such as a tablet; gesturing is reasonably natural for inexperienced users; and when done well, it allows the users to concentrate on the purpose of the virtual environment. This paper is concerned with the development of such an interface by two different mechanisms, describing first the use of a Microsoft Kinect and then an alternative built using a simple camera and OpenPose. The two approaches are then compared.

The virtual reality facility in which this work was performed is approximately $10 \times 4.2 \times 2.4$ m in size. An entire 4.2×2.4 m wall consists of a back-projection screen upon which images can be displayed by a pair of high-performance projectors, each capable of displaying 4096×2400-pixel images and driven by separate workstation-class machines. Updates to what is projected are synchronised to the millisecond using genlock. The projectors are fitted with wavelength multiplex visualization filters and the images they produce aligned to under 1 mm on the screen. Observers within the space wear spectacles that contain dichroic lenses so that a good stereoscopic impression of depth is achieved. A thin software layer that wraps around OpenGL and GLUT synchronises viewpoint updates across a local network.

2. Gestural interface design

At its simplest, motion through a virtual reality model can be achieved by taking 'steps' forward or backward and by turning a little to the left or to the right. Our aim was to have one of a group of people navigating them all through the world, so there also needs to be the ability to take control of the gesture interface and then to release it — coming to six gestures in total. The gestures for taking and releasing control need to be unusual, so that they do not happen accidentally; but when control of the gesture interface has been obtained, the remaining gestures need to be natural. Finally, we wanted the gestures to be static body poses rather than dynamic motions with an update rate of about two per second.

When using a camera-based interface to detect gestures, the gestures need to be gross rather than subtle. Both the Kinect and OpenPose identify users' skeletons, so gestures relative to a subject's own skeleton ("hold out your left arm between hip and shoulder height") are better than absolute ones ("your left arm needs to be 1.2–1.8 m above the floor level"). The following gestures were identified:

Taking control: *raising the left hand above the shoulder.*

Releasing control: *raising the right hand above the shoulder.*

Turn left: *holding out the left hand between hip and shoulder height.*

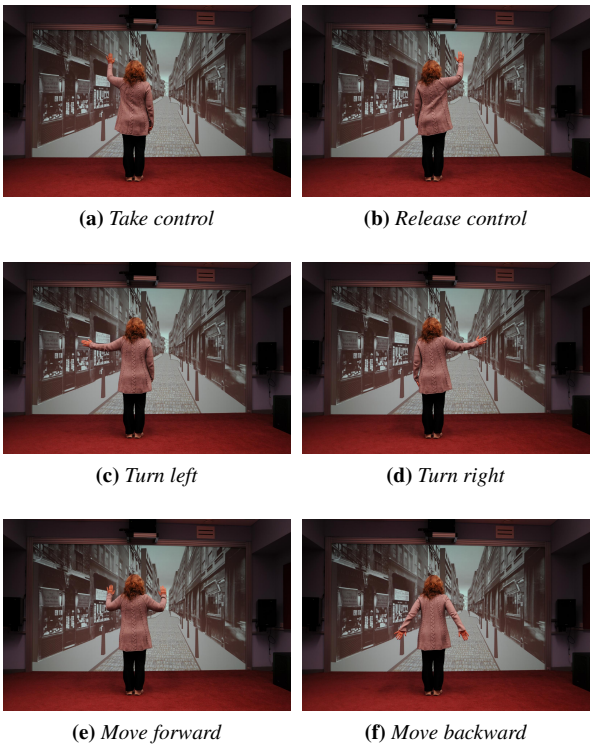
Turn right: *holding out the right hand between hip and shoulder height.*

Move forward: *hold both hands in front between hip and shoulder.*

Move backward: *hold both arms straight backward between hip and shoulder level.*

The gestures are summarized in figure 1; all fall into the category of *symbolic gestures* in the taxonomy of [RS91]. About 20% of users initially try to turn by rotating an imaginary steering wheel, a *pantomime gesture* in that taxonomy, but they quickly become accustomed to using the correct gesture. Further discussion of the gestures appears below.

DOI: 10.2312/cgvc.20191264



(a) *Take control* **(b)** *Release control*

(c) *Turn left* **(d)** *Turn right*

(e) *Move forward* **(f)** *Move backward*

Figure 1: *Gestures used to control motion through a virtual world*

3. Kinect and OpenPose implementations

The Microsoft Kinect was designed as a gaming peripheral. The original Kinect projects an infra-red pattern into the world and uses displacements of that pattern as a way of inferring depth. As the Kinect is also equipped with a visible-waveband camera having the same field of view, it can yield images whose pixels contain not only red, green and blue values but also the corresponding depths. A Kinect was affixed above the back-projection screen and complete program to interface with the Kinect, interpret the gestures and send the resulting motion commands to the VR system was written using its Python SDK interface, some 212 lines of code.

This hardware and software have been used for just under 6 years at the time of writing, and by several hundred people of all ages: the youngest user was about three years old and the oldest over ninety. Some shortcomings have been identified over the years and are discussed in section 4.

OpenPose [CSWS17] is a recently-developed SDK that is able to identify the pose of many people in an image. Given suitable hardware, it is able to run at video rate on live imagery; and recent developments have brought the computational requirements within the reach of a laptop PC graphics card. The 2018 release [CHS*18, Kim] of OpenPose introduced a Python-accessible SDK which was used in this work. OpenPose is inherently 2D, unlike the Kinect, but 2D joint locations are sufficient for recognising the gestures of interest. The code is currently about 700 lines of Python, though it contains more functionality than the Kinect code.

4. Comparing the approaches

Our long experience with the Kinect identified three common failure modes. Firstly, the user needs to stand a little closer to the Kinect than other users of the space in order to be distinguished, we believe because the skeleton location algorithms struggle to segment the user from the background if this is not the case. Secondly, motions within the group of participants is quite frequently recognised as taking control. Somewhat strangely, such motions are rarely recognised as releasing control. Finally, side to side movement of the person with control can be interpreted as gestures. Beyond these, the Kinect-based approach has proven to be reliable. The cause of all three failure modes can be attributed to shortcomings in the underlying analysis of the RGB-D imagery but the SDK that implements the skeletonization algorithms is closed-source so we are not able to improve them; and open-source projects have not yielded better solutions.

The OpenPose solution requires significantly more processing power: the underlying algorithms need GPU support in order to run in real time. Our initial experiments suggest that its robustness is comparable to the Kinect. The most significant difficulty with OpenPose is that the skeletons it identifies are identified by a number, but that number is allocated in a raster scan way, from top left to bottom right. Hence, a small vertical motion of participants in the field of view of the camera will often result in the controlling user being identified differently. We are attempting to overcome this problem by tracking skeletons across several video frames.

5. Conclusions

The Kinect is an excellent peripheral which has allowed the authors to interact with virtual environments using gestures for several years. However, its obsolescence, our identification of its shortcomings, and the emergence of OpenPose have all contributed to our exploring the latter as an alternative. Our initial experience is that the OpenPose solution is about as accurate at recognising gestures as the Kinect and its failure modes are more easily overcome. Our current gesture recognition algorithm is rule-based but, for more general applications, we have experimented with fuzzy-based gesture classification with some success [CLHC18]. Work to explore this is already under way.

References

[CHS*18] CAO Z., HIDALGO G., SIMON T., WEI S.-E., SHEIKH Y.: OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008* (2018).

[CLHC18] CLIFT L. G., LEPLEY J., HAGRAS H., CLARK A. F.: Autonomous computational intelligence-based behaviour recognition in security and surveillance. In *SPIE Proceedings* (2018), vol. 10802.

[CSWS17] CAO Z., SIMON T., WEI S.-E., SHEIKH Y.: Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR Proceedings* (2017).

[Kim] KIM I.: TF pose estimation. `https://github.com/ildoonet/tf-pose-estimation`.

[RS91] RIMÉ B., SCHIARATURA L.: Gesture and speech. In *Fundamentals of nonverbal behavior*, Feldman R. S., Rimé B., (Eds.). Cambridge University Press, Jan. 1991, pp. 239–281.