5-2010

A Modular Implementation of Wii Remote Head Tracking for Virtual Reality

Ryan A. Pavlik
Iowa State University, rpavlik@iastate.edu

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

A Modular Implementation of Wii Remote Head Tracking for Virtual Reality

Abstract
Virtual reality (VR) environments based on interactive rendering of 3D computer graphics often incorporate the use of position and orientation tracking on the user's head, hands, and control devices. The Wii Remote game controller is a massmarket peripheral that can provide a low-cost source of infrared point tracking and accelerometer data, making it attractive as a PC-based virtual reality head tracking system. This paper describes the development of an extension to the Virtual Reality Peripheral Network (VRPN) software to support the use of the Wii Remote game controller as a standard tracker object in a wide range of VR software applications. This implementation permits Wii Remote-based head tracking to directly substitute for more costly commercial trackers through the VRPN and VR Juggler Gadgeteer tracker interfaces. The head tracker provides up to 100Hz of head tracking input. It has been tested in a variety of VR applications on both Windows and Linux. The discussed solution has been released as open-source software.

Keywords
Virtual reality, tracking, human-computer interaction

Disciplines
Computer-Aided Engineering and Design | Computer and Systems Architecture | Digital Communications and Networking | Graphics and Human Computer Interfaces

Comments
This conference proceeding is from *Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality* (2010): 351, doi:10.1115/WINVR2010-3771. Posted with permission.

Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality
WINVR 2010
May 12-14, 2010, Ames, Iowa, USA

WINVR2010-3771

A MODULAR IMPLEMENTATION OF WII REMOTE HEAD TRACKING FOR VIRTUAL REALITY

Ryan A. Pavlik
Human-Computer Interaction Graduate Program
Virtual Reality Application Center
Iowa State University
Ames, Iowa 50011
Email: rpavlik@iastate.edu

Judy M. Vance
Department of Mechanical Engineering
Virtual Reality Application Center
Iowa State University
Ames, Iowa 50011
Email: jmvance@iastate.edu

ABSTRACT

Virtual reality (VR) environments based on interactive rendering of 3D computer graphics often incorporate the use of position and orientation tracking on the user's head, hands, and control devices. The Wii Remote game controller is a mass-market peripheral that can provide a low-cost source of infrared point tracking and accelerometer data, making it attractive as a PC-based virtual reality head tracking system. This paper describes the development of an extension to the Virtual Reality Peripheral Network (VRPN) software to support the use of the Wii Remote game controller as a standard tracker object in a wide range of VR software applications. This implementation permits Wii Remote-based head tracking to directly substitute for more costly commercial trackers through the VRPN and VR Juggler Gadgeteer tracker interfaces. The head tracker provides up to 100Hz of head tracking input. It has been tested in a variety of VR applications on both Windows and Linux. The discussed solution has been released as open-source software.

Keywords: Virtual reality, tracking, human-computer interaction.

INTRODUCTION

Position and orientation tracking is essential to virtual reality. Head tracking, in particular, allows the calculation of an off-axis perspective projection for interactive three-dimensional (3D) computer graphics. Such a display, whether monoscopic or stereoscopic, provides enhanced depth and distance cues to the user, increasing the sense of immersion and improving task performance [1, 2]. Immersive virtual reality displays can be used in the engineering and design process to allow human interaction with computer-aided design (CAD) data sets more naturally than non-immersive applications.

Recent developments in consumer game systems, such as the Nintendo Wii, have introduced tracking peripherals and game mechanics to a broad audience. The availability of such systems has also presented academic and hobby researchers with access to the Wii Remote hardware, which integrates linear accelerometers, infrared point tracking, digital and analog game controls, and a Bluetooth wireless interface into an affordable single game controller device.

The methods presented here result in a novel modular extension to the Virtual Reality Peripheral Network (VRPN) software that supports the use of the Wii Remote hardware for virtual reality head tracking. This work enables easy implementation of Wii Remote tracking with slightly modified commercial off-the-shelf (COTS) hardware for use within research-grade virtual reality frameworks. It serves as the tracking component of a simple virtual reality system that nonetheless supports a range of existing virtual reality applications designed for more sophisticated tracking systems.

MOTIVATION

The present research is motivated by the need for low-cost VR head tracking. As stereo televisions, stereo movies, game controllers, and 3D interaction devices become more affordable and commonly available, there is the opportunity to leverage these technological advances to support a broad range of virtual reality applications. Current magnetic, ultrasonic, and optical tracking systems are very expensive to consider for wide use. Low-cost VR head tracking would greatly enhance the immersive experience of some of these common stereo configurations.

At Iowa State University, researchers are also developing VR applications to encourage K-12 students to consider careers in science, technology, engineering and math (STEM). Outreach efforts in K-12 education require the availability of an inexpensive, portable tracking solution that can be set up and taken down quickly, easily transported from the lab to presentation sites, and operated independently of the physical configuration of the space. Another area of need is to supplement existing small desktop commercial magnetic trackers for research purposes. These trackers only provide two channels of tracking. Often, desktop VR requires three channels of tracking to accommodate two-handed tracking and head tracking. The ability to combine the existing magnetic tracker with low-cost head tracking provides a cost effective solution for three-channel requirements.

BACKGROUND

T. Pintaric and H. Kaufmann developed an optical position tracking system using infrared light and commercially-available hardware [3]. Their system relies on multiple cameras that are synchronized with a pulsed infrared flash mounted on each camera. Arrangements of retro-reflective, passive markers are placed on position-tracked objects. The goal of their work is to provide broad six degree-of-freedom (6DOF) position tracking for augmented reality and immersive virtual reality, within a room-sized space.

Work by J. C. Lee into novel applications of the Wii Remote hardware served to drive a variety of further research into using the Wii Remote for position tracking. Lee wrote a number of applications, including a sample head tracking application [4], using B. Peek's C# library for access to the Wii remote data under Windows [5]. His solution tracks two IR light-emitting diodes (LEDs) attached to the sides of a pair of eyeglasses. He uses the position data to calculate an off-axis projection that he renders on the user's standard computer monitor using DirectX. Lee's software is widely available and has introduced the potential for using the Wii Mote as a position tracker to a wide audience; however, his software is limited in scope and not easily incorporated into other VR applications.

Lee's head tracking application attaches the IR LEDs to the dynamic object (the user) while keeping the remote itself static. This is unlike the intended use of the Wii game console in which a "Sensor Bar" (actually a pair of IR light sources located a known distance apart) is placed in a stationary location above or below the user's television. In such a setup, moving the Wii Remote itself drives the interaction. As Lee explains, a setup with a fixed Wii Remote and moving IR markers instead results in a system that is primarily sensitive to translation, and less sensitive to orientation, which suits the needs of desktop VR head tracking. A strength of his solution is the total ease-of-use, including the fabrication of the required hardware. Even the LED-augmented glasses are commercially available, and can be modified for infrared use by simply swapping the LEDs.

T. Sko and H. Gardner designed a configuration to use a hand-held Wii Remote as a pointing controller in a two-walled virtual reality theatre [6, 7]. They placed multiple sensor bars in the environment to provide a wider usable range for the device. Their solution uses an agent-based algorithm to continually maintain a model of the controller's pose, based on the known physical location of the multiple sensor bars. The Wii Remote can only track up to four points simultaneously, so they spaced the sensor bars to avoid greater than four visible IR points at any time.

Y. W. Chow details a system using two Wii Remotes to provide both head tracking and controller input to a virtual environment running on a head-mounted display [8, 9]. As with the work by Lee, Chow uses a stationary Wii Remote observing head-mounted LED markers to provide head tracking input. In this effort, four IR markers are used to provide increased input data sufficient to drive the POSIT algorithm [10] and provide accurate pose estimation. The research also included a verification of the poses returned by the Wii Remote-based tracker by comparison with pose estimates from a commercial magnetic tracker. The system was able to produce estimates with less than one centimeter of average position difference and less than one degree of average orientation difference.

Work by Wognum permits Wii Remote head tracking using VRPN on Windows only [11]. His work ports the algorithm used by Lee to the VRPN package, using it as a versatile platform for implementing tracking software. The algorithm interacts directly with the WiiYourself! software library to access the Wii Remote, and uses only the IR point data returned by the controller. As a VRPN tracker driver, it provides compatibility with a variety of VR frameworks. At compile-time, the user must specify whether the remote is above or below the display, as a proxy for the orientation of the Wii Remote and its coordinate system. It requires the user to initiate a reset of the pose estimate through interaction with the tracker server if an inaccurate projection is noticed. It does not use any accelerometer or gravity measurements.

The solution presented here builds on Lee's configuration. It differs from previous work in that a standard software interface is designed which allows the Wii Remote to interface easily with existing academic and commercial VR applications built on open platforms. The input interface interacts with a Wii Remote over an existing standard protocol to provide additional flexibility. In this way, it can also accept recorded or simulated Wii Remote data, or even data from an alternate device preprocessed by a point-tracking computer vision system. This method supports the use of inexpensive hardware, coupled with the Wii Remote, to provide low-cost position tracking in place of more sophisticated commercial tracking devices.

DESIGN CONSIDERATIONS

The design of the solution discussed here was primarily driven by the dual need for tracking in outreach programs and supplementary tracking for research applications. An analysis of the functional requirements led to the selection of three principles to guide the development of the solution.

Limited Scope

The scope was limited to head tracking only, based on the motivating needs. A magnetic tracker can handle hand tracking for research applications, and outreach applications only need head-tracking at this time. This allowed an enumeration of assumptions and constraints to simplify implementation and limit system complexity. Tracked orientations and positions could be restricted to 3DOF translation with rotation about a sensor-normal axis for presentation of valid stereo pairs. The importance of translation was emphasized. Occlusion considerations were minimized, since occlusion of the tracked glasses would imply visual occlusion of the display for the user.

Compatibility

In order to satisfy the dual need for Wii Remote-based head tracking, the solution needed to be compatible with existing VR frameworks, especially VR Juggler. In order to support multiple existing VR frameworks in the same manner as commercial trackers, the application-facing side of the tracker driver needed to be sufficiently abstracted so as to directly substitute for other, more sophisticated trackers. This included presenting a full rigid-body 6DOF pose, even though the basic algorithm presently only supports calculating 4DOF. Unlike the tracking software presented by Lee, this abstraction resulted in a tracker that is fully independent of the actual virtual environment being executed. Furthermore, the output of the tracking module does not necessarily need to drive a head-coupled perspective view. It can also be compared to known values for verification and tuning.

Flexibility and Modularity

The software design principles of modularity and loose coupling were important in the design process to ensure maximum flexibility of the final system. On the Wii Remote-facing side of the tracker driver, accessing data via a *vrpn_Analog_Remote* interface provides loose coupling. In most live uses, the provider of the data is the *vrpn_WiiMote* device that exposes the state of a Wii Remote on Windows and Linux through *vrpn_Analog* channels. However, as any data source matching this interface will suffice, there are a variety of possibilities for data sources. It permits replacement of the underlying Wii Remote access library to permit a more cross-platform implementation. Furthermore, the algorithm's implementation is not strictly tied to the Wii Remote. The Wii Remote's prevalence, high refresh rate, and built-in computer vision processing made it a convenient choice as a data source. However, a computer vision system processing webcam input could also be implemented for use with the tracker.

The tracker operates on a number of values which lie in a known range, so an ideal solution would accept any source of suitable values. A useful implication of this is that verification of the software's latency, accuracy, and precision may be performed by substituting the source of values. A "virtual Wii Remote" may be implemented that calculates the expected sensor data for a given pose, then feeds that data into the tracker to permit comparison of the output and input.

Implementation of alternative pose estimation techniques also suggested an emphasis on modularity. Modularity permits replacement of the basic two LED tracking with a more sophisticated algorithm utilizing Kalman filtering [12], model evaluation, or more advanced techniques to provide a more robust response. The two LED system may even be substituted by a three LED tracker to disambiguate screen-orthogonal translation from rotation around a vertical axis.

Figure 1 shows the potential producers and consumers of the generic data interfaces manipulated by the tracking driver. The contribution described in this article is represented by the green "Tracking Driver Module" box. The primary producer of input data is the live Wii Remote. The primary consumers of tracking data are the user applications shown in orange. User applications written to use VRPN directly may directly consume the tracking output, while the input abstractions of the VR Juggler and Vizard systems serve as direct consumers for their respective user applications. The two pairs of sources and sinks connected by dotted lines are automated testing and verification capabilities made possible by the design features of the described solution.

SOFTWARE PLATFORM

This work builds on a number of existing software systems. The existing virtual assembly application motivating development is written using VR Juggler, an open-source virtual platform for virtual reality software development available at `http://www.vrjuggler.org` [13]. This framework provides a cross-platform method of developing virtual environments in C++ that can be run on multiple operating systems and on virtual reality hardware ranging from standard desktop workstations to immersive stereoscopic projection environments.

Vizard is a commercial virtual reality software platform similar in scope to VR Juggler. It provides virtual reality hardware access and scenegraph structures for VR. Vizard applications are

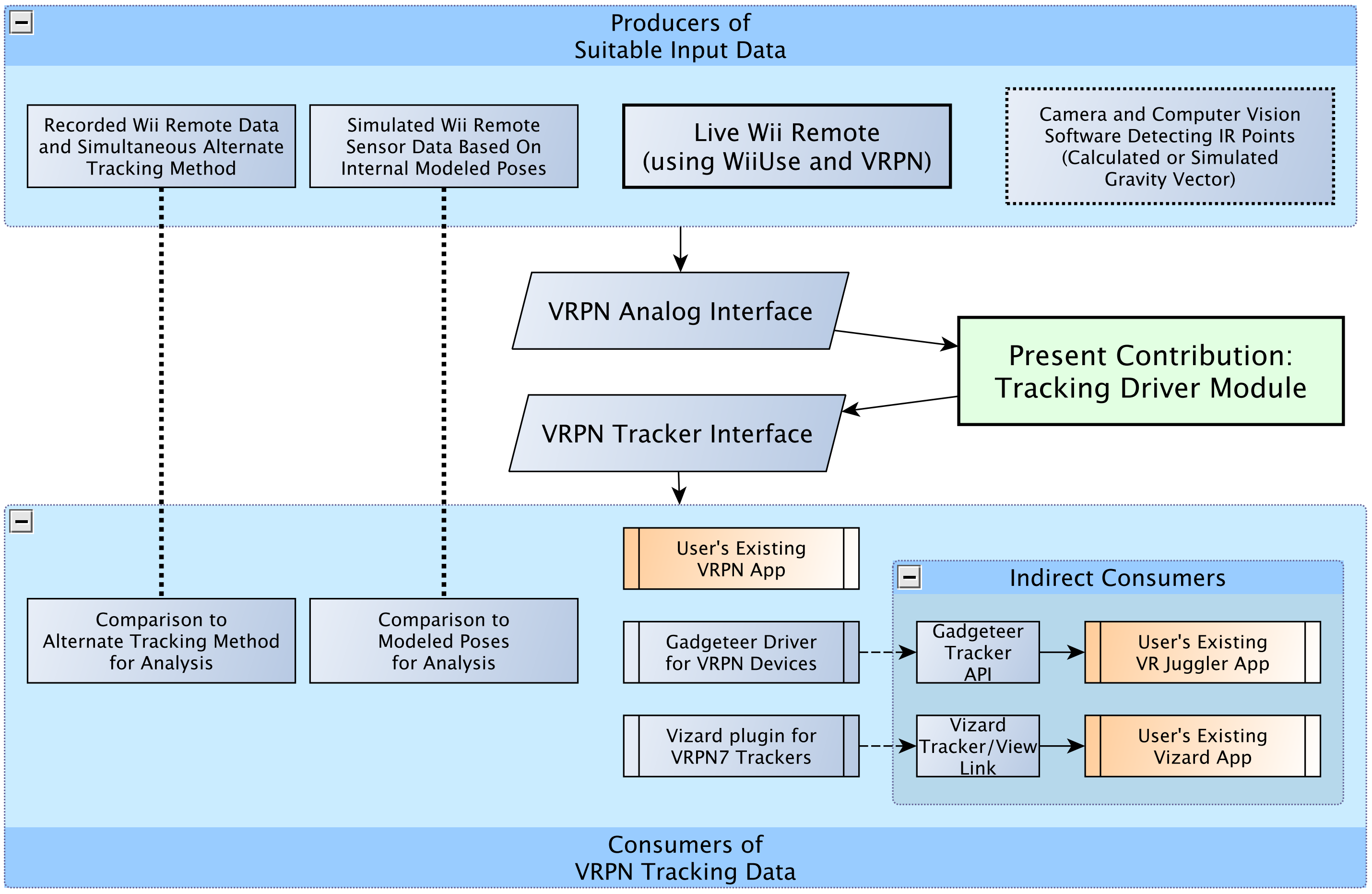

Producers of
Suitable Input Data
Recorded Wii Remote Data and Simultaneous Alternate Tracking Method
Simulated Wii Remote Sensor Data Based On Internal Modeled Poses
Live Wii Remote (using WiiUse and VRPN)
Camera and Computer Vision Software Detecting IR Points (Calculated or Simulated Gravity Vector)
VRPN Analog Interface
Present Contribution: Tracking Driver Module
VRPN Tracker Interface
User's Existing VRPN App
Comparison to Alternate Tracking Method for Analysis
Comparison to Modeled Poses for Analysis
Gadgeteer Driver for VRPN Devices
Vizard plugin for VRPN7 Trackers
Indirect Consumers
Gadgeteer Tracker API
User's Existing VR Juggler App
Vizard Tracker/View Link
User's Existing Vizard App
Consumers of
VRPN Tracking Data

FIGURE 1. **FLEXIBILITY IN SOURCES AND SINKS FOR THE CORE TRACKING ALGORITHM**

written in Python, an interpreted, dynamically-typed language.

The Wii Remote head tracking software itself is implemented using C++ as a module for the Virtual Reality Peripheral Network (VRPN) software package available at `http://www.vrpn.org`. This open-source system provides a transparently networked, generic interface to a wide range of VR devices and input systems [14]. VRPN is used directly in a range of VR applications. Generic VRPN interfaces can also be accessed through a Gadgeteer driver included with VR Juggler, or through the VRPN7 extension included with Vizard. As such, any tracker, analog input, or digital input (button) device accessible using VRPN can also be used in applications built on VR Juggler.

VRPN 07.26 provides a “WiiMote” device exposing the raw data returned using the WiiUse open-source library [15] to access the Wii Remote hardware over Bluetooth in Windows and Linux . This driver provides a number of standard VRPN analog channels to convey the sensor coordinates and relative sizes of up to four IR points, as well as the gravity vector as returned by the three integrated linear accelerometers, among other data.

A sample tracker client application was developed to evaluate the performance of the Wii Remote tracker. Figure 2 shows the head tracker data visualized in this standalone application, with the position and orientation of the teapot matching the pose reported by the head tracker. This client was developed using VRPN directly to access tracker data, along with the GLUI and GLUT graphics tool-kits. In addition, the tracker was used to demonstrate a VR Juggler-based virtual assembly application in which a user can interact with CAD models in a natural, spatial way with haptic force-feedback.

HARDWARE PLATFORM

The VR Juggler software allows the virtual assembly application to run on a variety of systems. The primary system used in this work is a desktop workstation with an Intel Xeon proces-

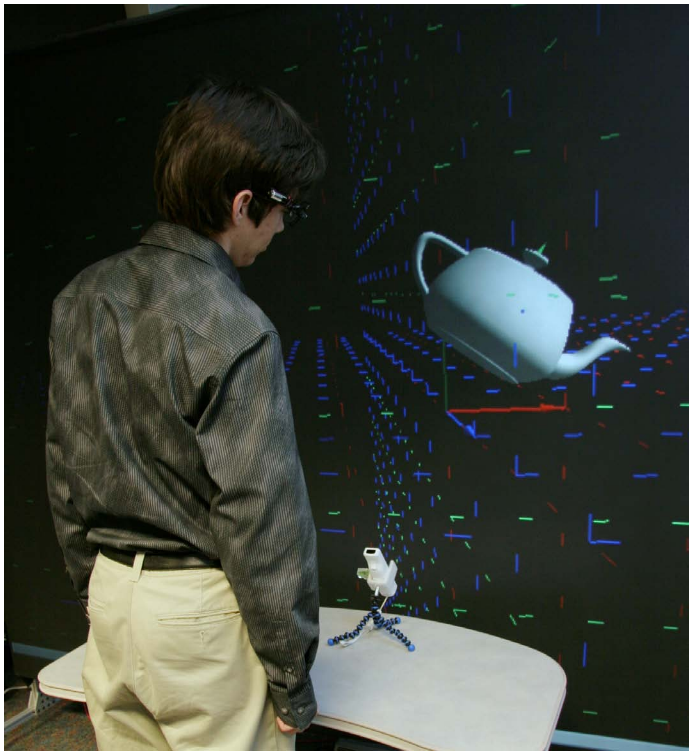


FIGURE 2. **HEAD TRACKING DATA SHOWN AS TEAPOT**

sor and an nVidia Quadro FX 1000 graphics card. Stereoscopic images are rear-projected by a DepthQ 3120 DLP projector displaying 800x600 resolution at 120Hz on an 80cm (32 inch) diagonal screen. The projector is used in conjunction with CrystalEyes active liquid crystal stereo shutter glasses and a Stereographics emitter, providing effectively 60Hz display refresh per eye. A Polhemus Patriot magnetic tracker provides two channels of 6DOF position and orientation tracking in the desktop workspace area.

Another hardware system used for undergraduate engineering education consists of two DLP projectors with circularly-polarized filters to provide passive stereo. A large silvered screen is used in a front-projection setup to provide a stereo viewing experience to a classroom of students. This system, also running on an Intel Xeon-based workstation with Quadro FX 3700 graphics, has no additional trackers besides the Wii Remote system described here.

A third system, consists of an Intel Xeon workstation with an nVidia Quadro FX 5800 graphics card. This system drives one of two display systems. It can display on either a 1680x1050 56cm (22 inch) diagonal widescreen LCD monitor at 60Hz providing a high-resolution mono display, or using a DepthQ 3120 projector, it can provide a stereoscopic display on a 3m (10 foot) diagonal rear-projection screen. This larger screen is used for investigating 1-1 scale interactions in virtual assembly and haptics, and presently features no pose tracking system beyond this Wii Remote system.

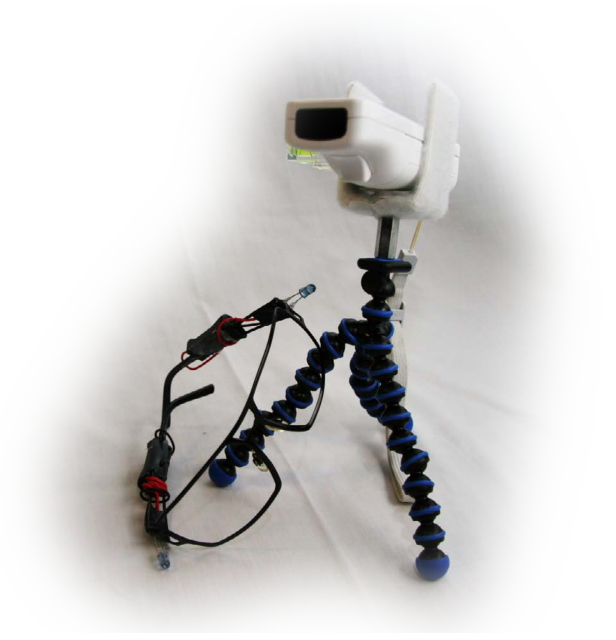
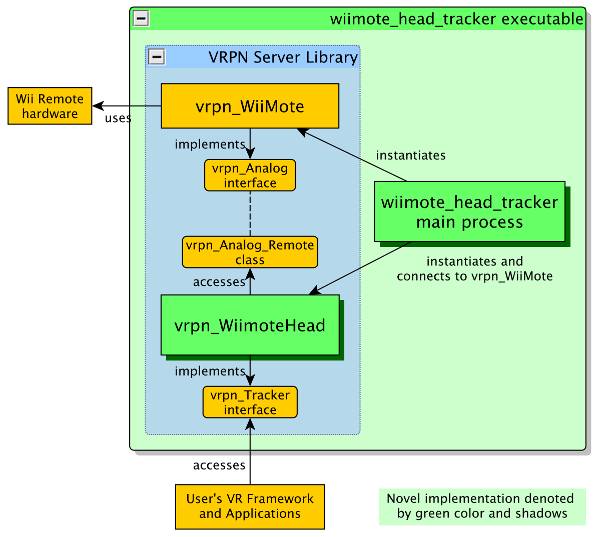


FIGURE 3. WII REMOTE, TRIPOD MOUNT, AND TRACKABLE GLASSES

The present research follows Lee's approach of using a stationary Wii Remote with IR LEDs fixed to the moving object. In both the stereo and mono configuration, the Wii Remote head tracking system relies on two battery-powered infrared LEDs attached to glasses (standard eyeglasses or the CrystalEyes shutter glasses), as shown in Fig. 3. The distance between the LEDs, typically 15cm, must be provided in advance to the tracking software. Since this hardware configuration is similar to that used by Lee, this research extends the utility of glasses designed or modified for Lee's popular application to use in academic and commercial virtual environment frameworks. The Wii Remote hardware itself is available commercially for about US$40. A mounting bracket was easily constructed to attach the Wii Remote to a variety of standard camera tripods.

IMPLEMENTATION DETAILS

The head tracking driver is implemented as a filter device that takes, as input, any source of 15 VRPN analog channels. These channels carry the three components of the measured gravity vector and up to four 3-tuples of size and *x*- and *y*-position of a tracked point on the 1024x768 sensor. While typical usage of the driver will be in conjunction with the *vrpn_WiiMote* driver that exposes these channels among other data, this generic input interface increases the modularity of the overall tracking system by permitting substitution of the *vrpn_WiiMote* device.



wiimote_head_tracker executable
VRPN Server Library
Wii Remote hardware
uses
vrpn_WiiMote
implements
vrpn_Analog interface
instantiates
wiimote_head_tracker main process
vrpn_Analog_Remote class
instantiates and connects to vrpn_WiiMote
accesses
vrpn_WiimoteHead
implements
vrpn_Tracker interface
accesses
User's VR Framework and Applications
Novel implementation denoted by green color and shadows

FIGURE 4. **MODULE-LEVEL VIEW OF TRACKER SERVER**

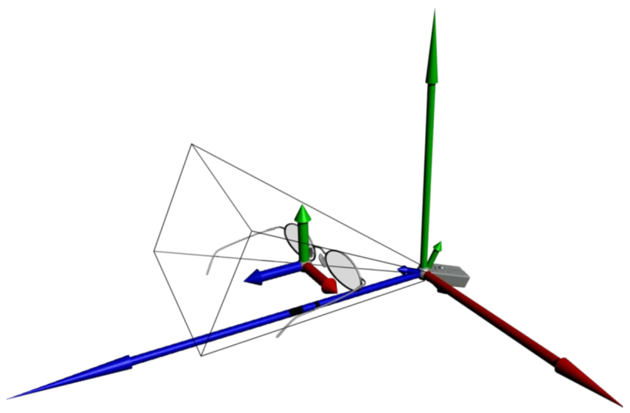


FIGURE 5. **TRANSFORMS AND COORDINATE SYSTEMS**

Implementation of the Wii Remote head tracking driver as a VRPN module permits a variety of uses for the completed code. It can be compiled into a library for direct use by an application, if desired. A standalone server application, shown at the module level in Fig. 4, was produced that allows a user to start using the tracker by simply starting a single application. The server, built against the VRPN server library, creates the vrpn_WiiMote device that uses the WiiUse library to access the device. It then initializes the tracker module with the VRPN device name of the Wii Remote. When started in this way, the head tracker can be accessed as a standard VRPN tracker server from the local host or from across the network. VR Juggler can be configured to access this tracker device using the *VRPN_drv* driver included in the upstream software suite. The position proxy corresponding to the user's head can be set to use this driver as its data source, providing head-coupled rendering.

Regardless of whether the tracker is standalone or part of an end-user application, when it is initialized with an analog device, it registers report callbacks for that device, so that it is notified when new data is available. When a callback is triggered, the raw data is stored in the tracker object for processing. Only the four IR points and the accelerometer data are retained: all other channels, including the buttons and any extension controllers, are ignored.

When the number of valid IR tracked points is equal to 2, calculations are performed using the fixed field of view and the known distance between the LED points to arrive at an estimate of x-, y-, and z-translation, as well as rotation about the sensor-normal z-axis. This calculation step is modular in the code, to support substitution of pose estimation techniques. Calculation proceeds in the coordinate system aligned with the Wii Remote. Later, during the VRPN main loop, the presence of new Wii Remote data or sufficient elapsed time since the last report trigger a final pose preparation and report.

To eliminate the need to adjust for the pitch and roll of the Wii Remote for every installation, an algorithm to achieve gravity correction was developed and applied before calculating the final pose. It is imperative to eliminate Wii Remote pitch and roll variability because, unlike other trackers with a more permanent installation method, the tracker base in this system is easily portable and attached to a tripod-style mount that introduces orientation variability over the long term. Implementing this correction provides a more accurate tracking result without the need for a user-initiated reset, unlike the work by Wognum [11]. The stored accelerometer data is used to rotate the estimated pose to account for the pitch and roll of the Wii Remote, and by extension, its IR sensor, as shown in Fig. 5. The reported pose is thus relative to a virtual tracker whose z-axis is along the projection of the Wii Remote position down on to a plane perpendicular to gravity, with the y-axis directly opposed to gravity and the x-axis proceeding from their intersection in a right-hand coordinate system. Effectively, the user may consider the tracker base to be level with respect to gravity, even when it is not. This approach also permits the user to freely position the Wii Remote at an incline as needed to ensure that head movements stay within the limited field-of-view of the Wii Remote's sensor. This calculation is performed in a "just-in-time" manner to ensure that the freshest data is reported in case multiple reports from the Wii Remote were received since the last main loop execution due to system performance. Finally, the gravity-corrected pose estimate is packed and transmitted as a VRPN tracker pose update.

Figure 6 describes the entire tracking and VR system. The three lanes denote the three concurrent processes; the first occurs in the Wii Remote hardware, while the second and third take place on the computer's processor. The interfaces between

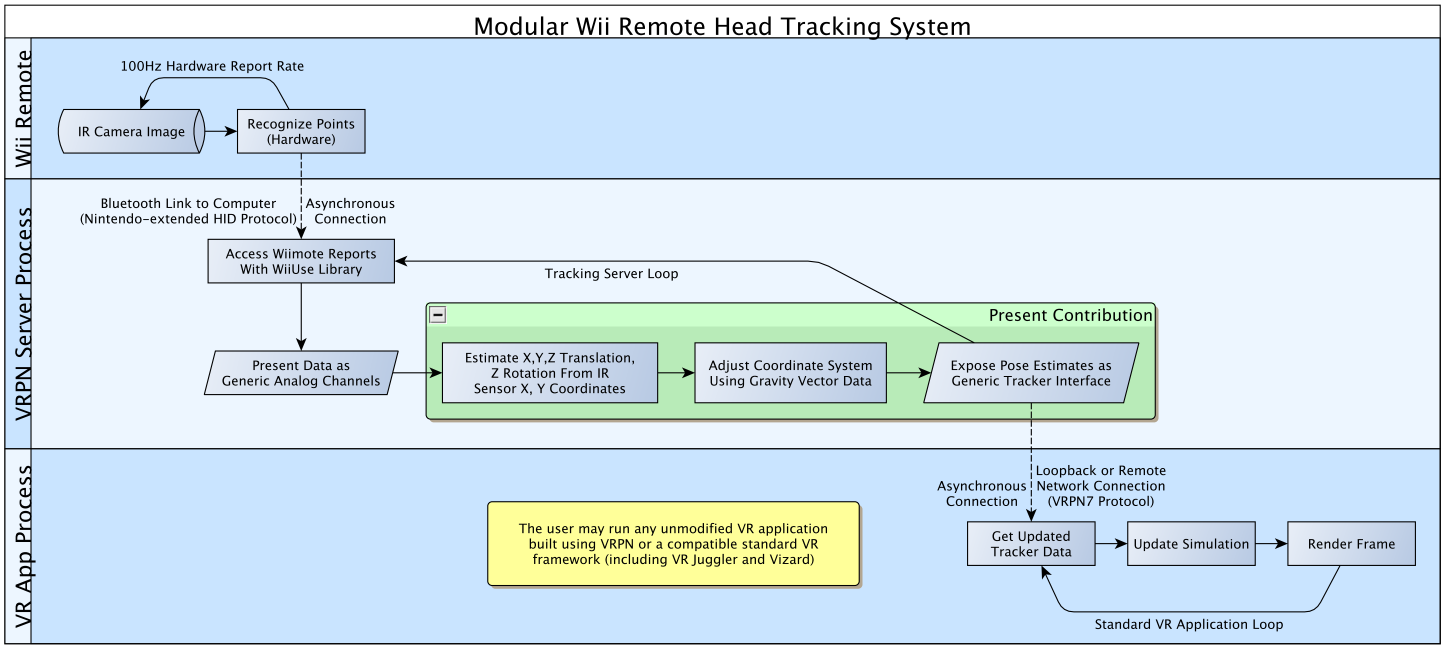

Modular Wii Remote Head Tracking System
Wii Remote
100Hz Hardware Report Rate
IR Camera Image
Recognize Points (Hardware)
VRPN Server Process
Bluetooth Link to Computer (Nintendo-extended HID Protocol)
Asynchronous Connection
Access Wiimote Reports With WiiUse Library
Tracking Server Loop
Present Contribution
Present Data as Generic Analog Channels
Estimate X,Y,Z Translation, Z Rotation From IR Sensor X, Y Coordinates
Adjust Coordinate System Using Gravity Vector Data
Expose Pose Estimates as Generic Tracker Interface
VR App Process
Asynchronous Connection
Loopback or Remote Network Connection (VRPN7 Protocol)
The user may run any unmodified VR application built using VRPN or a compatible standard VR framework (including VR Juggler and Vizard)
Get Updated Tracker Data
Update Simulation
Render Frame
Standard VR Application Loop

FIGURE 6. **FLOWCHART SHOWING CONCURRENCY BOUNDARIES**

all concurrent processes are asynchronous. The elements of the flowchart within the green box are part of the present contribution. They correspond to the steps taken by the tracking filter driver to transform the analog input into a tracker pose estimate.

RESULTS

The described system has been successfully implemented and used for its intended purposes. The tracking device, *vrpn_Tracker_WiimoteHead*, and the standalone server application, *wiimote_head_tracker*, have been contributed back to the VRPN community for inclusion with the package as open source software, downloadable from `http://www.vrpn.org`. The tracker software described is able to update its pose estimate at the full 100Hz rate of the Wii Remote. It provides continual tracking inputs for stereo and mono view calculation by VR Juggler. The tracker's CPU usage on a wide range of desktop workstations is very low. Execution does not interfere with the applications' displays nor with a high-frequency 1000Hz haptic rendering thread updating a force-feedback device in the test virtual assembly application. Running the head tracker server in a standalone mode results in tracking calculations taking place in a separate process from the virtual environment application, which suits the multi-processor architecture of modern personal computers. While the overall system latency has not yet been measured, anecdotal experiences using the tracker to provide head-coupled immersive display confirm the suitability of the head tracking server for the task.

The tracking system is easy to prepare and use. It generally only requires executing the tracker server application before starting the VR application. No run-time configuration or maintenance is required for the driver. Because of the gravity compensation feature, the user can configure the VR system as if the Wii Remote, as the tracker base, were completely level. The level tracker coordinate system simply needs to be located within the virtual world, as with any tracking device. The source of the tracking data should be entered as the VRPN device *Tracker0@localhost* in the most common case. On Linux, the only hardware-related setup required is for the user to press the *1* and *2* buttons on the Wii Remote to place it in visible mode immediately prior to starting the tracking application. Pairing is completed automatically. On Windows, the tested Bluetooth stacks require a more involved procedure immediately prior to each execution of the tracking server. The user must remove any existing Wii Remote pairing from the computer, press the *1* and *2* buttons on the Wii Remote, and initiate a new pairing without a passcode in a short amount of time. This slightly complicates the process of using the tracker on Windows. However, this seems to be an issue with the interaction between the Windows operating system and the modified Bluetooth Human Interface Device (HID) protocol used by Nintendo to communicate with the Wii Remote, rather than any aspect of the implementation of this tracker.

The relatively narrow field of view (FOV) of the Wii Remote sensor has been the most challenging aspect of using this system so far. As no attempts are made with the current pose estimation algorithm to update pose predictions when fewer than two IR points are observed, both the left and right extremes of the user's head must remain within the sensor's view volume or pose estimation stops until they return to view. The Wii Remote was experimentally measured to have a horizontal FOV of $\Theta_h = 43°$and a vertical FOV of $\Theta_v = 32°$. So, given a fixed distance between the LEDs d_{LED} (typically 0.15m), and considering only translation of the user's head, the trackable area of a plane located a distance d away from the sensor and orthogonal to its view vector can be calculated, as given in eq. 1.

$$A(d) = \left(2d\tan\frac{\Theta_h}{2} - d_{LED}\right)\left(2d\tan\frac{\Theta_v}{2}\right) \tag{1}$$

The volume of the tracker's effective region between two such planes (e.g. a minimum and maximum distance) d_1 and d_2, where $d_1 < d_2$, can be calculated as the volume of the view frustum with base of $A(d_2)$ truncated at d_1, as given in eq. 2.

$$V = \frac{A(d_2)\cdot d_2 - A(d_1)\cdot d_1}{3} \tag{2}$$

The limited field of view is mitigated by the gravity autocorrection feature of the implemented driver, as the user may adjust the tilt of the Wii Remote at run-time without further configuration in order to track a greater percentage of likely head poses if loss of tracking in normal motion is noted. This works well in practice in both desktop and large projection situations.

Depending on the specific IR LEDs chosen for the glasses, the emission angle and alignment of the LEDs can somewhat impact performance of the tracker. When battery levels become low, LEDs with a narrow field of view may be less consistently detected by a Wii Remote far above or below the user's head. In practice, manual adjustment to aim the LEDs toward the Wii Remote, as well as replacement of nearly-depleted batteries minimize the impact of this limitation.

FUTURE WORK

In addition to (x, y) coordinates for each sensor point, the Wii Remote returns a size value, with 16 possible values (4 bits). This information is not presently used in the estimation. Integration of filtering, such as a single-constraint-at-a-time (SCAAT) filter [16, 17], may permit improved pose estimation through use of all available data in standard cases, as well as continued tracking updates in case the user moves outside the sensor's field of view.

The addition of a third LED to the glasses, forming a triangle, can be used to determine the difference between a rotation around a vertical axis and a translation along a sensor-normal axis. When using only the sensor location of two LEDs, both of these transformations are observed by the sensor as a decreased distance between IR points and therefore are indistinguishable motions. However, their impact on an off-axis projection rendered using the tracking data is quite distinct. Implementation of a additional tracker module that handles this three LED case is currently being pursued.

The relative utility of the sensor's view volume varies based on the position and orientation of the Wii Remote. Accordingly, analysis of head-tracking data recorded with an alternate tracking system during similar virtual reality tasks on a similar display could be used to create a tool that suggests an optimal position for the Wii Remote to capture the most probable positions and orientations accurately, based on user-input data about the physical configuration of the workstation.

Finally, multiple Wii Remotes are accessible simultaneously over the same Bluetooth protocol. The low cost of the hardware means that even a multi-view setup can be cost-effective for widespread use. An extension to the described filter driver can be implemented that takes two Wii Remote-like inputs and calculates a more accurate pose estimation for the tracked glasses.

CONCLUSION

A dual need for low-cost head position and orientation tracking for use in virtual reality applications led to the design of a loosely-coupled software system specialized for desktop VR head-tracking with cross-platform capabilities and transparent compatibility with existing VR software frameworks. The implementation of a tracker based on the use of IR sensors and accelerometers on a stationary Wii Remote and two IR LEDs mounted on glasses worn by the user builds on the hardware designs and basic calculations from Lee's widely popular work in Wii Remote-based head tracking. The VRPN-based filter module provides for loose coupling on both the hardware-facing and application-facing side. This design enables the system to be used to provide high-rate head tracking input inexpensively to a wide range of existing VR applications compatible with VRPN, including applications based on Vizard or VR Juggler. The standalone Wii Remote head tracker server builds on VRPN and the filter module to provide a tracking data source in a single step, and has been tested to function smoothly on both Windows and Linux platforms.

ACKNOWLEDGMENT

The authors are grateful to Daniela Faas of Iowa State University for assistance in the integration of these results into the virtual assembly research platform; to Meisha Rosenberg and

Max Perkins of Iowa State University for support in hardware design, fabrication, and evaluation; and to the anonymous reviewers for their helpful feedback on a draft of this paper. This work was performed at the Virtual Reality Applications Center at Iowa State University, and was funded by NSF award CMMI-0928774.

REFERENCES

[1] Arthur, K. W., Booth, K. S., and Ware, C., 1993, "Evaluating 3D task performance for fish tank virtual worlds," ACM Transactions on Information Systems, **11**(3), pp. 239–265.

[2] Arsenault, R. and Ware, C., 2000, "Eye-hand co-ordination with force feedback," *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '00*, ACM, New York, NY, USA, pp. 408–414.

[3] Pintaric, T., Kaufmann, H., and Zachmann, G., 2007, "Affordable Infrared-Optical Pose Tracking for Virtual and Augmented Reality," *IEEE VR Workshop on Trends and Issues in Tracking for Virtual Environments*, pp. 44–51.

[4] Lee, J. C., 2008, "Hacking the Nintendo Wii Remote," IEEE Pervasive Computing, pp. 39–45.

[5] Peek, B., 2007, "Managed Library for Nintendo's Wimote," URL `http://www.brianpeek.com/blog/pages/wiimotelib.aspx`.

[6] Schou, T. and Gardner, H. J., 2007, "A Wii remote, a game engine, five sensor bars and a virtual reality theatre," *OZCHI '07: Proceedings of the 19th Australasian conference on Computer-Human Interaction*, ACM, New York, NY, USA, pp. 231–234.

[7] Sko, T. and Gardner, H., 2009, "The Wiimote with multiple sensor bars: creating an affordable, virtual reality controller," *CHINZ '09: Proceedings of the 10th International Conference NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, ACM, New York, NY, USA, pp. 41–44.

[8] Chow, Y.-W., 2009, "Low-Cost Multiple Degrees-of-Freedom Optical Tracking for 3D Interaction in Head-Mounted Display Virtual Reality," International Journal of Recent Trends in Engineering, **1**(1), pp. 52–56.

[9] Chow, Y.-W., 2008, "The Wii Remote as an input device for 3D interaction in immersive head-mounted display virtual reality," *Proceedings of IADIS International Conference Gaming 2008: Design for Engaging Experience and Social Interaction*, International Association for Development of the Information Society, pp. 85–92.

[10] DeMenthon, D. F. and Davis, L. S., 1995, "Model-Based Object Pose in 25 Lines of Code," International Journal of Computer Vision, **15**, pp. 123–141.

[11] Wognum, W., 2008, "Wii headtracking in VR Juggler through VRPN," URL `http://www.xs4all.nl/~wognum/wii/`.

[12] Kalman, R. E., 1960, "A New Approach to Linear Filtering and Prediction Problems," Transactions of the ASME–Journal of Basic Engineering, **82**(Series D), pp. 35–45.

[13] Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Cruz-Neira, C., 2001, "VR Juggler: A Virtual Platform for Virtual Reality Application Development," *VR 2001: IEEE Virtual Reality Conference 2001*, IEEE Computer Society, Los Alamitos, CA, USA, p. 89.

[14] Taylor, R. M., II, Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T., 2001, "VRPN: a device-independent, network-transparent VR peripheral system," *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, New York, NY, USA, pp. 55–61.

[15] "WiiUse," URL `http://wiiuse.net/`.

[16] Welch, G. and Bishop, G., 1997, "SCAAT: incremental tracking with incomplete information," *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 333–344.

[17] Bishop, G., Welch, G., and Allen, B. D., 2001, "Tracking: Beyond 15 minutes of thought," SIGGRAPH Course Pack.