ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots

Article *in* Proceedings of the ... IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE/RSJ International Conference on Intelligent Robots and Systems · July 2018

5 authors, including:

David Whitney
Brown University
10 PUBLICATIONS **77** CITATIONS
SEE PROFILE

Eric Rosen
Brown University
10 PUBLICATIONS **54** CITATIONS
SEE PROFILE

Elizabeth K Phillips
Brown University
37 PUBLICATIONS **233** CITATIONS
SEE PROFILE

Stefanie Tellex
Massachusetts Institute of Technology
62 PUBLICATIONS **1,613** CITATIONS
SEE PROFILE

ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots

David Whitney[1], Eric Rosen[1], Daniel Ullman[1], Elizabeth Phillips[1], Stefanie Tellex[1]

***Abstract*— Virtual reality (VR) systems let users intuitively interact with 3D environments and have been used extensively for robotic teleoperation tasks. While more immersive than their 2D counterparts, early VR systems were expensive and required specialized hardware. Fortunately, there has been a recent proliferation of consumer-grade VR systems at affordable price points. These systems are inexpensive, relatively portable, and can be integrated into existing robotic frameworks. Our group has designed a VR teleoperation package for the Robot Operating System (ROS), ROS Reality, that can be easily integrated into such frameworks. ROS Reality is an open-source, over-the-Internet teleoperation interface between any ROS-enabled robot and any Unity-compatible VR headset. We completed a pilot study to test the efficacy of our system, with expert human users controlling a Baxter robot via ROS Reality to complete 24 dexterous manipulation tasks, compared to the same users controlling the robot via direct kinesthetic handling. This study provides insight into the feasibility of robotic teleoperation tasks in VR with current consumer-grade resources and exposes issues that need to be addressed in these VR systems. In addition, this paper presents a description of ROS Reality, its components, and architecture. We hope this system will be adopted by other research groups to allow for easy integration of VR teleoperated robots into future experiments.**

I. INTRODUCTION

Virtual reality (VR) provides a compelling interface for robots because it enables fluid interactions in the real physical world, and allows users to specify points and transforms in an intuitive way. VR interfaces provide potential for teleoperation, robot teaching and learning from demonstration, as well as debugging and fixing problems on the robot remotely.

A major benefit of VR systems is that they allow non-expert users to control robots. A direct mapping of robot manipulators to VR hand controllers creates an interface in which manipulators act as extensions of the users' hands. This human-system interface can allow everyday users to intuitively perform a variety of dexterous robot manipulation tasks without extensive training. On the other hand, many tasks require fine-grained manipulation which necessitates expert performance, taking time to acquire such experience. Thus, VR interfaces may also be a means to leverage the proficiency of expert human users to facilitate robots learning complex, fine-grained manipulation tasks. VR therefore permits non-experts to control robots, as well as leverages experts' experience in challenging domains.

[1]Brown University. {david_whitney, eric_rosen, daniel_ullman, elizabeth_phillips1}@brown.edu, stefie10@cs.brown.edu

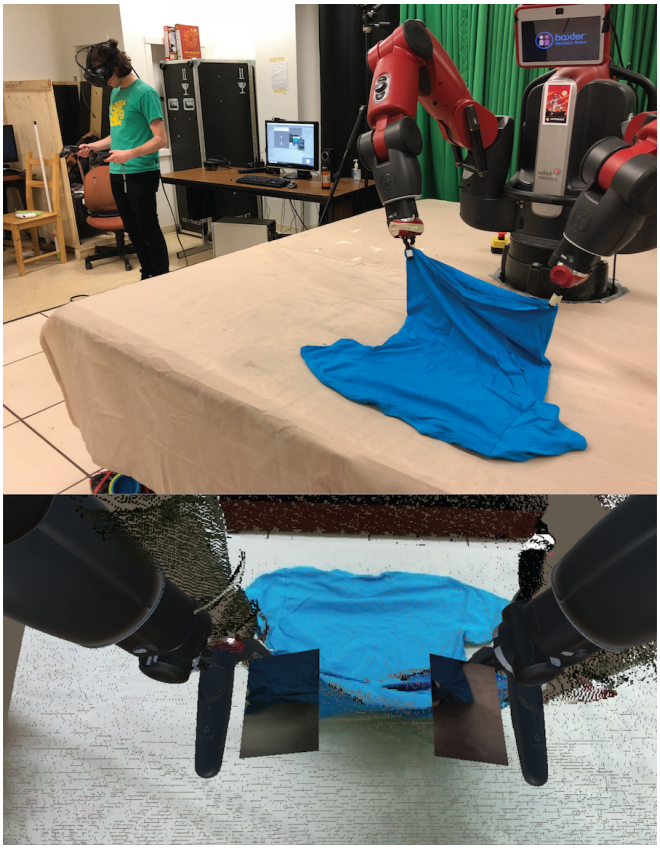

Fig. 1. Top image: An operator using ROS Reality VR to teleoperate a Baxter to fold a shirt. Bottom image: View of scene from VR headset. Note a point cloud, mesh model of robot, VR controllers, and wrist camera feeds from robot are all visible to the user.

However, integrating robots with a VR system is challenging. There is no standard interface to connect ROS [12] to standard virtual reality paradigms, such as Unity, so that it can be used with consumer-grade hardware, such as the HTC Vive. Additionally there is a lack of standardization in terms of tasks and use cases for these systems.

With this in mind, we present ROS Reality. ROS Reality is a VR and Augmented Reality (AR) teleoperation interface using consumer-grade VR and AR hardware with ROS-enabled robots. It allows users to view and control robots over-the-Internet using consumer-grade VR and AR hardware. ROS Reality has served as the technical basis for the VR research in Whitney et al. [18], and for the AR research in Rosen et al. [13]. A VR teleoperation demonstration using ROS Reality is shown in Fig. I. In this work, we focus on the VR system architecture and application of ROS Reality.

In this work, we detail our consumer-grade VR and AR teleoperation interface, ROS Reality. We discuss how the package allows for a ROS-networked robot, like Baxter from Rethink Robotics, to bilaterally communicate over the Internet with an HTC Vive through the Unity game engine. We also present the results of a pilot study conducted to test the efficacy of using ROS Reality to teleoperate a robot to perform 24 dexterous manipulation tasks. Portions of this work previously appeared in an extended abstract by Rosen et al. [14].

II. Related Work

Teleoperation enables robots to complete tasks that would otherwise be too difficult to complete autonomously, such as in the DARPA Robotics Challenge [5], and also allows humans to operate by proxy in environments that would normally place them in harm's way [1].

2D interfaces for robot teleoperation, especially over the Internet, have been popular in recent years [6]. Monitor and keyboard setups have been used to control robots for a variety of classical tasks, like motion planing and item grasping [16]. Further, web browsers have proven especially useful in allowing anyone around the world with a computer to teleoperate a robot, broadening the user-base of operators [11]. However, 2D monitor interfaces do not reflect the natural way that humans observe and interact with the 3D world. Our research has shown that a VR interface can address this problem as non-exert users were faster, more efficient, and preferred using a VR interface over a 2D monitor interface for teleoperating a robot [17].

Virtual reality interfaces and gantry systems offer intuitive means to directly map a user's actions to those of the robot they are controlling [17]. For example, the da Vinci Robot System is an immersive haptic telesurgery system which has improved surgical performance for both novice and experienced users [2]. Although powerful, the da Vinci robot and its interface is very task-specific to the surgical domain and stationary. Mallwitz et al. [10] developed a portable and easily-dressable exoskeleton that allowed a human user to naturally teleoperate a complex humanoid robot. This system is very intuitive to control, but again is limited to specific robots and is extremely expensive, heavily limiting the potential operator-base compared to web-based interfaces.

Recent advancements in graphics have made commercially available VR systems accessible to the gaming community. Systems like the HTC Vive, Oculus Rift, and Google Cardboard offer cheap and portable VR hardware. As a result, lab researchers have recently begun exploring these VR systems for robot teleoperation. Zhang et al. [20] used an HTC Vive to teleoperate a PR2 and perform imitation learning. Lipton et al. [9] also used a commercially available VR system for performing teleoperation on a Baxter. Our previous work [17] on comparing VR to 2D teleoperation systems also used an HTC Vive for the VR interface, enabled through ROS Reality. By having labs use the same VR systems, results and interfaces are easier to duplicate.

The proliferation of consumer-grade VR systems is very recent, so there has been little research on the efficacy of teleoperation interfaces that use this technology (e.g., [20]). Although task completion depends heavily on interface type and the particular robot, we were interested in exploring what complex tasks could be completed on our open-source software using a common research robot.

Our choice of objects and manipulation tasks to evaluate on was inspired by previous work on robot task benchmarks. Kasper et al. [8] created a program to generate an open-database of over 100 object models to be used for evaluating recognition, localization, and manipulation capabilities in service robots. Goldfeder et al. [7] released a collected dataset of items and stable grasps as a means for conducting machine learning and benchmarking grasp planning algorithms. One notable benchmark is the YCB object and model set, which is a set of accessible items chosen to include a wide range of common object sizes, shapes, and colors to test a variety of robot manipulation skills using accepted protocols [3]. The YCB dataset has made it easy and cheap for any research lab to evaluate a robot manipulator on general tasks over a large object dataset. Several of the tasks performed in this paper come from the YCB dataset.

III. ROS Reality

This section first provides a brief synopsis of interacting with a robot in virtual reality, and then a technical description of ROS Reality[1].

A. VR as a Teleoperation Interface

The two most common virtual reality systems today are the Oculus Rift and the HTC Vive. Our group develops with the HTC Vive due to superior room-scale tracking, but the following description of how to use VR as a teleoperation interface applies to both systems.

There are multiple ways of displaying the robot's state to the user, and mapping the user's input to the robot. We bin these different methods into two main categories: egocentric or robocentric.

In egocentric models, the human is the center of the virtual world, and virtually inhabits the same space as the robot. Lipton et al. [9]'s homunculus work and Zhang et al. [20] are examples of this egocentric mapping. Under these conditions, human users have reported feeling like they 'become the robot' or 'see out of the robot's eyes'.

In a robocentric model, the human and robot share a virtual space, but are not necessarily superimposed on one another. The model we used for evaluating ROS Reality [17], falls into this category. Under this model, the human walks around a virtual model of the robot, and controls its arms by virtually grabbing and dragging them. We therefore call this model a virtual gantry system.

[1]Full source code is available at `https://github.com/h2r/ros_reality`

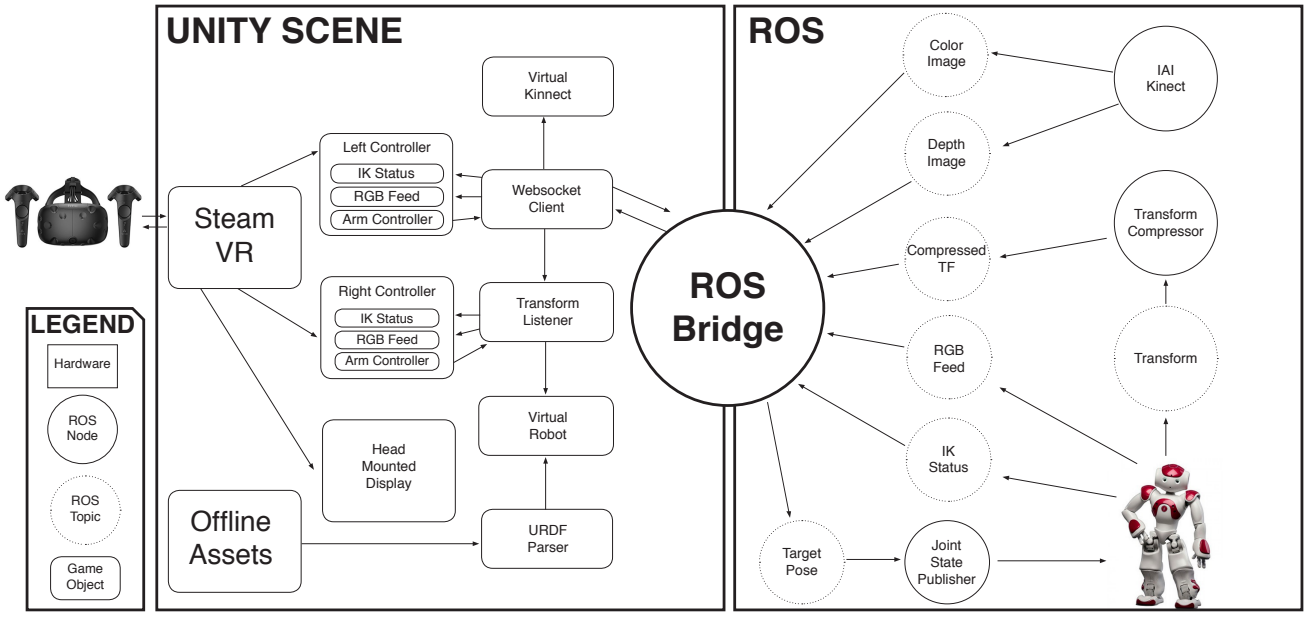

Fig. 2. A diagram detailing the architecture of the ROS Reality system

B. System Overview

An HTC Vive is connected to a computer running the Unity game engine. Unity builds a local copy of our robot based on its URDF with a custom-made URDF parser. Unity connects to a ROS network over the Internet via a Rosbridge WebSocket connection [4]. The pose and wrist cameras of the robot are sent via this WebSocket connection, as well as the color and depth image of a Kinect 2 mounted to the robot's head. The color and depth image are built into a point cloud in Unity via a custom shader. When the user holds down a deadman's switch, the pose of the user's controllers are sent back to the robot, which uses an inverse kinematics solver to move the robot's end effectors to the specified poses. Refer to Fig. 2 for a visual overview of the ROS Reality system.

C. ROS

ROS (Robot Operating System) is a set of tools and libraries to help program robot applications. ROS connects processes of programs, known as nodes, that perform different functions. Nodes communicate by streaming data over channels, or topics, on a local TCP network, known as a ROS network. Nodes create publisher objects to publish data over the network on a topic, or subscriber objects to subscribe to a topic. ROS provides an API to create nodes in C++ or Python. All nodes written for ROS Reality were written in Python.

ROS Reality launches a Kinect2 ROS node [19], two RGB camera feeds (one for each wrist camera of the robot), a Rosbridge WebSocket server [4], a custom ROS node that converts the full transform (TF) of the robot to a compact string, and another ROS node that listens for target poses from the VR systems, queries the robot's Inverse Kinematic (IK) solver, and moves the robot to the IK solution if found, or reports an IK failure if one is not found.

D. HTC Vive

The HTC Vive is a consumer-grade virtual reality system. It has three tracked objects: one head-mounted display (HMD), and two wand controllers. Each device is tracked via a set of two infrared pulse laser emitters, known as lighthouses, allowing for tracking via time-of-flight calculations[2]. Each tracked object is positionally and rotationally tracked, with roughly 1-2mm of error. The wand controllers are fully wireless, and the HMD connects to a computer via a USB and HDMI cable. Each controller has a touch-pad, trigger, and two buttons for user input.

The HTC Vive supports several game and physics engines, but the initial (and in our opinion best supported) development platform is Unity[3]. The Vive connects to Unity through a software package called SteamVR.

E. Unity

Unity is a game engine that is used for many popular 2D, 3D, and Virtual/Augmented/Mixed Reality applications. It has a built-in physics engine that can handle contact dynamics, as well as material simulation (such as water, sand, or cloth). It supports integration with most common VR (and AR) hardware, and provides a shader language for writing custom GPU shaders.

An open Unity environment is called a scene. In this scene are a collection of the atomic unit of Unity, the GameObject. Attached to each GameObject are a set of Components.

[2]The Oculus Rift uses multiple cameras to track the HMD and controllers.
[3]Formally known as Unity3D.

Fig. 3. An image of the PR2 robot visualized in Unity from the URDF Parser of ROS Reality.

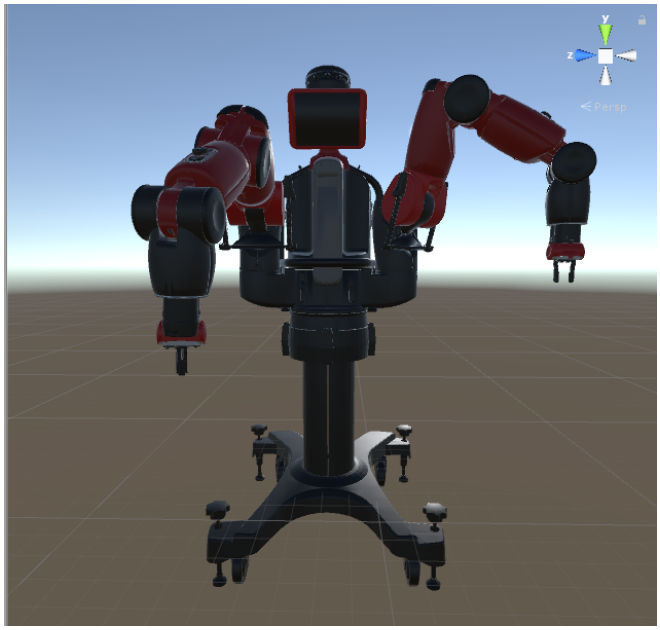

Fig. 4. An image of the Baxter robot visualized in Unity from the URDF Parser of ROS Reality.

There are dozens of types of components, but the most important for our purposes is the script. A script is a small C# program that is executed at every rendering frame. The functionality of ROS Reality is implemented via a set of these Unity scripts.

F. ROS Reality

ROS Reality is a set of programs that allows a user to view and control a ROS-enabled robot over-the-Internet in VR. ROS Reality is composed of a set of C# scripts, described below.

1) WebSocket Client: This script is a C# implementation of the default Rosbridge client, roslibjs [15]. It supports advertising, subscribing, and publishing to ROS topics. All messages are sent and received in a JSON format, and data is encoded in base64 as per the Rosbridge specification.

2) URDF Parser: This script parses a Unified Robot Description Format (URDF) file and builds a hierarchy of GameObjects which comprise the robot. URDF is an XML-based specification for representing robot models common to all ROS-enabled robots. URDFs include information about each part of the robot, known as links, and how the links of a robot are connected, known as joints. The URDF Parser creates a GameObject for each link, and connects them according to the joints. Currently, we have successfully tested our URDF parser with a PR2, and Baxter, as seen in Figs. 3 and 4.

The virtual robot has physical properties that can be simulated via Unity's physics engine. This allows the robot to interact with other GameObjects, useful for practicing teleoperation interactions in simulated scenarios.

3) Transform Listener: The Transform Listener subscribes to (a compact representation of) the robot's transform (TF) and moves the virtual robot to the same pose as the real robot. The ROS TF topic has the position and rotation (represented as a quaternion) of each link, which this script reads and applies to each link of the simulated robot.

One difficulty in doing this is that ROS and Unity use different coordinate frames. The Transform Listener therefore first converts the ROS positions and rotations via the following equations.

Positions:

$$x_{unity} = -x_{ros} \tag{1}$$
$$y_{unity} = z_{ros} \tag{2}$$
$$z_{unity} = -y_{ros} \tag{3}$$

and rotations:

$$qx_{unity} = qx_{ros} \tag{4}$$
$$qy_{unity} = -qz_{ros} \tag{5}$$
$$qz_{unity} = qy_{ros} \tag{6}$$
$$qw_{unity} = qw_{ros} \tag{7}$$

4) RGB Camera Visualizer: To visualize camera feeds from the robot, this script subscribes to a specified camera topic. When it receives the camera image it converts it from base64 and textures a plane GameObject with the camera feed. The plane GameObject is attached to the user's wand controller, so the user can always see it during manipulation. This script supports images in JPG or PNG formats, but ROS Reality always uses JPG for bandwidth reasons.

5) Kinect PointCloud Visualizer: The Kinect PointCloud Visualizer script uses a GPU shader to construct a point cloud out of the RGB camera image and raw depth map

from a Kinect 2. The script subscribes to the RGB and depth topics of the Kinect and passes them as textures to a custom geometry shader. This shader creates a colored quad for each pair of pixels in the RGB and depth images. The color of the quad is simply the color of the associated RBG pixel. The position must be calculated. Each pixel in the depth image is the distance in millimeters of that pixel from the camera plane, so first we convert from millimeters to meters, and calculate the position of the quad relative to the camera. We then multiply that position by the transformation matrix of the Kinect in the Unity scene to get the world space position of the quad. The world space position is finally multiplied by the view and projection matrices, and passed to the vertex shader.

6) Arm Controller: This script allows the user to send target end effector coordinates to the robot. When a dead-man's switch is held down (the side grip buttons on an HTC Vive) the current position and orientation of the controller are converted from the Unity coordinate frame to the ROS coordinate frame and published over a topic to a node in the ROS network that queries the robot's built in IK solver and moves the robot if a solution is found. Additionally, this script lets the user open and close the gripper with the trigger of the wand controller. This is also accomplished by sending a message over a topic to the robot.

The conversion from Unity to ROS for positions can be inferred from equations 1, 2, 3, and for rotation is calculated via the following quaternion operation:

$$(qx, qy, qz, qw)_{ros} = (qx, qz, -qy, qw)_{unity} * (0, 1, 0, 0) \tag{8}$$

7) IK Status Visualizer: This script subscribes to the current status of the robot's IK solver and turns the users wand controller red if the IK solver failed. This lets the user know if the target position they sent to the robot cannot be reached.

G. Robot

We use a Baxter from Rethink Robotics. Baxter is a robot designed for industrial automation applications, also serving as a useful research platform. Baxter has a fixed base and display screen head, with two 7 DoF arms and grippers with force sensing that enable Baxter to dexterously manipulate a variety of objects. We attached rubber grips that come in the Baxter toolkit in order to maximize the friction at the end effector.

We have also connected ROS Reality to a simulated PR2 in Gazebo, and have been able to watch the robot move in real time, but have not yet set up the infrastructure to control that robot.

IV. Long-Distance Teleoperation Trial

In order to test the efficacy of ROS Reality for long-distance teleoperation, we had a human operator control a robot 41 miles away, at a separate university. In this trial, we were able to successfully stack 12 cups back to back, as well as play a short game of chess by picking and placing pieces. The user reported no lag or bandwidth issues.

V. VR Teleoperation Task Feasibility

We considered desirable skills for a manipulator robot to have. Our goal was to answer two questions:

1) Is the robot physically capable of performing certain tasks?
2) If so, can a human teleoperating the robot in VR complete this task?

Because the physical capabilities of the robot depends on the hardware, our specific study used a research Baxter robot. Refer to Section III-G for more information.

In order to answer these two questions, two authors of this paper acted as the expert teleoperators for performing the trials. For question one, we physically moved the robot's arms in real life to complete the task. Direct manipulation of the robot's arms gives users the best perception of the scene, along with direct haptic feedback from the robot and the environment. We used this methodology of Direct Manipulation in our previous VR study as a good measure for task feasibility [17]. For question two, we used our ROS Reality interface mentioned in Section III to perform VR teleoperation to complete the tasks.

Baxter's 7 DoF arms are equipped with parallel electrical grippers at the end effector, such that Baxter is effectively able to grip, push, pull, and rotate objects. However, Baxter's ability to grip objects is limited by the nature of its parallel electrical grippers, as well as the ability of its arms to exert push and pull forces.

We derived a set of 24 tasks by choosing different common manipulation tasks that could be relevant for manipulator robots in a variety of domains (e.g., home personal assistant uses, socially assistive applications), while simultaneously attempting to pick tasks that we believed possible to implement on Baxter. Two groups of tasks were chosen, such that one half of the tasks could be completed using one manipulator and the other half required use of both manipulators at the same time. In addition, tasks were chosen to represent an array of different movements (i.e., grip, push, pull, and rotate).

For each task performed via direction control and via ROS Reality, we performed a maximum of 5 attempts to complete the task. A given task was deemed feasible if we were able to complete it at least once. We report our results for the tasks in Table I and Fig. V-B.

A. Manipulation Tasks

1) *Block Stacking* - Stack ten 3x3cm wood blocks in a column.
2) *Unscrew Bottle* - Unscrew the cap to a bottle.
3) *Uncap Marker* - Remove cap from an Expo marker.
4) *Hinge Board* - Open all six latches on Melissa and Doug Latches Wooden Activity Board.
5) *Stir Pot* - Stir a wooden spoon in a metal pot.
6) *Push Spacebar* - Push the spacebar button on a keyboard.

TABLE I

TASK FEASIBILITY EVALUATIONS

List of tasks and performances. One manipulator tasks are above the line, while two manipulator tasks are below the line.

Task	Task Number	Direct?	VR?
Block Stacking	1	Yes	No
Unscrew Bottle	2	No	-
Uncap Marker	3	Yes	Yes
Hinge Board	4	No	-
Stir Pot	5	Yes	Yes
Push Spacebar	6	Yes	Yes
Move Checker Piece	7	Yes	Yes
Squeeze Purell	8	Yes	Yes
Insert Connect 4 Piece	9	Yes	Yes
Toss and Catch Ball	10	No	-
Use Fork	11	Yes	Yes
Unzip Zipper	12	No	-
Open Chips	13	No	-
Carry plate	14	Yes	Yes
Open Glass Bottle	15	No	-
Peel Potato	16	Yes	No
Uncap Marker	17	Yes	Yes
Dust Pan	18	Yes	Yes
Fold Shirt	19	Yes	Yes
Handover Expo	20	Yes	Yes
Open Box	21	Yes	Yes
Tap a Paradiddle	22	Yes	Yes
Toss and Catch Ball	23	No	-
Tie Shoelace	24	No	-

7) *Move Checker Piece* - Pick and place a checker piece on a board.
8) *Squeeze Purell* - Squeeze out Purell from the bottle.
9) *Insert Connect 4 Piece* - Insert a Connect 4 piece into the slot.
10) *Toss and Catch Ball* - Toss a juggling ball up and catch it in the same hand.
11) *Use Fork* - Get a piece of food onto a plastic fork.
12) *Unzip Zipper* - Unzip a loose zipper.
13) *Open Chips* - Open a plastic bag of chips.
14) *Carry Plate* - Carry a plate with an item on it from one location to another.
15) *Open Glass Bottle* - Use a bottle opener to open a glass bottle.
16) *Peel Potato* - Use a peeler to peel the skin of a potato.
17) *Uncap Marker* - Remove cap from an Expo marker.
18) *Dust Pan* - Use a dust pan to sweep small blocks.
19) *Fold shirt* - Fold a T-shirt.
20) *Handover Expo* - Handover a pen from one manipulator to the other.
21) *Open Box* - Open a shoebox.
22) *Tap a Paradiddle* - Tap to the rhythm of paradiddle.
23) *Toss and Catch Ball* - Toss a ball from one manipulator and catch in other.
24) *Tie Shoelace* - Tie a shoelace into a knot.

B. Discussion

Overall, VR control of the Baxter robot via ROS Reality was a success. For single manipulator tasks, eight out of twelve were achieved through direct manipulation, with seven of those eight tasks achieved through VR. For the two manipulator tasks, eight out of twelve were also achieved through direct manipulation, and again, seven of those eight tasks were completed through VR.

The two expert teleoperators who attempted the trials in VR reported a relative ease of use of the system, with several observations of note. In general, the direct kinesthetic manipulation of the robot permitted the easiest, fastest completion of tasks for the tasks that proved physically possible for the robot. This is unsurprising, given the familiar nature of guiding a human on how to physically move to perform a task, as well as getting to directly observe the robot's workspace. The users found VR most useful when the task required complex movements of the robot's joints. During direct manipulation, resistance in the robot's manipulators forced the operators to use two hands to move the robot's limb. This meant the operator could effectively only move one manipulator at a time, and had to extend a fair amount of force to move the manipulator to a complex position. By contrast, in VR the user does not have to constantly parametrize joint angles of the robot, instead specifying end effector pose and having the robot calculate and navigate the correct trajectory.

Our trials revealed that force exerted by the robot was a limiting factor in whether or not tasks could be completed in the first place, with the robot unable to generate sufficient force for certain tasks (e.g., toss and catch ball, open chips). However, manipulation tasks that did not require substantial force were largely successful. Rotation did not pose a major obstacle to task completion. Manipulation tasks requiring dexterous grasping were also limited by the parallel electrical grippers on the robot used in our evaluation but are likely to be much easier for robots with higher DoF end effectors. Finally, the robot's built-in collision detection system prevented completion of the two manipulator task of opening the bag of chips, with the system preventing the robot's arms from coming close enough to grip the bag of potato chips on both sides of the bag.

VI. FUTURE RESEARCH

The system described in this paper serves as the foundation for a host of future research. For instance, Learning from Demonstration (LfD) is a popular approach to teach robots complex manipulation tasks because it utilizes human expertise, judgment, and decision making. However, obtaining demonstrations from human participants in laboratory studies is both time and resource intensive. An approach to addressing this problem has been to develop algorithms that require fewer and fewer demonstrations from humans; which is challenging, and will inevitably require at least one, if not more, demonstrations by human users with physical robots. Even with this solution, at some point for most tasks, users will have to interact with robots to help train them. However, using VR as a mechanism to gather LfD data at scale is a promising alternative. Learning complex tasks from task experts can be challenging for autonomous systems, with VR sidestepping this issue by enabling users to directly control systems while leveraging the benefits of the system.

Fig. 5. The results of the VR task attempts. Green outlines indicates task was completed in VR, red outlines indicates that the task failed in VR.

Demonstrations could be provided to virtual robots by users accessed over the Internet in a crowdsourcing paradigm, completing tasks at scale, and thus addressing participant and resource limitations that currently plague extant LfD training methods. VR coupled with ROS Reality has the potential to offer a cost and time-effective solution to this challenge.

VII. Conclusion

Virtual reality is becoming increasingly available to everyday users, as hardware platforms steadily decrease in cost. These systems also represent an intuitive interface for controlling robots. In this paper we offer an open-source VR teleoperation package, ROS Reality, that makes any ROS-enabled robot controllable by any Unity-compatible VR headset. This work also identifies and tests robot manipulation tasks using ROS Reality with a consumer-grade VR headset.

References

[1] Carlos Beltrán-González, Antonios Gasteratos, Angelos Amanatiadis, Dimitrios Chrysostomou, Roberto Guzman, András Tóth, Loránd Szollosi, András Juhász, and Péter Galambos. Methods and techniques for intelligent navigation and manipulation for bomb disposal and rescue operations. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pages 1–6. IEEE, 2007.

[2] John C Byrn, Stefanie Schluender, Celia M Divino, John Conrad, Brooke Gurland, Edward Shlasko, and Amir Szold. Three-dimensional imaging improves surgical performance for both novice and experienced operators using the da vinci robot system. *The American Journal of Surgery*, 193(4):519–522, 2007.

[3] Berk Calli, Arjun Singh, Aaron Walsman, Siddhartha Srinivasa, Pieter Abbeel, and Aaron M Dollar. The ycb object and model set: Towards common benchmarks for manipulation research. In *Advanced Robotics (ICAR), 2015 International Conference on*, pages 510–517. IEEE, 2015.

[4] Christopher Crick, Graylin Jay, Sarah Osentoski, Benjamin Pitzer, and Odest Chadwicke Jenkins. Rosbridge: Ros for non-ros users. In *Robotics Research*, pages 493–504. Springer, 2017.

[5] Christopher M Dellin, Kyle Strabala, G Clark Haynes, David Stager, and Siddhartha S Srinivasa. Guided manipulation planning at the darpa robotics challenge trials. In *Experimental Robotics*, pages 149–163. Springer, 2016.

[6] Ken Goldberg, Michael Mascha, Steve Gentner, Nick Rothenberg, Carl Sutter, and Jeff Wiegley. Desktop teleoperation via the world wide web. In *Robotics and Automation, 1995. Proceedings., 1995 IEEE International Conference on*, volume 1, pages 654–659. IEEE, 1995.

[7] Corey Goldfeder, Matei Ciocarlie, Hao Dang, and Peter K Allen. The columbia grasp database. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 1710–1716. IEEE, 2009.

[8] Alexander Kasper, Zhixing Xue, and Rüdiger Dillmann. The kit object models database: An object model database for object recognition, localization and manipulation in service robotics. *The International Journal of Robotics Research*, 31(8):927–934, 2012.

[9] Jeffrey I Lipton, Aidan J Fay, and Daniela Rus. Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. *IEEE Robotics and Automation Letters*,

3(1):179–186, 2018.

[10] Martin Mallwitz, Niels Will, Johannes Teiwes, and Elsa Andrea Kirchner. The capio active upper body exoskeleton and its application for teleoperation. In *Proceedings of the 13th Symposium on Advanced Space Technologies in Robotics and Automation. ESA/Estec Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA-2015). ESA*, 2015.

[11] Gunter Niemeyer and Jean-Jacques E Slotine. Toward bilateral internet teleoperation. *Beyond Webcams: An Introduction to Online Robots*, page 193, 2002.

[12] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[13] Eric Rosen, David Whitney, Elizabeth Phillips, Gary Chien, James Tompkin, George Konidaris, and Stefanie Tellex. Communicating Robot Arm Motion Intent Through Mixed Reality Head-mounted Displays. In *International Symposium on Robotics Research*, 2017 in press.

[14] Eric Rosen, David Whitney, Elizabeth Phillips, Daniel Ullman, and Stefanie Tellex. Testing Robot Teleoperation using a Virtual Reality Interface with ROS Reality. *Human-Robot Interacton (HRI), 2018 Workshop on Virtual, Augmented and Mixed Reality*, 2018.

[15] Russell Toris, Julius Kammerl, David V Lu, Jihoon Lee, Odest Chadwicke Jenkins, Sarah Osentoski, Mitchell Wills, and Sonia Chernova. Robot web tools: Efficient messaging for cloud robotics. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 4530–4537. IEEE, 2015.

[16] Jean Vertut. *Teleoperation and Robotics: Applications and Technology*, volume 3. Springer Science & Business Media, 2013.

[17] David Whitney, Eric Rosen, Elizabeth Phillips, George Konidaris, and Stefanie Tellex. Comparing robot grasping teleoperation across desktop and virtual reality with ros reality. 2017.

[18] David Whitney, Eric Rosen, Elizabeth Phillips, George Konidaris, and Stefanie Tellex. Comparing Robot Grasping Teleoperation across Desktop and Virtual Reality with ROS Reality. In *International Symposium on Robotics Research*, 2017 in press.

[19] Thiemo Wiedemeyer. IAI Kinect2. `https://github.com/code-iai/iai_kinect2`, 2014 – 2015. Accessed June 12, 2015.

[20] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *arXiv preprint arXiv:1710.04615*, 2017.