Teleoperating a Humanoid Robot with Virtual Reality

Jordan Allspaw*
University of Massachusetts Lowell
Lowell, MA, USA
Jordan_Allspaw@uml.edu

Jonathan Roche
University of Massachusetts Lowell
Lowell, MA, USA
Jonathan_Roche@student.uml.edu

Adam Norton
University of Massachusetts Lowell
Lowell, MA, USA
Adam_Norton@uml.edu

Holly Yanco
University of Massachusetts Lowell
Lowell, MA, USA
Holly_Yanco@uml.edu

ABSTRACT

Teleoperating robots, particularly humanoid robots, from a remote environment remains a difficult problem. Significant research has been conducted in human-robot interaction for teleoperating mobile ground robots for both novice and experienced operators with a variety of interfaces and strategies; however, there is significantly less research on teleoperating humanoid robots. By their nature, humanoid robots tend to operate in very 3D environments, with variable terrain. With humanoid robots, much of the focus is on performing the tasks that a human would normally do, in the way a human would normally perform it, in the environment that the human would normally perform it in. This focus greatly increases task complexity and the required effort. Current operator control interfaces for humanoid robots often require very experienced operators and significant amounts of time for planning. A large amount of the planning and cognitive load is attributable to the operator attempting to gain adequate 3D situation and task awareness while viewing a 2D screen. This is one area where virtual reality (VR) has enormous potential to provide benefits to allow the operator to quickly and accurately understand the state of a robot in a scanned 3D environment, and to issue accurate commands with less cognitive load. In this paper, we will describe how we developed and created a VR interface to remotely control a humanoid robot and explain our methodology for the types of visualization and control we present.

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; • **Computer systems organization** → **Robotic control**; *External interfaces for robotics*;

KEYWORDS

Human-robot interaction (HRI), virtual reality (VR), teleoperation of humanoid robots

*This is the corresponding author

Virtual, Augmented, and Mixed Reality for HRI, March 5, 2018, Chicago, IL, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Jordan Allspaw, Jonathan Roche, Adam Norton, and Holly Yanco. 2018. Teleoperating a Humanoid Robot with Virtual Reality. In *Proceedings of HRI-18 Workshop on Virtual, Augmented, and Mixed Reality for Human-Robot Interaction (Virtual, Augmented, and Mixed Reality for HRI).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In October 2012, the DARPA Robotics Challenge (DRC) was announced, a competition where teams would compete with the overall goal of developing semi-autonomous humanoid robots that could perform "complex tasks in dangerous, degraded, human-engineered environments" [4]. The competition consisted of three rounds: the Virtual Robotics Challenge (VRC), the DRC Trials , and the DRC Finals. During the Trials and Finals, disaster response tasks, such as driving and exiting a modified Polaris, opening and walking through a door, operating a drill, and walking across debris, were completed using physical, typically humanoid robots. During the Trials, tasks were performed in individual runs, whereas the Finals had all tasks performed in succession in a combined run. The most successful teams from the Trials continued on to the Finals.

An analysis of the human-robot interaction (HRI) techniques used at the DRC Trials [18] and the DRC Finals [10] was performed. Both studies examined approaches taken by the teams for interaction design, such as the number of operators, types of control, how information was conveyed to the operator(s), and varying approaches to semi-autonomous task execution. There were a number of strategies employed by the various teams, but they predominately relied on a combination of mouse, keyboard, and gamepad interfaces. For the visualization of sensor data in the DRC Finals, the teams in the study averaged 5.1 active screens, 6.2 camera views, and 2.5 different point cloud visualizations. There was an average of 1.6 active operators (using an input device for robot control) and 2.8 passive operators (watching over the shoulders of active operators, offering strategic advice). All of this represents a large amount of manpower with the goal of allowing the operator to gain proper situation and task awareness of the remote environment by interpreting sensor data from a 2D interface (the screen) and building a 3D mental model.

Another interesting and common trait was that teams frequently had different visualization and control paradigms depending on the task they were performing. For example, one team went so far as to intentionally lock the interface of the operator depending on the task selected, preventing them from switching to different

visualization or control modes [11]. This strategy is also evident from the DRC HRI analysis; most teams would adjust what sensor data was being displayed, what types of control to give the operator, and changing whether a gamepad would be used or keyboard and mouse. All of this seems to imply that each team spent a significant amount of effort creating a specialized interface for each task, in order to provide enough situation awareness to allow the operator(s) to complete the task within the allotted time.

The only known uses of VR in the competition were teams utilizing the Oculus Rift Developer Kit (DK) [2] for viewing point clouds, but this was used as a situation awareness complement to viewing point clouds through traditional means. No teams used VR as their primary interface for either visualization or control. A discussion with a member of one of the teams revealed that they had investigated using the Oculus Rift DK, but given its limitations found it to be of limited use and did not end up using it at the Finals.

The Oculus Rift DK required the wearer to be in a fixed sitting or standing position, ideally keeping their head in a set position. While the headset could handle rotation in the pitch and yaw axis, it could not handle lateral movements. If the operator moved their head significantly, the virtual world would be unable to adapt and the operator would frequently get motion sickness [5]. Even when using it in the ideal position, low pixel density and low refresh rate made it difficult for VR to reach its potential. Shortly after the DRC Finals concluded in June 2015, several powerful consumer virtual reality headsets were released, including the consumer ready Oculus Rift and the HTC Vive [1]. Both of these headsets were significant improvements over the Oculus Rift DK, and while we will focus on the HTC Vive headset, they are largely interchangeable.

The newer headsets remove the fixed position limitation and have built in accurate position tracking which allows what is called "room scale" or the ability for VR experiences which allows users to freely walk around a play area, with their real-life motion reflected in the VR environment [3]. Coupled with other improvements such as increased screen resolution and increased refresh rate, we believe that another look at using VR for visualizing and controlling humanoid robots is warranted, and have designed and created a humanoid robot VR interface to be used in examining HRI.

2 SYSTEM

While fully autonomous robots are popular in research and have had success in very specific domains, for many complex tasks having a human operator in the loop is still often preferred. For the DRC specifically, many of the teams utilized very little autonomy. The teams that did have some autonomy typically used it in very specific and limited cases. For example, IHMC started with scripts that could perform tasks on their own and over time broke up those scripts into smaller pieces. The end result was a series of steps where the robot would plan a smaller action, and the operator would either confirm the plan or, more often, make adjustments to the robot's state or plan before approving [7]. After the DRC Finals, many teams published papers about their robot control interfaces and lessons learned including Team IHMC which allowed their robot operator to "command the robot via interaction with the three-dimensional (3D) graphic tool rather than specifying robot motion or joint positions directly" [7]. With all of that in mind, we are primarily interested in an interface where the operator retains control of most of the robot functions. Similar strategies were common among several of the teams, albeit all on portrayed on 2D screen interfaces, and usually interacted with by a mouse and keyboard.

While the teams competing in the DRC had a specific goal, namely completing a series of structured tasks in a controlled competition, it still represents one of the most significant events in HRI for humanoid robots, and so we built off of the lessons learned when building our own VR interface. For instance, teams with greater success at performing tasks relied on a fused display of the 3D robot avatar, point cloud, and camera images, in a common reference frame [10]. However, 2D interaction methods, like a mouse and keyboard, were used to maneuver the displays and issue commands. HRI for humanoid robots consists inherently of 3D data, for which VR offers a unified solution for both 3D display and control. Our goal was to create an interface that would leverage VR in order to increase operator situation and task awareness when visualizing a remote location, while also providing adequate control methods to compete with traditional 2D screen with mouse and keyboard interfaces.

Our proposed interface utilizes the HTC Vive [6], a commercially available VR headset combined optionally with the Manus VR [8] gloves, a pair of gloves that allows accurate finger tracking, as well as tracking the wrists' location with the HTC Vive Trackers [6]. As an alternative to the glove, we also use the standard controllers that come with the headset.

2.1 Robot platform

While our interface has been designed to be applicable to any humanoid robot with similar features, we developed and tested it using the Valkyrie R5 created by NASA [14]. Valkyrie is a 6 foot tall, 300 pound humanoid robot with a pair of four fingered hands attached to 7DOF arms. The robot has several sensors built in including a Carnegie Robotics Multisense [15], located in its head, capable of generating both high and low density point clouds, depending on bandwidth requirements, as well as a stereo camera view. The robot also has an additional pair of stereo cameras in the lower torso. Finally in addition to accurate joint state and torque tracking on each joint the robot has embedded tactile sensors in the fingers and palm to detect grasping success. The types of actions the robot perform are to move a foot to a location in 3D space and hold without taking a step; to move a foot to a location in 3D space and take a step; to command the torso, pelvis, or neck to a commanded position; and to command one of the arms to a location in 3D space.

2.2 Software Platform

Valkyrie's sensors and controller interface communicate using ROS [13], a communication middle-ware commonly used in robotics. When this project was started, the Linux driver for the HTC Vive was lacking many important features so we built our application within the Windows operating system. In order to communicate with the Robot we utilized a Unity [17] plugin of ROS.NET [16] which handled all of the communication and conversion between standard ROS topics and types into things that could be uses within Unity. With the infrastructure settled, we were able to use the most

supported and feature complete SDK for the HTC Vive, with the added bonus of being able to leverage several other features within Unity.

3 INTERFACE

We have built our VR interface to attempt to take full advantage of the available features. With the operator wearing the VR headset and wearing either the gloves or controllers, the operator finds themselves inside a virtual world. The operator is able to physically move around a clear work space in order to move an avatar in the virtual world. Since the physical work space can be significantly smaller than the remote robot work space, we have also allowed the operator to teleport around the virtual world by pointing their controller at a location on the ground and pressing a trigger button. This allows the operator to continue to navigate in remote environments that are much larger than the operator's own environment.

The robot is capable of accurately tracking its own relative movement in the world via an IMU, as well as the joint state of each of its limbs. This means that with a known robot model, we are able to render a virtual version of the robot for the operator that updates in real time as seen in figure 1. The operator is then able to navigate around using room scale to view the virtual robot from any angle and position.

3.1 Visualization

Sensors capable of scanning the nearby environment and generating 3D point clouds have seen increased use on both research and commercial robot platforms. Traditionally the operator can view the point clouds by navigating a virtual camera throughout the point cloud to different viewpoints, in order to build up their own internal 3D understanding of the environment layout, a process called moving parallax. Once the operator has a solid mental model of the remote environment, they can then plan out the actions the robot should perform. This process can be time consuming, mentally taxing, and error prone; however, the ability to visualize point clouds as virtual environments is one of the most powerful features VR can bring to teleoperating robots. Rather than attempt to rebuild a 3D mental model of the environment from 2D vantage points, the operator can view and virtually navigate the 3D environment as is, at proper life scale. The environment is also presented to the operator in stereo vision by the VR headset, allowing for enhanced innate understanding, compared to using a 2D display. By rendering the point cloud data into the virtual world, the operator can understand the state of the remote world in a way that requires significantly less time and effort.

3.2 Rendering

As discussed in Davis et al. [5], cybersickness remains a concern. Whereas in many traditional interfaces, the frame rate of the system can be considered insignificant, for VR it is recommended to maintain a frame rate above 90 frames per second. One method for reducing system load is to throttle the sensor data which can work for static environments; however, for dynamic environments the operator needs up-to-date information in real time. To that end, particular care was taken when handling point clouds to make sure that receiving and rendering the point cloud in VR was both fast and efficient. Figure 2 gives an example of rendering a high density point cloud of a cluttered table with minimal impact to system performance.

VR poses several unique challenges when rendering 3D data. Unlike a 2D interface, the computer can not simply pause rendering to load new sensor data to the graphics card. One must worry about both average frame rate that may be slowed down simply by the quantity of data in the scene, and the maximum latency between frames, which is caused by loading new sensor data.

In an effort to keep the total data as low as possible, the decision was made to render each point in a point cloud as octahedrons, which only have 8 triangles and 6 vertices, compared to a cube which has 12 triangles and 8 vertices. Voxel filtering and surface reconstruction are also highly effective methods of reducing the total number of points that have to be rendered.

The problem of reducing the maximum frame latency is equally challenging, if not more so. When a new point cloud is received by Unity, it must be broken up into vertex, triangle and color arrays that unity mesh objects can understand. This can be handled on a thread other than the main rendering thread, but loading those arrays to a Unity mesh object to be visualized must pause rendering while they are loaded. As long as the user maintains an average frame rate of 90 frames per second, a brief drop to 50 or 60 for a single frame is generally not a problem, but that still only allows for a maximum of 20 milliseconds to load new point cloud data to the mesh object and then to the graphics card. To overcome this limitation, the point cloud may be broken down into several different meshes, such that each may be loaded in under 20 milliseconds. These meshes are split into different color groupings, all of which get updated in the same frame. This color grouping prevents having discreet chunks of the point cloud update in different frames. Special care is also taken, if there are several different point clouds from different sensors in the scene, to avoid updating them in the same frame to prevent a drastic increase in frame latency.

3.3 Usability

Of particular use is the ability for the operator to view the point cloud in relation to the robot as seen in figure 3. Notice the low density point cloud of a doorway in front of the robot. Because the scene is recreated in a one-to-one scale, both for the robot and the operator's virtual avatar, the operator can see both the position and size of the door relative to the robot and themselves, then maneuver their viewpoint throughout the virtual space as necessary. This should allow operators to plan their actions quickly and reliably. This design follows recommendations from Nielsen, Goodrich, and Ricks [9] and Okura et al. [12], where it is suggested to use a common reference frame, with variable perspectives, for 3D displays of the environment and the robot avatar to provide greater surroundings awareness to the operator.

This VR system also has the capability to load static models or even static point cloud scans of an environment if available. By adding this additional information, the operator can more easily localize themself and understand the robot's position and environment. Static models and point clouds help to fill in information about what the robot may not currently be looking at.

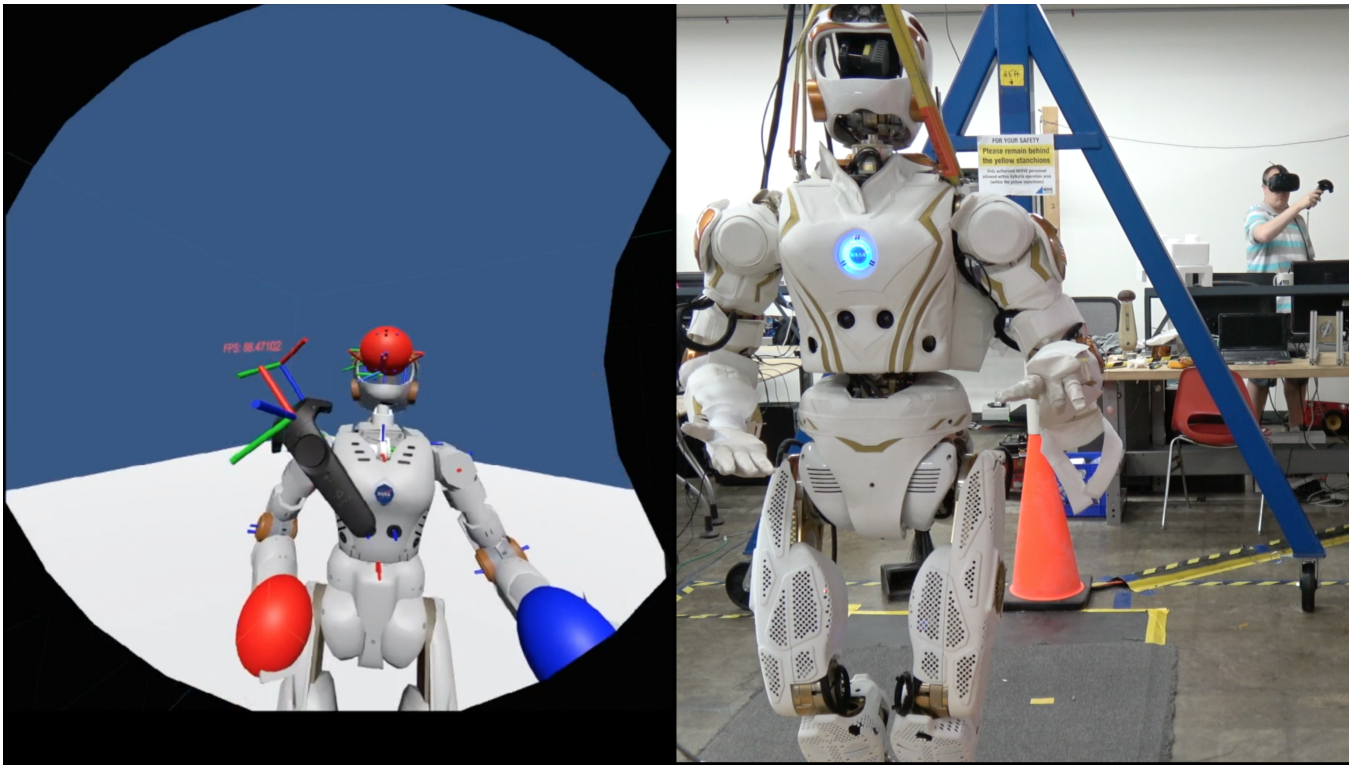

Figure 1: Screenshot of an operator controlling a robot in VR. On the left is the virtual view that the operator sees; on the right is the real robot with the operator standing behind it.

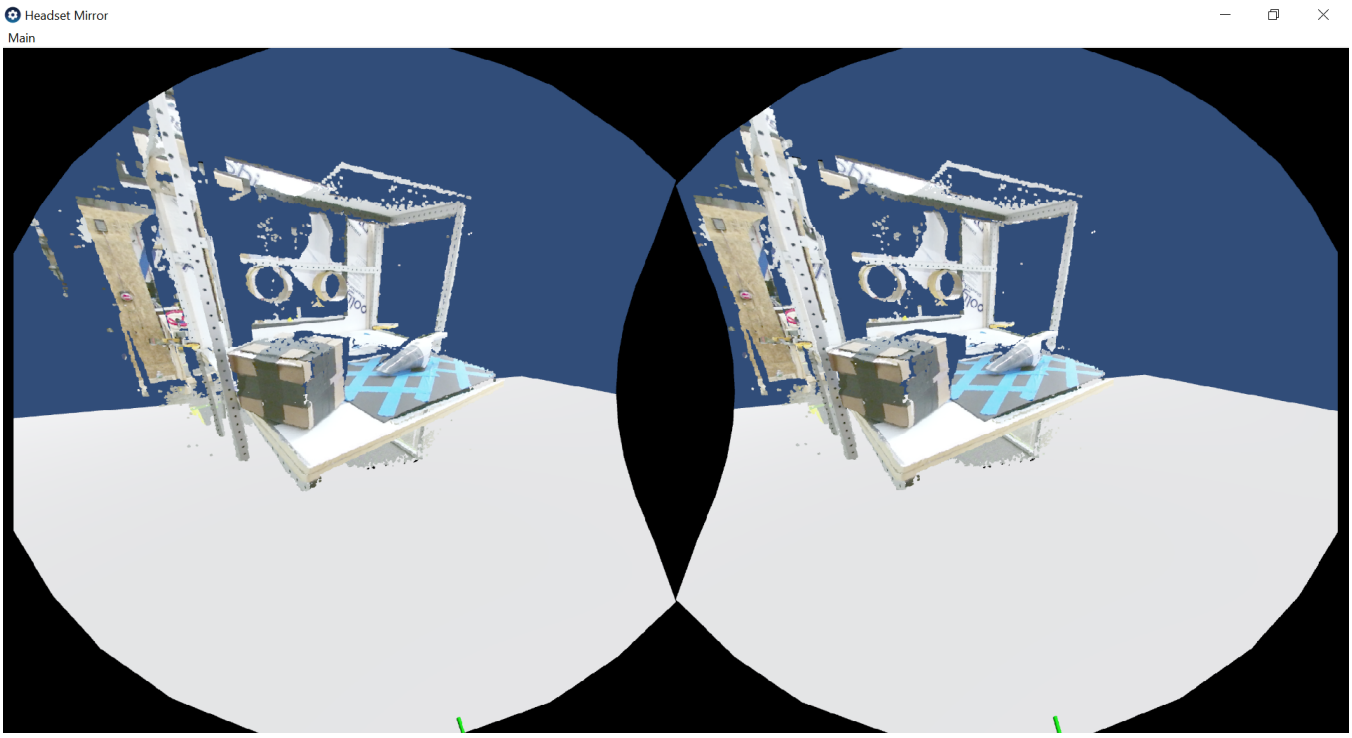

Figure 2: Visualization of the high density point cloud reconstruction.

Figure 3: Virtual world with sparse point cloud visualization, robot facing and walking towards a mock doorway

While point clouds transition naturally to a 3D environment, taking advantage of cameras in VR poses an interesting challenge. The camera view itself is 2D and so gains no immediate benefit by being in a 3D space; however, we were able to provide ease of use when viewing the camera streams. By creating rectangular "windows" to where the cameras are rendered in the virtual world, as seen in figure 4. The operator is able to "grab" the virtual windows with their controller. Once interacting with the virtual item, they can move it to a different location or resize the window. This allows the operator to position the windows in strategic locations: for example, placing the camera windows roughly where the actual camera field of view would align. Alternatively, the operator can choose to pin a camera window to hover at a specific position from the operator, serving as a heads up display. As the operator moves throughout the virtual environment, they can see camera stream whenever they wish. These options allow the operator to retain the same functionality as the 2D display, at a minimum, while providing the possibility of more context and convenience. By allowing both fixed perspectives (camera views) and variable perspectives (point clouds, robot avatar), we follow the design guideline for sensor fusion techniques demonstrated to be successful at the DRC Finals [10].

3.4 Control

One of the difficulties faced by DRC teams trying to use the Oculus Rift DK1 was that there was no built in input device. While viewing the sensor data in 3D was seen as possibly useful, the issue was that the other interface functions, such as commanding the robot, were still on the 2D interface. In order for the VR interface to be effective, it needs to have both the visualization and control of the robot to be within VR. We also wanted to translate the operator's movement of their arms, hands, and fingers to those on the robot, such that the correlation between operator movement and intended robot movement would be more apparent and direct. This translation involves some automation from the robot, as the operator is not controlling individual joints, but rather defining higher-level trajectories (e.g., operator moves their hand to a point in space and the robot moves its hand and arm on its own to reach the same relative location).

The two primary types of controllers the operator can use to interact with the virtual world are either the Manus VR glove or the HTC Vive controllers. Both have 3D tracking within the virtual world and allow the operator to interact with the virtual objects in order to manipulate them, such as the camera windows. The default controllers, with more buttons, allow for more options for activating and deactivating specific features. The primary advantage of the Manus glove is the ability to directly control the fingers. With the finger tracking enabled, as the operator opens or closes their respective fingers the robot will mimic, as seen in figure 5. This allows for much finer grasping than a simple "open" and "close" configuration, while keeping the speed of use much faster than manually controlling each finger by keyboard.

For controlling the arms, there are two ways the operator can send commands to the robot. The first is by pressing down a "trigger' button, then the robot will move its respective hand to that controller's location. You can hold down the trigger button, while moving your own hands, which will cause the robot to continuously match your movements with its own. While useful for quick movements where prevision is not a priority, relying entirely on an inverse kinematic solution for movement can lead to undesirable movements. The second way is by grasping the virtual limb and moving it into the configuration the operator wants, on a joint by joint basis. After confirming, the real robot will then move to the correct configuration. These two methods provide a balance between speed and precision, and were both methods used in the DRC.

The current method for commanding the robot to walk is a fairly straightforward joystick command. By pressing up on the joystick built in to the controller, the robot will continuously walk forward, with footstep length correlating to the magnitude of the joystick. While a simple design, the operator is able to view the robots planned footprints in the 3D environment. Alternatively, the operator can command the robot to walk by pointing at a location in the world and pressing a trigger, then the robot will plan out the footsteps to the location. A third proposed solution is to allow the operator to place virtual objects as planned footprints, then the robot would attempt to take those exact steps. As with manipulating the arms, the combination of these three methods should allow the operator to choose between speed and precision based on the task needs.

Currently the primary method for controlling the head involves the operator "grabbing" the head, similar to grabbing the interactable windows, and pulls away to where the operator would like the robot to look, an example of which can be seen in figure 1 The robot will then turn its head to look in that direction. An alternate strategy was considered and implemented, whereby the the operator could control the head in a direct control method. By turning

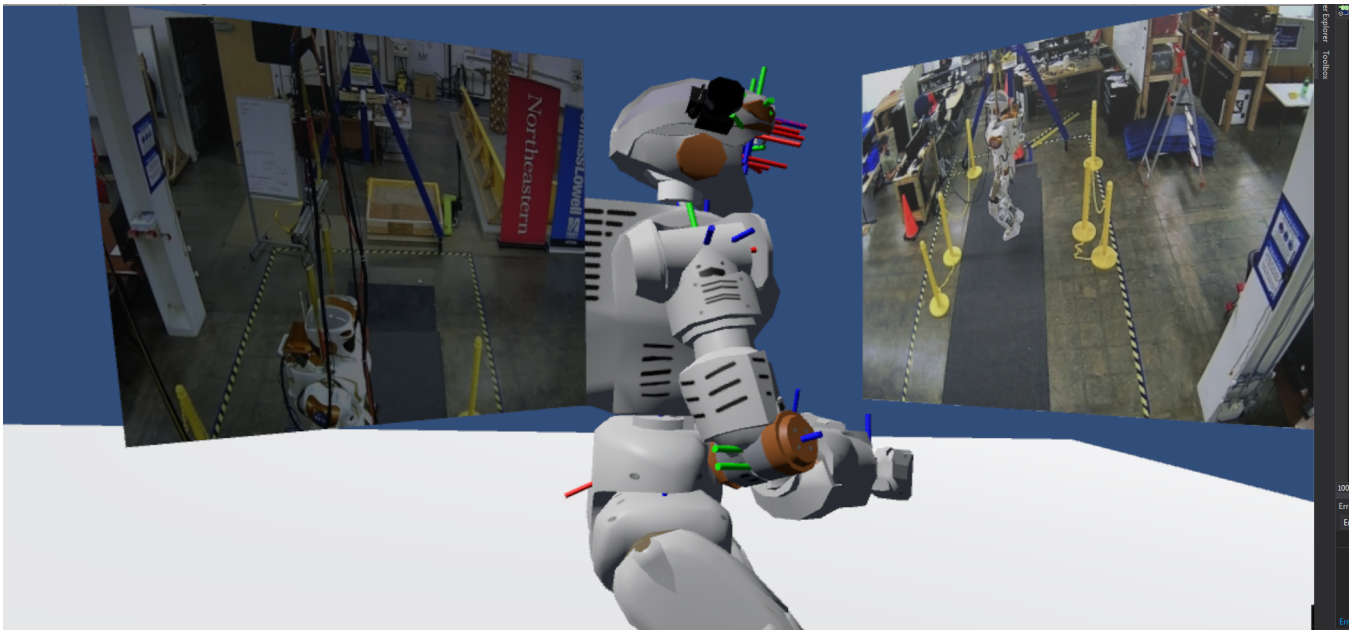

Figure 4: Virtual world with interactable windows, both displaying different external cameras near the robot.

Figure 5: Operator wearing the Manus VR glove to directly control the hands of the Valkyrie robot.

on an egocentric mode, the head would look in whichever direction the operator was looking in the virtual world.

4 FUTURE WORK

Our next steps are to perform a head-to-head comparison of our VR interface versus a functionally similar 2D keyboard and mouse interface. We will also examine what types of control and visualizations are most effective in different scenarios (e.g., commanding the robot to walk in open space or over an obstacle, pushing a fixed object like a button or manipulating a door handle). During such a study, users will interact with point clouds in a VR environment, hopefully making the use more comfortable and enhancing the capabilities of assisted autonomy behaviors.

5 CONCLUSION

When large amounts of sensor data are available, such as multiple camera images and point clouds, as is often the case with humanoid robots, using traditional interfaces can be cumbersome. As seen in the DRC Finals where an operator could have 4 display monitors and 6 camera feeds in addition to the robot model and point cloud rendering, the possibility of operator cognitive overload is very high. A temporary solution has been to use a combination of sensor fusion, as well as large amounts of training time and multiple operators in order to perform tasks acceptably. VR has enormous potential to change the way that operators operate remote robots, particularly complex robots working in 3D environments where the benefits of VR can be utilized. Shifting both the visualization and control into VR space can greatly reduce the cognitive load of interpreting sensor information to construct a mental 3D model of the environment. In this paper, we have laid out the creation and design our VR interface, which was inspired by similar 2D interfaces that saw success when used on humanoid robots in the DRC, adapted to take advantage of the unique benefits VR can provide. It is our hope that other researchers can use this information when considering to utilize a VR interface for remote robot control.

6 ACKNOWLEDGEMENTS

The work described in this paper was supported in part by the Department of Energy under award DE-EM0004482, by NASA under award NNX16AC48A, and by the National Science Foundation under award 1451427.

REFERENCES

[1] 2018. HTC Vive. (Jan 2018). https://en.wikipedia.org/wiki/HTC_Vive
[2] 2018. Oculus Rift. (Jan 2018). https://en.wikipedia.org/wiki/Oculus_Rift

[3] 2018. Room scale. (Feb 2018). https://en.wikipedia.org/wiki/Room_scale

[4] DARPA. [n. d.]. DARPA Robotics Challenge (DRC) (Archived). ([n. d.]). https://www.darpa.mil/program/darpa-robotics-challenge

[5] Simon Davis, Keith Nesbitt, and Eugene Nalivaiko. 2015. Comparing the onset of cybersickness using the Oculus Rift and two virtual roller coasters. In *Proceedings of the 11th Australasian Conference on Interactive Entertainment (IE 2015)*, Vol. 27. 30.

[6] HTC. [n. d.]. Discover Virtual Reality Beyond Imagination. ([n. d.]). https://www.vive.com/us/

[7] Matthew Johnson, Brandon Shrewsbury, Sylvain Bertrand, Tingfan Wu, Daniel Duran, Marshall Floyd, Peter Abeles, Douglas Stephen, Nathan Mertins, Alex Lesman, et al. 2015. Team IHMC's lessons learned from the DARPA robotics challenge trials. *Journal of Field Robotics* 32, 2 (2015), 192–208.

[8] ManusVR. [n. d.]. Manus VR | The Pinnacle of Virtual Reality Controllers. ([n. d.]). https://manus-vr.com/

[9] Curtis W Nielsen, Michael A Goodrich, and Robert W Ricks. 2007. Ecological interfaces for improving mobile robot teleoperation. *IEEE Transactions on Robotics* 23, 5 (2007), 927–941.

[10] Adam Norton, Willard Ober, Lisa Baraniecki, Eric McCann, Jean Scholtz, David Shane, Anna Skinner, Robert Watson, and Holly Yanco. 2017. Analysis of human–robot interaction at the DARPA Robotics Challenge Finals. *The International Journal of Robotics Research* 36, 5-7 (2017), 483–513.

[11] Paul Oh, Kiwon Sohn, Giho Jang, Youngbum Jun, and Baek-Kyu Cho. 2017. Technical Overview of Team DRC-Hubo@ UNLV's Approach to the 2015 DARPA Robotics Challenge Finals. *Journal of Field Robotics* 34, 5 (2017), 874–896.

[12] Fumio Okura, Yuko Ueda, Tomokazu Sato, and Naokazu Yokoya. 2013. Teleoperation of mobile robots by generating augmented free-viewpoint images. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 665–671.

[13] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. 2009. ROS: an open-source Robot Operating System. In *ICRA workshop on open source software*, Vol. 3. Kobe, Japan, 5.

[14] Nicolaus A Radford, Philip Strawser, Kimberly Hambuchen, Joshua S Mehling, William K Verdeyen, A Stuart Donnan, James Holley, Jairo Sanchez, Vienny Nguyen, Lyndon Bridgwater, et al. 2015. Valkyrie: Nasa's first bipedal humanoid robot. *Journal of Field Robotics* 32, 3 (2015), 397–419.

[15] Carnegie Robotics. [n. d.]. MultiSense SL. ([n. d.]). https://carnegierobotics.com/multisense-sl/

[16] uml robotics. [n. d.]. uml-robotics/ROS.NETUnity. ([n. d.]). https://github.com/uml-robotics/ROS.NET_Unity

[17] Unity. [n. d.]. Unity. ([n. d.]). https://unity3d.com/

[18] Holly A Yanco, Adam Norton, Willard Ober, David Shane, Anna Skinner, and Jack Vice. 2015. Analysis of human-robot interaction at the darpa robotics challenge trials. *Journal of Field Robotics* 32, 3 (2015), 420–444.