Proceedings of the ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference
IDETC/CIE 2013
August 4-7, 2013, Portland, Oregon, USA

DETC2013-13441

INTERACTING WITH A LARGE VIRTUAL ENVIRONMENT BY COMBINING A GROUND-BASED HAPTIC DEVICE AND A MOBILE ROBOT BASE

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
Greg R. Luecke
Department of Mechanical Engineering
Virtual Reality Application Center
Iowa State University
Ames, Iowa 50011
Email: jmvance@iastate.edu, grluecke@iastate.edu

ABSTRACT

Ground-based haptic devices provide the capability of adding force feedback to virtual environments; however, the physical workspace of such devices is very limited due to the fixed base. By mounting a haptic device on a mobile robot, rather than a fixed stand, the reachable volume can be extended to function in full-scale virtual environments. This work presents the hardware, software, and integration developed to use such a mobile base with a Haption Virtuose™ 6D35-45. A mobile robot with a Mecanum-style omni-directional drive base and an Arduino-compatible microcontroller development board communicates with software on a host computer to provide a VRPN-based control and data acquisition interface. The position of the mobile robot in the physical space is tracked using an optical tracking system. The SPARTA virtual assembly software was extended to 1) apply transformations to the haptic device data based on the tracked base position, and 2) capture the error between the haptic device's end effector and the center of its workspace and command the robot over VRPN to minimize this error. The completed system allows use of the haptic device in a wide area projection screen or head-mounted display virtual environment, providing smooth free-space motion and stiff display of forces to the user throughout the entire space. The availability of haptics in large immersive environments can contribute to future advances in virtual assembly planning, factory simulation, and other operations where haptics is an essential part of the simulation experience.

Keywords: Haptic devices, virtual reality, robotics, human-computer interaction.

INTRODUCTION

The goal of this work is to expand the workspace of a ground-based haptic device to encompass the full area of a multi-wall projection screen or large area position tracked virtual reality facility. The result will support user interaction with force feedback within a large virtual reality facility. To illustrate, the large workspace Haption Virtuose™ 6D35-45 haptic device is designed to provide forces and torques within a cube-shaped workspace 45 cm on each side. Therefore, using this device, only the collision of virtual objects within the 45 cm cube-shaped workspace can provide force feedback to the user. Other small ground-based haptic devices have smaller workspace areas. With today's availability of large area position tracking systems, virtual reality facilities can be constructed of any size. Methods are needed to support using haptic devices in a large area position tracked virtual environment that could potentially include several square meters of floor space. The assembly scenarios of interest take place in a typical work cell of 2 m to 3 m square. Thus, expanding the effective workspace of the haptic device permits richer, more realistic simulations of assembly scenarios by al-

*Address all correspondence to this author.

lowing the user to walk around in the available physical space while still receiving haptic force and torque feedback. The approach is to mount a commercially-available haptic device on a powered omni-directional mobile robot, further develop a control scheme originally designed for smaller devices, and produce modular software that provides this functionality in a way that is easy to use. The major hardware components are commercial, off-the-shelf (COTS) components. It is the integration of the hardware into a full system, the extension of the control system and the software to integrate it all that is novel. The prototype system consists of a Haption Virtuose™ 6D35-45 haptic device, mounted on a stand which is placed on an omni-directional Mecanum-style mobile robot drive base.

BACKGROUND

Body-based haptic devices are inherently mobile since they are grounded to the user; however, these devices can only provide relative forces, such as grasp forces. Ground-based haptic devices are, as the name implied, fixed to the ground so these devices are able to display absolute forces, such as weight. Our desire is to provide the ability to feel absolute forces and torques in a large area virtual reality facility so our research in focused on the use of ground-based devices only.

There are several existing approaches to extending haptic force feedback to a large working area. One approach, involves simply building the haptic device as large as the physical virtual facility work area. Tensed-cable devices are based on a system of cables and motorized pulleys, starting with the pioneering SPIDAR [1], and proceeding on to its family of successors [2] and commercial adaptations of the design [3]. The cables are fastened to a fixed structure and are controlled with motorized pulleys. Within the virtual reality facility, one or more handles are attached to the cables. There is typically one more cable per handle than degrees of freedom in feedback. As the user moves the handle, the pulley motors actuate the cables to provide force and torque feedback and also to encode the position of the handle. One of the strengths of this design is that the size of the frame determines the extent of the force and torque feedback area, and the frame can be built to encompass very large spaces. However, these systems can be complex to control, and the same cables that offer such freedom of workspace size also present singularities which limit the orientations that are reachable and provide feedback [4].

Other approaches consist of the addition of redundant axes to ground based haptic devices to expand the workspace and overcome workspace singularities. Gosselin *et al.* suspended the Haption Virtuose 6D35-40 six degree-of-freedom haptic (6-DOF) from an overhead beam structure mounted to the floor. A DC motor and a back-drivable cable capstan reducer powered translation of the haptic device along the beam [5]. This extra degree of freedom expanded the haptic workspace in one direction along the length of the overhead beam. Borro *et al.* [6] took a similar, but different, approach. The LHIfAM consists of a custom designed haptic device attached to a vertically oriented gantry structure. The structure allows linear translation in the horizontal direction along a cross beam that extends between two fixed vertical end supports, and vertical translation as the cross beam slides up and down along the fixed vertical end supports. Both the Gosselin device and the LHIfAM are well suited to the application situation where a single projection screen displays virtual components for assembly and maintenance evaluation; however, this solution only extends the haptic workspace in one direction or the 2D vertical plane, which is not sufficient for reaching into a more general large scale virtual reality facility such as a CAVE™. Further development of Gosselin's research led to a commercialized device offered by Haption called Scale 1. This consists of a floor mounted structure that contains three orthogonal motorized booms which move the Virtuose 6D35-40 in 3D space [7]. The support structure is positioned outside of a multi-wall immersive environment with one of the supporting motorized beams extending into the space. Ueberle *et al.* developed a specialized device called the VISHARD10 to eliminate workspace singularities present in standard non-redundant 6DOF haptic devices [8]. VISHARD10 is a admittance controlled robotic arm with 10 degrees of freedom and a cylindrical workspace measuring 1.7m in diameter and 0.6m in height. It has the ability to produce large forces within the workspace; however, the cylindrical workspace in this configuration is height limiting and not able to support the standard reach height of a human. Like all the devices presented so far, its ground-based design requires permanent, or at least semi-permanent, installation and the device itself still limits the bounds of the haptic workspace.

One way of adding redundant axes to an existing haptic device to provide a theoretically limitless range of motion is through integrating the haptic device with a mobile robot, to produce what Nitzsche *et al.* call a "mobile haptic interface," or MHI [9]. The particular MHI discussed in Nitzsche *et al.* couples an omni-directional mobile base with a Sensable Phantom Premium 1.0 device. Later the mobile base was used with the VISHARD7 [10], the mobile-targeted successor of the VISHARD10. Barbagli *et al.* explored mounting Sensable Phantom Premium 1.5 devices on two different mobile robot bases [11]. They investigated the performance of those bases and the functioning of the combined control scheme through the use of a simulated user input to the haptic device. Despite the results of the work of Nitzsche *et al.* and Barbagli *et al.*, Gosselin *et al.* de-emphasized the category of haptic devices with mobile bases in the general search for large-workspace haptic interaction in a 2008 review of the field [4], citing problems with slip and the negative visual impact of using such devices in a projection-screen environment. However, alternate robot types and control schemes for mobile haptics interfaces have yet to be explored,

thus providing motivation for the current research presented in this paper.

DESIGN AND IMPLEMENTATION

Transparency when using a haptic device is highly desirable [9]. The user should be free to move about in the space whether or not a virtual object is being manipulated. Additionally, the user should feel appropriate collision forces throughout the space when manipulating an object. An example of the desired experience using the virtual assembly scenario follows. A user holds on to the end effector of the haptic device and walks within the physical confines of the virtual reality facility to the location of a virtual object of interest displayed in the virtual reality system. The user reaches out and virtually "grabs" the part by selecting it using the end effector. The user can then move elsewhere in the space, such as to a workbench, and "release" or place the virtual object with the rest of the virtual objects on the bench. Throughout these interactions, the mobile base and haptic device work together transparently.

To avoid confusion, the following nomenclature will be adopted for the remainder of this work. "Haptic device" shall refer to a ground-based haptic interaction device, such as a Haption Virtuose™ 6D35-45 or a Sensable Phantom Omni®. A coordinate system is defined at some fixed location on the haptic device referred to as the "base." "Mobile robot" shall refer to the powered omni-directional mobile robot.

On a technical level, the system works as follows. As the user manipulates the end effector of the haptic device, the mobile robot moves the haptic device base to follow movements of the user through the space. This expands the usable workspace of the haptic device from the reachable volume of the haptic device to the entire position tracked area in the virtual reality(VR) facility. The mobile robot carries a position tracking target. The tracking system in the VR facility reports the position of the mobile robot relative to the room. Composing the room-to-base and base-to-end-effector position transforms results in the obtaining the overall position and orientation of the end effector in the room. This combined system effectively functions as a haptic device with a physical workspace encompassing a substantial portion of the volume in the virtual reality system, much larger than the physical workspace of the haptic device on its own.

The haptic device communicates with the computer simulation to report its end effector position and receive force commands, just as if it were on a fixed base. The optical tracking system in the VR system provides the location of the base of the haptic device in the physical space to the simulation. The mobile robot's velocity (translational on the plane of the floor) is driven proportionally to the error between the haptic device's end effector position and a configured neutral position relative to the base, both within the coordinate system of the haptic device. This control system is based on the scheme described by [12].

Hardware

The system's hardware consists of three separate conceptual parts: a haptic device, a tracking target in the virtual reality system, and a mobile robot base. The experimental system described here uses a Haption Virtuose™ 6D35-45 haptic device, ART optical tracking, and an omni-directional mobile robot. The Virtuose provides substantial force capabilities with six actuated degrees of freedom (force and torque feedback) and a workspace roughly equivalent to a human arm pivoting at the shoulder. While this provides the desired experience and large workspace, the general system is not dependent on this specific device. For example, a Sensable Phantom Omni® device was mounted on the mobile robot early in the development process so software and controls integration could proceed concurrent with the work to develop the physical mount for the Virtuose device. Phantom Omni devices have three actuated (force feedback) and three passive (rotation sensing without torque feedback) degrees of freedom and a workspace roughly comparable to a human hand pivoting at the wrist. The conceptual integration and most of the software described in this work can be applied to many haptic devices, subject to the physical limitations of the Mecanum-style drive base used. The combination of the haptic device and the mobile robot needs to be tracked by the virtual reality system, so the method used varies according to the virtual reality system in which the mobile robot will operate. This experimental system was used within the Multimodal Experience Testbed and Laboratory ("METaL") immersive facility[1] at Iowa State University's Virtual Reality Applications Center. This $4\ \mathrm{m} \times 3\ \mathrm{m} \times 3\ \mathrm{m}$ CAVE™, with three projected surfaces (two walls and the floor), uses an ART TrackPack4 optical tracking system. The base of the Virtuose is tracked within METaL by attaching a "Claw" tracking target to it. This passive optical tracking target consists of four 20 mm diameter retro-reflective spheres rigidly arranged, which allows the four camera TrackPack4 system to determine the base's complete position and orientation within the room. The Virtuose is rigidly bolted to a stand holding it at a comfortable working height above the mobile robot. The entire setup in METaL can be seen in Fig. 1. One of the tracking cameras is visible in this photo above the corner between the screens, and the tracking target is partially visible just left of center, attached to the black base of the Virtuose. Some early testing of the software was done using a Phantom Omni instead of the Haption Virtuose and with another ART optical tracker as well as an InterSense IS900 hybrid ultrasonic-inertial tracking system.

The omni-directional mobile robot selected, pictured in Fig. 2, is based on the "4WD Mecanum Wheel Robot Kit," model 10011, from Nexus Robot[2]. This robot has four 100 mm diameter Mecanum wheels, which each have 9 rubber rollers at a

[1] http://www.vrac.iastate.edu/METaL/
[2] http://nexusrobot.com/product.php?id_product=67

FIGURE 1. HAPTION VIRTUOSE ON MOBILE ROBOT IN METaL VIRTUAL REALITY SYSTEM

45º angle to the axle. By controlling the wheel directions and velocities individually, the Mecanum-style drive can move by translation in the plane in any direction, as well as rotation in the plane with or without simultaneous translation. Each wheel is driven by a Faulhaber 2342L012CR coreless, brushed DC motor, running on nominally 12 VDC, with a no-load speed of 8100 RPM and a recommended top speed of 7000 RPM. Each motor is mated to a Nexus-built 12 CPM optical quadrature encoder, as well as to a 64:1 planetary gearhead which drives the wheel. Driving the wheels at the recommended top speed results in an overall theoretical rate of the robot of 0.57 m/s.

The robot kit also includes a custom Arduino™-compatible Atmel AVR ATmega328p-based microcontroller unit (MCU) development board and input/output expansion board. The Nexus development boards stray from the design of the Arduino Duemilanove board primarily by their integration of four L298-based motor driver channels. Each motor driver channel takes as input one pulse-width modulation (PWM) pin on the MCU, as well as an additional digital output pin to indicate direction/sign, and outputs $[-12, 12]$ VDC to the connected motor. Each motor's encoder provides an A and B phase output, which are connected to digital input pins on the MCU so that the embedded software may monitor the behavior of the motors. The hardware serial interface (UART) provided by the MCU is both directly accessible via TTL-level pin-outs as well as usable through an integrated USB-serial converter chip. Either UART access allows for communication between the MCU and a computer, and the USB interface also allows in-circuit programming using an Arduino-compatible bootloader and the `avrdude` programming software.

Software

Three separate software applications are involved in this integrated solution. Embedded software running on the mobile robot provides velocity control of the robot's four wheels, as well as a translation and rotation interface to a connected computer along with performance data. On a computer connected over a serial channel to the robot (either via USB or over Bluetooth RFCOMM), a "robot server" runs and adapts the serial communications with the robot into a network-transparent collection of VRPN analog and analog output servers. Finally, the SPARTA virtual assembly software, extended with adaptations to handle a device with a moving base and to output error measurements to the VRPN robot server, permits the use of the overall system as effectively one very-large-workspace haptic device capable of taking advantage of the full range of virtual assembly and interaction features.

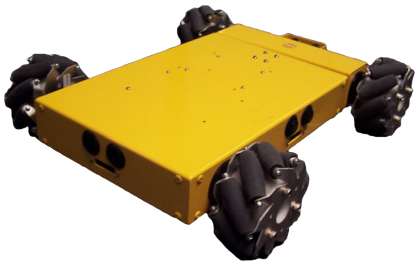

FIGURE 2. OMNI-DIRECTIONAL MOBILE ROBOT WITH MECANUM-STYLE DRIVE

The following sections describe and build on software layer diagrams shown in Figures 3 and 4. These diagrams share a common notation for distinguishing novel components from incremental improvements to existing software and stock software components used as-is. In both these figures, nodes represent software components or libraries, and edges show their dependencies in lower layers. Nodes with a solid border (colored blue) represent new software developed in this work, as an advancement of the state of the art. Nodes with a dashed border (colored green) are third-party libraries that were modified, typically to enable use in embedded processors, in this work and also represent advancements. Nodes with a dotted border (colored yellow) are stock third-party libraries used relatively as-is, and are shown to describe the base upon which this work builds.

Mobile Robot Software A custom embedded software stack was developed for the Arduino-compatible MCU in the mobile robot. Initially, the Arduino integrated development environment (IDE) was used for development and testing, later being mostly supplanted by the arduino-mk Makefile system. The software was written in C++ using a toolchain consisting of `avr-gcc` 4.7, `avr-libc` 1.8.0, and `binutils-avr` 2.20.1. In the interest of developing a modular, maintainable codebase [13], the MCU software consists of a small main application calling into a number of libraries. Figure 3 is a layer diagram showing the libraries used and their dependencies, with the ap-

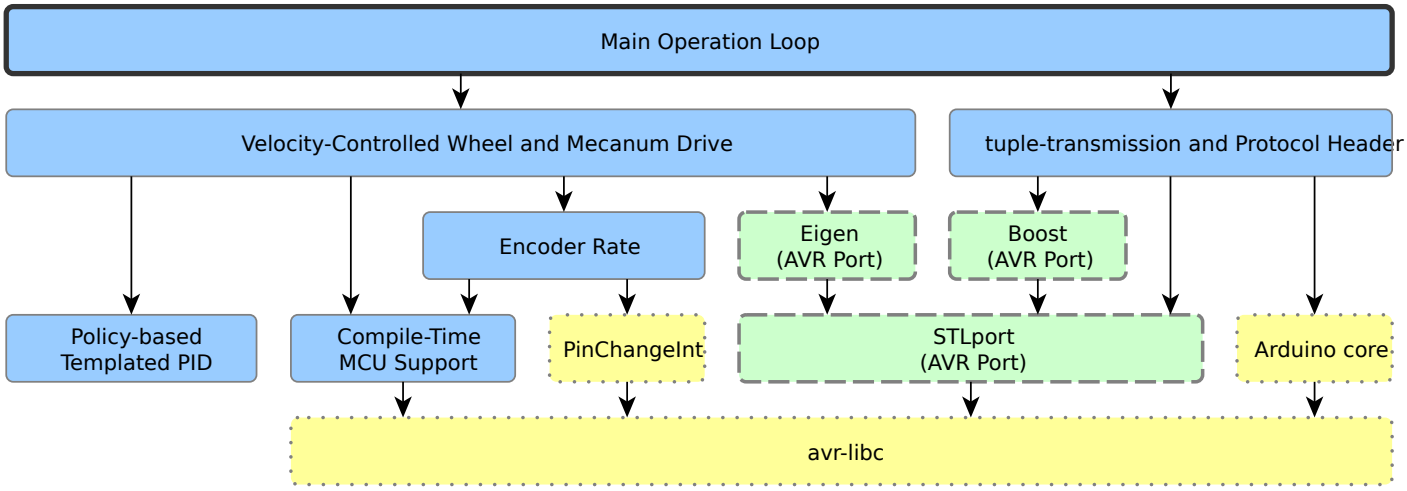

FIGURE 3. EMBEDDED SOFTWARE LAYER DIAGRAM

pearance of each node in the diagram holding the significance discussed earlier. In particular, in this diagram, the top node, with the thick border, represents the main loop of the embedded software.

A discussion of these components in more detail is warranted to describe the role they play and the contributions involved. A subset of the Arduino 1.0.2 core library, modified for GCC 4.7 compatibility, was used for some basic functionality. Registration and dispatch of pin-change interrupt handlers, used to monitor the encoders, was performed using the PinChangeInt 1.7[3] Arduino library developed by Lex Talionis and Michael Schwager. Three open-source third-party libraries were modified to support use on AVR microcontrollers during the course of this work. The STLport C++ standard library implementation was ported to AVR, and the modified version is publicly available[4] with instructions for use in the Arduino environment. Some AVR-specific porting and Arduino convenience modifications were also made to the header-only portions of the Boost C++ libraries version 1.51.0, released[5], and used in the embedded software. Additionally, the Eigen C++ template library for vector and matrix math was modified to avoid name collisions with `avr-libc` and remove assumptions about type sizes that do not hold when compiling for a 16-bit address space, and made available along with corresponding Arduino convenience headers[6]. These libraries form the base upon which the higher-level functionality for robot control was built.

In the interest of modularity, software written from scratch for use in this project was also divided up into logical libraries. The "tuple-transmission" library uses typelists and C++ template meta-programming techniques to generate efficient code for serializing and de-serializing finite, known collections of messages. Template meta-programming allows substantial portions of the code to be specialized and optimized automatically at compile time, rather than invoked with branches incurring performance costs at runtime. The tuple-transmission library, by design, is used on both the MCU and the computer to permit two-way communication. The full protocol used is defined in a single header file, designed for use in identical form on both ends of communication, allowing more efficient code and minimal overhead by not requiring message contents to be self-describing. The tuple-transmission library has been open sourced under the Boost Software License and published online[7], and the protocol header used is also publicly available[8].

Some additional libraries were written to encapsulate the details of elements of the system. A custom C++ microcontroller support library provides, among other features, a virtual "signed analog output" port for motor control, where a signed output value is turned into a PWM duty cycle and a direction bit on a pair of physical output pins. A template-based, layered proportional-integral-derivative (PID) controller library is used for velocity PI control of each individual wheel. A library for reporting rates based on quadrature encoder input, built using policy-based design principles [14], handles pin-change interrupts for all four quadrature encoders and computes instantaneous motor rotational velocity. This library successfully handles the $7000\,RPM \times 12\,CPR \times 4$ transitions per click $\times$ 4 wheels $= 1344000$ interrupts per minute, or 22400 interrupts

[3] `http://arduino-pinchangeint.googlecode.com`
[4] `https://github.com/vancegroup/stlport-avr`
[5] `https://github.com/vancegroup/arduino-boost`
[6] `https://github.com/vancegroup/EigenArduino`
[7] `https://github.com/vancegroup/tuple-transmission`
[8] `https://github.com/vancegroup/NexusRobotProtocol`

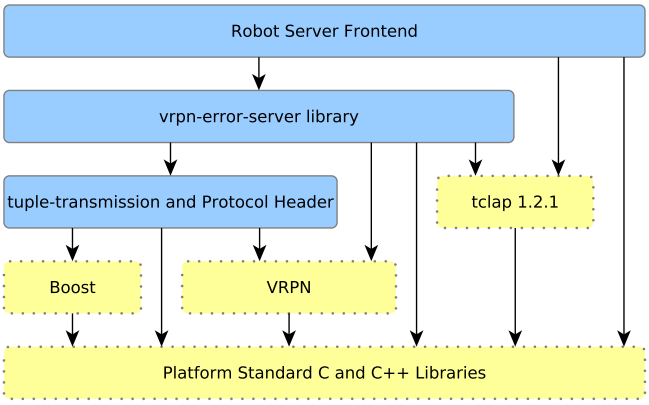

FIGURE 4. ROBOT SERVER SOFTWARE LAYER DIGRAM

per second, generated during maximum recommended velocity robot motion. Finally, a library encapsulating velocity-controlled individual wheel drive and, from that, four-wheel Mecanum drive, provides the highest level interface to mobile robot motion to the embedded software. This permits not only the "production firmware," but also a number of verification and testing applications, to be built with concise, expressive high-level code.

Robot Server Software The robot server is the simulation computer's counterpart to the embedded software. It communicates using the same tuple-transmission library and protocol headers as the embedded software, and serves to interface the established serial protocol of the robot with network-transparent VRPN (Virtual Reality Peripheral Network) [15] server devices. Figure 4 shows the layers of software libraries involved in the robot server, using the same conventions as Fig. 3 to indicate component origin. Two library components appear here that did not appear earlier. The first, TCLAP, is an open-source command line option parsing/handling library[9]. Internally, the server application is actually a front-end to an internal "vrpn-error-server" library providing a common subset of robot server features. The front-end application used with the haptic device system provides control of the robot by a `vrpn_Analog_Output` server set up to receive two-dimensional, floating-point error vectors, scale them by a command-line-configurable proportional gain, then send them to the mobile robot as signed integer velocity commands. As with the embedded software, the creation of a library layer with a thin application layer over it permits additional applications for testing and verification to be easily built. The entire package, which has been built and tested on both Linux and Windows, is available as open-source software[10].

[9] `http://tclap.sourceforge.net/`
[10] `https://github.com/vancegroup/vrpn-error-server`

SPARTA Simulation Software The final piece of the integration puzzle is the actual virtual assembly simulation software. In this work, SPARTA, the *Scriptable Platform for Advanced Research and Teaching in Assembly* [16], was extended with two modules to enable the use of the haptic device on the mobile robot. SPARTA, as the successor to the SHARP family of applications [17], provides a virtual environment for physically-based, haptic interaction with computer-aided design (CAD) models. Its physics simulation and range of haptic device drivers are written in object-oriented C++, while the graphics, audio feedback, and high-level interactions are written in Lua, taking advantage of the VR JuggLua framework [18] based on OpenSceneGraph and VR Juggler. Haptic device configuration, as well as virtual assembly scenario creation, is done with a domain-specific language built within Lua.

A particularly relevant aspect of SPARTA's design is the generic interaction device interface defined and used by the simulation. As a first step, this permits different types of input devices to present a uniform API, including haptic devices from different vendors, devices without haptic feedback, and so on. Adding a device driver to SPARTA by implementing this interface for a given device makes it immediately usable with all applicable functionality of the software: the design of the simulation is very loosely coupled to particular interaction devices. Furthermore, the generic device interface can also be implemented by "virtual" or "filter" devices: software objects that behave like input devices, but do not correspond directly to physical hardware, instead wrapping another input device and observing or modifying the data flow. It follows that supporting interaction with a haptic device on a mobile robot in SPARTA can be reduced to including a driver for the haptic device and producing one or more filter devices to interact with the mobile robot and account for its effects on the data reported by the haptic device. SPARTA already included support for the Haption Virtuose, as well as the Phantom Omni used for early testing, before this research began. The remaining additions to SPARTA are neatly split into two filter devices. The first, called `TrackedTransform`, is not strictly limited to use with a powered mobile robot base. Its function is to appropriately transform data both going to and coming from the simulation, based on the position of a tracker target assumed to be fixed to the base of its contained device.

The remaining task is to control the movement of the mobile robot based on data from the haptic device. In this research, based on the control scheme investigated by Garlington [12], the error (vector) between the position of the haptic device's end effector and a pre-defined neutral position (roughly centered in the physical workspace of the haptic device) drives the robot. The control seeks to minimize that error and move the base so the end effector is neutrally located at all times. The SPARTA filter device for driving the robot is configured by providing the contained device and its corresponding neutral end effector position, as well as the device name of a `vrpn_Analog_Output` de-

vice that should receive the error vector. `VRPNMobileBase` does not modify any of the data flows, but on each position update it computes an updated error, connects to the robot server (which may be running on the same computer), which applies a gain to the error and passes it on to the mobile robot embedded software as a goal velocity.

RESULTS

The full system as described successfully provides haptic interaction within a large workspace by having a mobile robot drive a ground-based haptic device around within the tracked area. Figure 5 shows the system in use, with the same hardware as pictured in Fig. 1. Before this image was captured, the user grasped the white pin (currently seen on the right side of the image) with the purple cursor (representing the end effector location in the simulation) and removed it from the rest of the assembly (green object and blue object). In the photographed moment, the user is now physically walking across the space while grasping the pin in order to set it down on the other side of the room, and the mobile robot is moving the base of the haptic device to keep up and allow this single movement to span a wider area than could be reached with the haptic device alone. Haptic feedback of colliding and sliding parts was felt during the removal of the pin, and now the simulated mass of the pin is all that is felt during movement of the haptic device.

The modularity and loose coupling described in the preceding sections produced a highly robust system: any one or more of the SPARTA simulation, robot server, VR system tracker, or mobile robot motor power can be shut down and started up again without bringing down any other part of the system. The ability to restart the robot server application was particularly useful during tuning of the proportional gain used to compute velocity from end effector positional error, since the gain was specified as a command-line argument to the robot server. Stopping it and starting it again with a different value while leaving the rest of the system operational and ready to use supported a very short test cycle time.

Following implementation of all components, the gain was found interactively by increasing the gain incrementally until the experience of moving a grasped object in space began to produce undesired haptic artifacts. A very high gain was able to be used, resulting in the mobile robot quickly "following" the movement of the end effector and keeping up with movement throughout the space. Free space movement in the entire area is possible and the user feeling is subjectively light and transparent. With the Haption Virtuose 6D35-45 device, capable of 35 N peak force and 10 N continuous, mounted on the mobile robot, simulated collision forces are stiff and crisp, and subjectively comparable to the experience of using the device on a fixed base. As intended, the full tracked area in the virtual reality system was made usable for haptic interaction with this work, allowing direct interaction with a larger, fuller simulation than with the ground-based haptic device alone.

FIGURE 5. SYSTEM IN USE SIMULATING DISASSEMBLY IN METaL VIRTUAL REALITY SYSTEM WITH HAPTION VIRTUOSE

CONCLUSIONS AND FUTURE WORK

The full system is operational and suitable for further use, analysis, and development. Because the design of the SPARTA software allowed the necessary parts of the solution to be interposed essentially invisibly between the haptic device hardware and the simulation, the "haptic-device-on-powered-mobile-robot-base" can be used with all the scenarios and interactions built on SPARTA already; Fig. 5 is in fact one such example.

There are many opportunities for future work. One area of improvement involves implementing a method to allow the user to operate in a virtual world that is larger than the physical workspace of the optical tracking system. We are currently exploring various methods of navigating within a large virtual environment in which the user has haptic capabilities. Navigation in this sense refers to interacting with environments larger than the physically tracked area in a given VR system by moving the physical room around in the virtual space. Combining navigation and this system would allow, for example, factory or large assembly line walk-throughs, while still retaining the ability to interact haptically in the full physical workspace due to the mobile base.

Another area of future work involves exploring how to limit the mobile robot to avoid collisions in the physical space. In a virtual reality system of any sort, there are always physical limits, whether they are projection-screen walls or walls of an area with wide-area tracking and head-mounted displays. Physical object avoidance by the mobile robot needs to be implemented. Possibilities include use of on-board sensors on the mobile robot to detect limits and modify the velocity commands received accordingly, as well as computer-side approaches using the tracked position of the base to modify velocity commands before they are

sent. There are presently no specific safeties, other than operator interaction, integrated into this system to prevent the mobile robot from contacting the projection screens or the walls in the space.

There also exist conditions in the projection-screen environment where the haptic device occludes the user's view of the virtual environments. This is only an issue in projection-screen based systems, as full head mounted display systems do not project the real environment onto the user's view. One approach would be to place the robot behind the user but this would create other issues, such as needing a larger floor area than the projected floor area. We will continue to explore the possibility of this configuration.

Cable management for the haptic device could also be improved. Presently no explicit cable management beyond bundling/looming and the inherent stiffness of the cables is being performed; however, the stiffness of the cable effectively kept it away from the mobile robot wheels. Future work will include exploring cable management techniques.

Additional work is needed on the control scheme to counteract reaction torques at the interface between the floor and the mobile robot. In the current setup, the system assumes that the mobile robot begins and moves aligned with the axes established for the room. This is generally true, however, as noticed particularly during extended testing, a powerful or extended simulated collision with the haptic device's arm extended produces a torque about the axis between the mobile robot and the haptic device's base. This can result in some rotational slippage of the mobile base so that it is no longer aligned with the room axes, which violates assumptions presently made at some layers of the system. Work on a rotational control scheme to keep the robot aligned with the axes could improve the experience. Applying rotational control to the mobile robot based on the rotation of the end effector, the behavior of the user, or both, could improve usability by keeping the haptic device base both appropriately located and conveniently oriented. This would also require appropriately handling rotation of the base at all layers in the combined configuration, revisiting assumptions.

ACKNOWLEDGEMENT

The authors would like to thank Mylene Campana (ENS Cachan Antenne de Bretagne, France) and Adam Carlson for their work in developing familiarity with the Mecanum-style drive, and to Luke Swanson and Maria Blanek for the physical mount design and load testing. This research was performed at the Virtual Reality Applications Center at Iowa State University. This material is based upon work supported by the National Science Foundation under grant #CMMI-1061458.

REFERENCES

[1] Hirata, Y., and Sato, M., 1992. "3-dimensional Interface Device For Virtual Work Space". In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Vol. 2, IEEE, pp. 889–896.

[2] Sato, M., 2001. "Development of String-based Force Display: SPIDAR". In 8th International Conference on Virtual Systems and MultiMedia.

[3] Haption S.A. Haption Inca 6D datasheet.

[4] Gosselin, F., Andriot, C., Savall, J., and Martín, J., 2008. "Large Workspace Haptic Devices for Human-Scale Interaction: A Survey". In EuroHaptics 2008, M. Ferre, ed., Vol. 5024 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, pp. 523–528.

[5] Gosselin, F., Andriot, C., Bergez, F., and Merlhiot, X., 2007. "Widening 6-DOF haptic devices workspace with an additional degree of freedom". In Second Joint EuroHaptics Conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (WHC'07), IEEE, pp. 452–457.

[6] Borro, D., Savall, J., Amundarain, A., Gil, J. J., Garcia-Alonso, A., and Matey, L., 2004. "A large haptic device for aircraft engine maintainability". *IEEE Computer Graphics and Applications,* ***24***(6), Nov., pp. 70–74.

[7] Haption S.A. Scale 1 Haption Motorized Boom datasheet.

[8] Ueberle, M., Mock, N., and Buss, M., 2004. "VISHARD10, a novel hyper-redundant haptic interface". In 12th International Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2004. HAPTICS '04. Proceedings., Ieee, pp. 58–65.

[9] Nitzsche, N., Hanebeck, U. D., and Schmidt, G., 2003. "Design issues of mobile haptic interfaces". *Journal of Robotic Systems,* ***20***(9), Sept., pp. 549–556.

[10] Peer, A., Komoguchi, Y., and Buss, M., 2007. "Towards a mobile haptic interface for bimanual manipulations". In 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, pp. 384–391.

[11] Barbagli, F., Formaglio, A., Franzini, M., Giannitrapani, A., and Prattichizzo, D., 2006. "An Experimental Study of the Limitations of Mobile Haptic Interfaces". In *Experimental Robotics IX*, M. H. Ang Jr. and O. Khatib, eds., Vol. 21 of *Springer Tracts in Advanced Robotics*. Springer Berlin Heidelberg, pp. 533–542.

[12] Garlington, I., 2012. "Expanding the usable workspace of a haptic device by placing it on a moving base.". Masters thesis, Iowa State University.

[13] Parnas, D. L., Clements, P., and Weiss, D. M., 1985. "The Modular Structure of Complex Systems". *IEEE Transactions on Software Engineering,* ***SE-11***(3), Mar., pp. 259–266.

[14] Alexandrescu, A., 2001. *Modern C++ Design: Generic Programming and Design Patterns Applied*. Addison-

Wesley Longman Publishing Co., Inc., Boston, MA, USA.
[15] Taylor II, R. M., Hudson, T. C., Seeger, A., Weber, H., Juliano, J., and Helser, A. T., 2001. "VRPN: a device-independent, network-transparent VR peripheral system". In Proceedings of the ACM symposium on Virtual reality software and technology - VRST '01, ACM Press, p. 55.
[16] Pavlik, R. A., and Vance, J. M., 2011. "Expanding Haptic Workspace for Coupled-Object Manipulation". In ASME 2011 World Conference on Innovative Virtual Reality (WINVR 2011), ASME, pp. 293–299.
[17] Seth, A., Su, H.-J., and Vance, J. M., 2008. "Development of a Dual-Handed Haptic Assembly System: SHARP". *Journal of Computing and Information Science in Engineering,* ***8***(4), p. 044502.
[18] Pavlik, R. A., and Vance, J. M., 2011. "VR JuggLua: A Framework for VR Applications Combining Lua, OpenSceneGraph, and VR Juggler". In Workshop on Software Engineering and Architectures for Realtime Interactive Systems (SEARIS) in IEEE Virtual Reality, IEEE.