2014

Rapid Development of Natural User Interaction using Kinect Sensors and VRPN

Timothy B. Morgan
Iowa State University, tbmorgan@iastate.edu

Diana Jarrell
Iowa State University

Judy M. Vance
Iowa State University, jmvance@iastate.edu

Rapid Development of Natural User Interaction using Kinect Sensors and VRPN

Abstract
The availability of low-cost natural user interaction (NUI) hardware took a significant step forward in 2010 with the introduction of the Microsoft Kinect. Despite significant work on the available software development kits for the Kinect, tasks beyond simple single-Kinect skeleton tracking remain challenging to implement. This paper introduces a software tool that significantly accelerates the prototyping and implementation of NUI in virtual reality, particularly for developers with limited programming skills. This is achieved by creating a graphical user interface to provide a consistent development environment for defining Kinect settings and voice commands. This is coupled with a server to transmit skeleton and voice information using the Virtual Reality Peripheral Network (VRPN). Furthermore, the system is capable of combining data from multiple Kinect sensors into one data stream, abstracting the implementation details so the designer may focus on the environment creation and development.

Keywords
Natural user interaction, Microsoft Kinect, depth camera, virtual reality, Virtual Reality Applications Center

Disciplines
Computer-Aided Engineering and Design | Graphics and Human Computer Interfaces | Other Mechanical Engineering

Comments
This is a manuscript of a conference proceeding from IEEE Symposium on 3D User Interfaces (2014): 1, doi:10.1109/3DUI.2014.6798871. Copyright IEEE.

Rapid Development of Natural User Interaction using Kinect Sensors and VRPN

Timothy B. Morgan*
Virtual Reality Applications Center
Iowa State University
Ames, IA

Diana Jarrell†
Virtual Reality Applications Center
Iowa State University
Ames, IA

Judy M. Vance‡
Virtual Reality Applications Center
Iowa State University
Ames, IA

ABSTRACT

The availability of low-cost natural user interaction (NUI) hardware took a significant step forward in 2010 with the introduction of the Microsoft Kinect. Despite significant work on the available software development kits for the Kinect, tasks beyond simple single-Kinect skeleton tracking remain challenging to implement. This paper introduces a software tool that significantly accelerates the prototyping and implementation of NUI in virtual reality, particularly for developers with limited programming skills. This is achieved by creating a graphical user interface to provide a consistent development environment for defining Kinect settings and voice commands. This is coupled with a server to transmit skeleton and voice information using the Virtual Reality Peripheral Network (VRPN). Furthermore, the system is capable of combining data from multiple Kinect sensors into one data stream, abstracting the implementation details so the designer may focus on the environment creation and development.

Keywords: Natural user interaction, Microsoft Kinect, depth camera, virtual reality.

Index Terms: H.5.1 Multimedia Information System – Artificial, augmented, and virtual realities; H.5.2 User interfaces – Input devices and strategies.

1 INTRODUCTION

The difficulty of integrating input devices is not new. In the past, as new input devices became available, custom software was needed to implement the features of the new devices into existing applications. Furthermore, different applications using the same device often use the device in different ways. For example, different programs require different voice recognition vocabularies. In a worst case scenario, these differences are hard-coded into the software, preventing non-programmers from changing the interaction.

One advance that has helped reduce device implementation problems is VRPN [1]. VRPN provides a consistent interface for the most common types of inputs in virtual reality. While a server still needs to be written for each device, once a server is written, it can be used with any virtual environment that supports VRPN. Furthermore, because VRPN operates over a standard computer network, the processing required for an input device can be offloaded to another computer, freeing up more processing power on the rendering computer for other tasks. Due to the advantages of VRPN, programmers have implemented VRPN servers for many low cost input devices. Pavlik has released VRPN drivers for both the Razer Hydra and the Nintendo Wii Remote (WiiMote) [2]–[4].

The Kinect is another low cost input device that supports non-contact NUI in virtual environments. The Flexible Action and Articulated Skeleton Toolkit (FAAST) by Suma et al. provides a VRPN server for the Kinect [5]. FAAST allows users to implement full skeleton tracking and provides basic support for hierarchical gestures. While this is a significant step towards providing easy to use interaction, it does not provide access to one of the major abilities of the Kinect, namely the microphone array for voice recognition. Furthermore, FAAST does not support the data integration of multiple Kinects. This paper describes the development of a software tool to support rapid development of NUI using the Kinect through VRPN.

2 IMPLEMENTATION

The server was implemented using the official Microsoft Kinect for Windows SDK version 1.8 and the Microsoft C# .NET. To implement VRPN in the server, VanderKnyff's VrpnNet library was used [6]. However, a custom version was compiled to fix a bug that was discovered while implementing the server (this build is available at https://github.com/vancegroup).

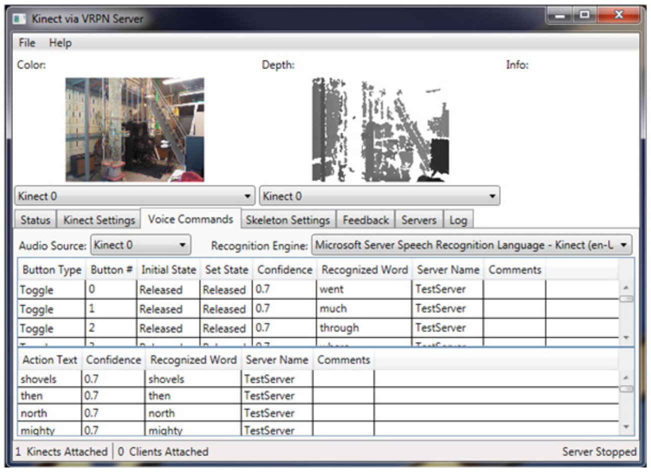
Due to the limited programming experience of many virtual reality users, the Kinect via VRPN tool was designed to operate from a GUI. The GUI presents the user with all the pertinent settings for the server such as the position of each Kinect and the vocabulary to be recognized (Fig. 1). Based on these settings, the program translates the Kinect events into VRPN events that can be recognized by preexisting VR software. These settings can be exported and reused in the future. Advanced users also have the option to load these settings through a command line version of the program to support scripting and remote execution on clustered VR systems.

Voice recognition was implemented using the Microsoft Speech Platform SDK v11. This tool provides a programmatic interface to define words and grammar, and returns recognized events with probabilities that a word has been detected. The graphical user interface contains a table where the user can input all the vocabulary and pertinent options. The user has the option to send each recognized word as either a button or a text string. In addition to the voice recognition itself, beam forming and audio source estimation were implemented to enhance the usefulness of the voice recognition.

2.1 Multiple Kinect Skeleton Data

Another key feature is native support for the use of multiple Kinect sensors. In order to integrate skeleton data from multiple sensors, all the skeleton data from all the Kinects must first be

* tbmorgan@iastate.edu
† djarrell@iastate.edu
‡ jmvance@iastate.edu



Kinect 0

Kinect 0

Audio Source: Kinect 0

Recognition Engine: Microsoft Server Speech Recognition Language - Kinect (en-U

Button Type	Button #	Initial State	Set State	Confidence	Recognized Word	Server Name	Comments
Toggle	0	Released	Released	0.7	went	TestServer	
Toggle	1	Released	Released	0.7	much	TestServer	
Toggle	2	Released	Released	0.7	through	TestServer	

Action Text	Confidence	Recognized Word	Server Name	Comments
shovels	0.7	shovels	TestServer	
then	0.7	then	TestServer	
north	0.7	north	TestServer	
mighty	0.7	mighty	TestServer	

1 Kinects Attached | 0 Clients Attached

Server Stopped

Figure 1: The main page of the GUI.

transformed into a common coordinate system. The choice of coordinate systems is arbitrary, so we chose the direction of the y-axis to be directly opposing gravity, leaving the yaw and translation of the coordinate system user adjustable. This approach allows the pitch and roll of the Kinect to be automatically calibrated. This is achieved by using the Kinect's internal accelerometer to determine the direction of gravity and rotate the Kinect coordinate system such that the gravity vector is aligned with the negative y-axis [2]. The remaining Kinect position and yaw information is sufficiently easy to measure and therefore is left to the user for input.

Once all the skeletons are transformed into a common coordinate system, the positions of each skeleton (the center of mass of the skeleton, not each individual point) are compared with all the skeletons from the other Kinects. Once skeletons from the different Kinects are determined to be from the same person, the joint positions are determined. This process is repeated for all skeletons, from all Kinects, until a merged list which represents each person in the measured area as a unique skeleton is created.

2.2 VRPN Latency

There has also been some concern over the latency that the VRPN interface might introduce into the end application. To test latency a simple VRPN client was written that records the time at which each VRPN message is received. For the testing of voice recognition latency, a user speaks a given word, and then clicks an onscreen button to record the time of the end of the word.

To create a realistic vocabulary for testing, a list of 100 words was generated by selecting 100 random words from the text of Mark Twain's *Adventures of Huckleberry Finn*, as obtained from Project Gutenberg [7]. All 100 words were required to meet the following conditions: 1) four letters in length or longer, 2) not a proper noun 3) no two words on the list could be homophones, 4) not a contraction. The testing was conducted in a room with an ambient sound pressure level of 23 dB with a voice sound pressure level of approximately 42 dB at the Kinect.

The total latency of voice commands averaged 912.4 ms, of which 912.0 ms was the latency of the voice recognition engine itself. The voice command measurement has an error of ± 126 ms and the VRPN measurement has an error of ± 0.5 ms. The latency of the voice engine itself does not appear to be dependent on the size of vocabulary for the range tested. The average VRPN latency is increased from 0.4 ms to 0.8 ms from one to one hundred words, largely due to the increase in the size of the settings list that must be parsed prior to transmitting the command. However, given that in all cases, the latency is less than 0.1% the latency of the voice recognition engine itself, the latency due to the VRPN implementation is considered insignificant.

3 Discussion

The critical measure of success for the Kinect via VRPN software is whether or not it allows novice users to develop user interactions faster than with existing methods. While it has, thus far, been infeasible to conduct user studies to provide quantitative evidence to that end, several cases of anecdotal evidence exist.

The first of these cases was the development of a demonstration of the differences and effectiveness of various low-cost user input devices. In this case, a pre-existing virtual assembly software was used, which had been built prior to the decision to use the Kinect as an input device. Through the use of the Kinect via VRPN software, the developer was able to build Kinect-based user interaction (both voice and skeleton tracking) for the virtual assembly software in 20 minutes, without changing any code in the virtual assembly software.

Another case where the Kinect via VRPN software has found success is in virtual reality education. For a class project on the subject of virtual environments, a student with limited programming background was working to develop software to help young children with word associations while learning English. In this case, the student was able to implement voice recognition in the project using the Kinect via VRPN implementation after being provided only a single, simple example.

4 Conclusion

This development effort has resulted in a simple tool for implementing natural user interaction in virtual reality using multiple Kinects and voice recognition. The latency added by using VRPN to transmit information has been shown to be minimal compared to the underlying latency of the Kinect sensor. All future improvements will be made publicly available at the project website (https://github.com/vancegroup).

Acknowledgments

The authors would like to thank Patrick Carlson and Ryan Pavlik for their many fruitful discussions about the implementation of this project and their work recompiling VRJuggler to support the VRPN text device.

References

[1] R. M. I. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser, "VRPN: A Device-Independent, Network-Transparent VR Peripheral System," in *VRST '01: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, 2001, pp. 55–61.

[2] R. A. Pavlik and J. M. Vance, "A Modular Implementation of Wii Remote Head Tracking for Virtual Reality," in *ASME 2010 World Conference on Innovative Virtual Reality*, 2010, pp. 351–359.

[3] R. A. Pavlik, "rpavlik/razer-hydra-hid-protocol," *GitHub*, 2013. [Online]. Available: https://github.com/rpavlik/razer-hydra-hid-protocol. [Accessed: 01-Sep-2013].

[4] R. A. Pavlik, "rpavlik/vrpn," *GitHub*, 2013. [Online]. Available: https://github.com/rpavlik/vrpn. [Accessed: 01-Sep-2013].

[5] E. A. Suma, B. Lange, A. "Skip" Rizzo, D. M. Krum, and M. Bolas, "FAAST: The Flexible Action and Articulated Skeleton Toolkit," in *2011 IEEE Virtual Reality Conference*, 2011, pp. 247–248.

[6] C. VanderKnyff, "VrpnNet 1.1.1," 2008. [Online]. Available: http://wwwx.cs.unc.edu/~chrisv/vrpnnet. [Accessed: 15-Mar-2012].

[7] M. Twain, *Adventures of Huckleberry Finn*. Project Gutenberg, 1885.