Baxter's Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing

Jeffrey I. Lipton, Aidan J. Fay, and Daniela Rus

***Abstract*—We demonstrate a low-cost telerobotic system that leverages commercial virtual reality (VR) technology and integrates it with existing robotics control infrastructure. The system runs on a commercial gaming engine using off-the-shelf VR hardware and can be deployed on multiple network architectures. The system is based on the homunculus model of mind wherein we embed the user in a VR control room. The control room allows for multiple sensor displays, and dynamic mapping between the user and robot. This dynamic mapping allows for selective engagement between the user and the robot. We compared our system with state-of-the-art automation algorithms and standard VR-based telepresence systems by performing a user study. The study showed that new users were faster and more accurate than the automation or a direct telepresence system. We also demonstrate that our system can be used for pick and place, assembly, and manufacturing tasks.**

***Index Terms*—Human-centered automation, human-centered robotics, telerobotics and teleoperation, virtual reality and interfaces.**

I. Introduction

WE BELIEVE that the future of manufacturing will require a combination of robots and people, interacting in the physical and virtual worlds to enhance each other's capabilities. Humans could supervise the robots from a distance and, when necessary, connect seamlessly to a robot to command it to do a task, teach a new task, or help it recover from a failure. Barriers to working such as physical health, location, or security clearance could be reduced by decoupling physicality from manufacturing tasks. Virtual reality (VR) is a key enabling technology for this vision. We demonstrate a VR system for achieving such human-robot collaboration using the homunculus model of mind.

The homunculus is a logical fallacy illustrated in the "Cartesian Theater" criticism of Descartes's mind body dualism [1], [2]. It is said to be the thing sitting in a control room inside of a persons head, looking through a person's eyes, controlling their actions. It is a non-terminating recursive definition because inside each homunculus it is implied there is another smaller homunculus. While this is a terrible definition of human intelligence, it is an appropriate definition of the intelligence of a teleoperated robot. Inside the robot there is a human who is aware, in a control room, seeing through its eyes and controlling its actions.

Our homunculus inspired approach lets a virtual reality system give a human user the same view point and feeling of co-location with the robot, without the need for the system to directly connect the states of a user and robot. The homunculus model of VR teleoperation accomplishes this by embedding the user inside of a Virtual Reality Control Room (VRCR). The user's state is mapped to the VRCR, and the VRCR is mapped to the robotic system. This decouples the provision of sensory stimuli to the user from communication with the robot. The user selectively engages with controls inside the VRCR to adjust the coupling of their movements to the robot's movements.

We built our system using current consumer VR hardware, thereby leveraging existing commercial software and gaming infrastructure and integrated the commercial software with existing standard robotics infrastructure. The combination of the VRCR and our system architecture enables teleoperation of a robot over a local wired network, a wireless connection in the same building, and even over a hotel's wireless connection from a different city.

We tested our system against a direct mimicking system, and automated processes by having users grab and then stack blocks into an assembly. The homunculus system performed with greater accuracy and speed when compared to a direct mimicking system. Our system had a 75% success rate, compared with a 45% success rate for the direct mimicking system and a a 66% success rate reported for the state of the art algorithm. As a demonstration of system capability and flexibility we performed fixture-less assembly tasks, manipulated and secured wires onto wooden structures with a commercial staple gun, picked up screws, handled flexible materials, and complex shapes.

In this letter we contribute:

1) The homunculus model for VR teleoperation
2) Experimental performance validation of the system
3) A series of demonstrations of the telerobotic system

II. Background

Virtual reality based teleoperation systems have generally fallen into two categories we call: direct and cyber-physical. Each of these systems selectively map parts of an object's state between the user's reference frame U to the robot's reference frame R. Each object's state consists of a position and orientation along with any other state information such as an image

Manuscript received February 15, 2017; accepted July 9, 2017. Date of publication August 9, 2017; date of current version August 17, 2017. This letter was recommended for publication by Associate Editor G. Salvietti and Editor Y. Yokokohji upon evaluation of the reviewers' comments. This work was supported by the Boeing Corporation. *(Corresponding author: Jeffrey I. Lipton.)*

The authors are with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: jlipton@mit.edu; afay2017@francisparker.org; rus@csail.mit.edu).

Digital Object Identifier 10.1109/LRA.2017.2737046

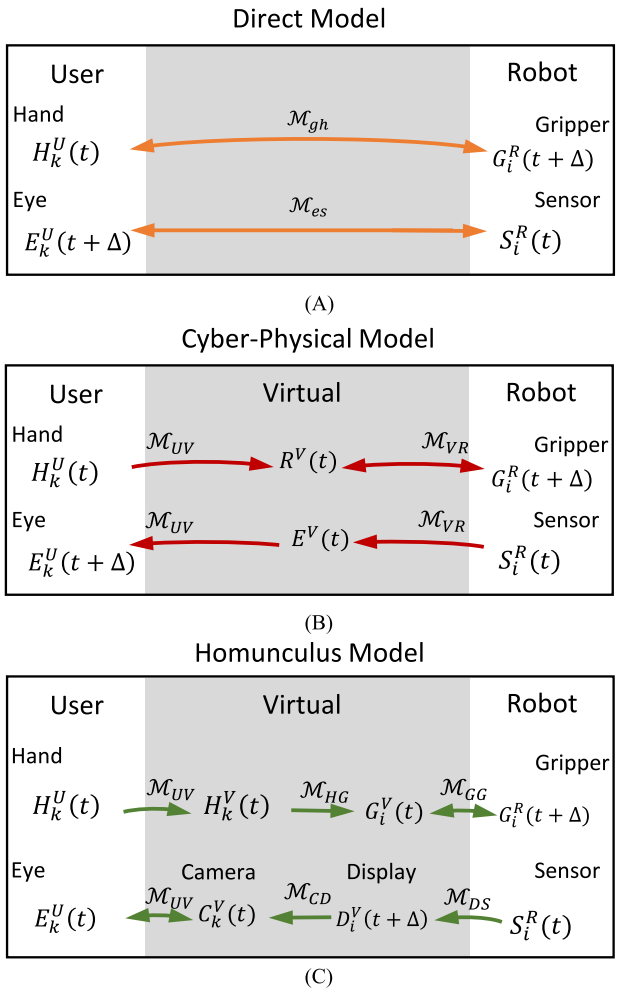

Fig. 1. Telerobotic system's mapping models. Mapping direction is denoted by an arrow and the mapping is named by $\mathscr{M}$ with subscripts. Cyber-physical and homunculus both have a virtual space, but use it differently.

or sensor values. The mappings directly affect the user experience. A user's proprioception is engaged by directly mapping their movements to an noticeable output in a space. This is what causes a person to feel present in a virtual reality environment [3]. The mappings may be one way, or bi-directional, and may map part or all of the state. The mapping method is critical to system function [4]. Mappings also have a temporal component, mapping the user's state at time t to time $t+\Delta$ on the robot. Unexpected incongruities in mappings, such as long or variable delays or relative motions, between proprioception and vision can lead to the nausea and motion sickness known as simulator sickness [5]. This means quickly mapping user eye movement and hand movement to new stimuli is critical to user experience.

The position and orientation of objects are represented in a specified reference frame by either a transformation matrix ($\bar{T}$), or a position vector ($\overrightarrow{r}$) with a unit quaternion (Q). We denote each object's reference frame by a super script. We use subscripts to denote which object we are referencing in the set of similar items. The state of each of the user's hands (H_k^U) and eyes (E_k^U) can be affected by and can effect a robot's grippers (G_i^R) and sensors (S_i^R) using a series of mappings($\mathscr{M}$) shown in Fig. 1. Mappings are denoted with subscripts referencing the object types (H,E,S, or G) it maps between. Various teleoperation systems have different mappings and required latencies between action and feedback.

In direct VR systems, there is no intermediate space between the user and the robot. The users vision is directly coupled to the robots state as seen in $\mathscr{M}_{es}$ in Fig. 1(a). If the cameras are stationary relative to the robot then $\mathscr{M}_{es}$ only transmits images from the sensor to the eyes. Movement of the user's head does not affect the image they see. The brain of the user constantly expects updates based on motion and does not receive it because of the mapping. This can lead to user fatigue and nausea. An alternative is to have the mapping $\mathscr{M}_{es}$ include movement feedback from the user. In these mimicking systems the state of the cameras is affected by the user. If only part of orientation and movement of the user is used, the user must limit their motion. The UPenn developed Dora platform attempts to mimic the complete position and orientation of the user's eyes with a moving camera system [6]. The large and complex nature of the camera rig limits its viability. VR applications have a minimum frame rate of 60 Hz to prevent nausea [7]. Because in direct systems, the user's vision is directly mapped from a camera, any delay in the update from the cameras due to network conditions or camera hardware will limit the rate of updates to the user. Additionally, a hardware-in-the-loop solution will have to eliminate the delays from mechanical hardware response.

For direct systems, there are several options for connecting the user and the robot's arms. With direct piloting systems, a user operates a standard controller such as a keyboard and mouse, haptic pen, or video game controller while wearing a VR headset [4], [8], [9]. This does not lend the user to feel co-located with the robot. Mimicking the body movements of a user can also be problematic. Mimicking of the user hands (H_k^U) can be done with hand controllers or robotic arm systems [10], [11]. Providing complete capture of body movements requires motion capture systems such as Kinect sensors or Vicon systems [12], [13]. The incongruities between a user's and the robot's body shape poses numerous challenges. Mapping movements directly can lead to user discomfort or failure to utilize the entire work-space. For example, Baxter can reach positions and contortions that are painful or impossible for a user, such as putting both arms behind itself. One solution for these systems is to closely mirror the abilities of a human user by changing the shape and design of the robot [14]. However this level of mirroring is not compatible with most commercial robots. The other solution is to only mimic the position and orientation of the end effector. However this creates a mismatch between the proprioception of the user, and the visual state of the user.

The other approach to VR is the Cyber-Physical System (CPS) approach. In these systems, shared virtual space for the user and robot is created in which a virtual dual of the robotic system (R^V), the dual of robot's environment (E^V) and dual of the user is represented in a computer [15]. As seen in Fig. 1(b), the mapping $\mathscr{M}_{UV}$ maps the user into the virtual space and is typically a 1–1 mapping. Mapping $\mathscr{M}_{VR}$ maps the robot and the environment 1–1 into the virtual space. The user interacts with the robot while inhabiting a shared virtual space [16]. The duals are constantly updated from the actual robot, environment, and

user. This requires making a complete model of the robot and environment, and transmitting updates to the models over the network. While this is a great tool for training robotic systems and teaching repetitive tasks, it lacks the human in the loop of the controls, making it difficult to respond to dynamic situations [16]. It also requires a large amount of data to capture the environment for virtual representation, or a significant amount of a priori knowledge for building models of the environment.

III. Design of VR Environments for The Homunculus Model

In the homunculus model, the mappings between the user's state and the robot's state are mediated in a Virtual Reality Control Room (VRCR). Unlike the CPS model, the virtual space is not shared with a complete virtual dual of the robot. Instead the user interacts with displays and objects in the space itself. The human user space is mapped into the virtual space, and the virtual space is then mapped into the robot space to provide a sense of co-location. The user is mapped into VRCR with a 1–1 correspondence. The VRCR is shaped after the head of the robot with a window showing the robot's environment. Information about the environment is mapped to 2D surfaces outside the window, rather than to the voxels in the virtual space itself. Forward facing cameras collect information from a 1 sq meter area and are mapped to a display which is 11 meters wide by 10 meters tall and 6 meters away from the user in virtual space. The user's brain infers the 3D representations directly, rather than having a GPU or CPU interpret the data to 3D and then back into images for each eye. This reduces the amount of data which must be streamed and processed by the system. It also causes the user to see the objects in the window as much larger than themselves. Unlike the mimicking model many different sensors can be displayed in a VRCR without overlaying on top of other camera and sensor feeds. Multiple types and sources of information can be arranged in a virtual space. This display method separates the mapping $\mathscr{M}_{VR}$ of the CPS model into $\mathscr{M}_{GG}$ and $\mathscr{M}_{DS}$ in the homunculus model because environmental information is treated differently from robot state information.

Like the CPS systems, the state of the user's eyes are decoupled from the robotic systems state by the virtual space. The users eyes are represented by virtual camera duals in the space. Their state is measured from the users position and the image state of the users eye is rendered in the virtual space in a separate loop from the robot movements and sensor feeds. This allow the system to compensate for head movement since the virtual camera updates are rendered on a local computer, regardless of sensor update rates. Using commercial VR game engines to render most of the environment allows for the use of previously developed solutions such as Asynchronous Timewarp to compensate for delays in rendering [17]. The effect of a delay from the camera signal is negligible to the user since their vision queues are generated from the VRCR rather than from the camera directly.

Interacting with the robot state information is done with controls that are scaled 1–1 into the virtual space. Control orbs and position markers float in the VRCR. By grasping control orbs in various ways, or not grasping at all, users selectively engage and manage their connection to the grippers. With the controls released, the user is free to move, fidget, or change their focus without affecting the state of the robot. In Algorithm 1, we see that the user can select the mapping $\mathscr{M}_{HG}$ they use. There are three types of mapping options: position and orientation, position and single angle, and position only. In the position and orientation mapping, the entire state of the user's hand is transmitted to the robot. In the position and single angle mapping, the quaternion describing the hands orientation is replaced with a quaternion generated from a preassigned unit vector $\hat{n}_o$ and a TaitBryan angle θ_H^U. This contains the arm to operate tangent to a family of planes defined by $\hat{n}_o$. In the position only mapping, the hand quaternion is replaced by a prefixed quaternion Q_o^U. This locks the arm to only translate. When the user releases a control for the gripper, the system uses a motion planning solver to find the joint states. If a solution is not found, the control object turns red to indicate the error. If a solution is found, the arm moves to the specified position and orientation. The current position and orientation of the robot's end-effectors are represented in the virtual space by arrow markers. Relative positioning of the arms to each other can be achieved by seeing the current location markers and commanded end points of both arms at the same time.

A user can plan movements based on the relative distance between the arm's current location marker and their hand while looking at the live display of the arm. A user is capable of selectively changing the focus of their attention between the whole system and a single arm by shifting their body, thereby increasing focus during a task. A user can turn towards a single side of the room and focus on a single arm's camera and range finder, while maintaining sight of the arm in 3D with their peripheral vision, allowing for simultaneous consuming multiple perspective on the state of the robot arm.

Producing an equivalent physical control room for robots would be prohibitively expensive. Aside from requiring dedicated space, it would require installing 3D displays, multiple monitors and a suite of robotics to move the reference markers, displays. Meanwhile a setup for a VRCR only requires a VR head set and a computer.

IV. System Implementation

For our particular system we used an Oculus Rift with Touch controllers, and a Baxter robot. For the slave side, we chose the Baxter robot for its commercial omnipresence, its humanoid appearance and its distinctly non-humanoid arm movements. Baxter provides a camera and range sensor in each arm. three configurations of the Baxter robot were used: two parallel plate grippers, and a gripper and a staple gun, and a hard gripper and a soft tube gripper. The stereo camera feed was generated by two Logitech C930e cameras were placed on Baxter's head. The staple gun configuration can be seen in Fig. 2(a).

The master side hardware was an Oculus Rift with touch controllers, pictured in Fig. 2(c), and a dedicated Oculus ready computer. The user wears the Oculus (Fig. 2(e)) and sees the virtual control room (Fig. 2(b)) to provide immersive VR. The

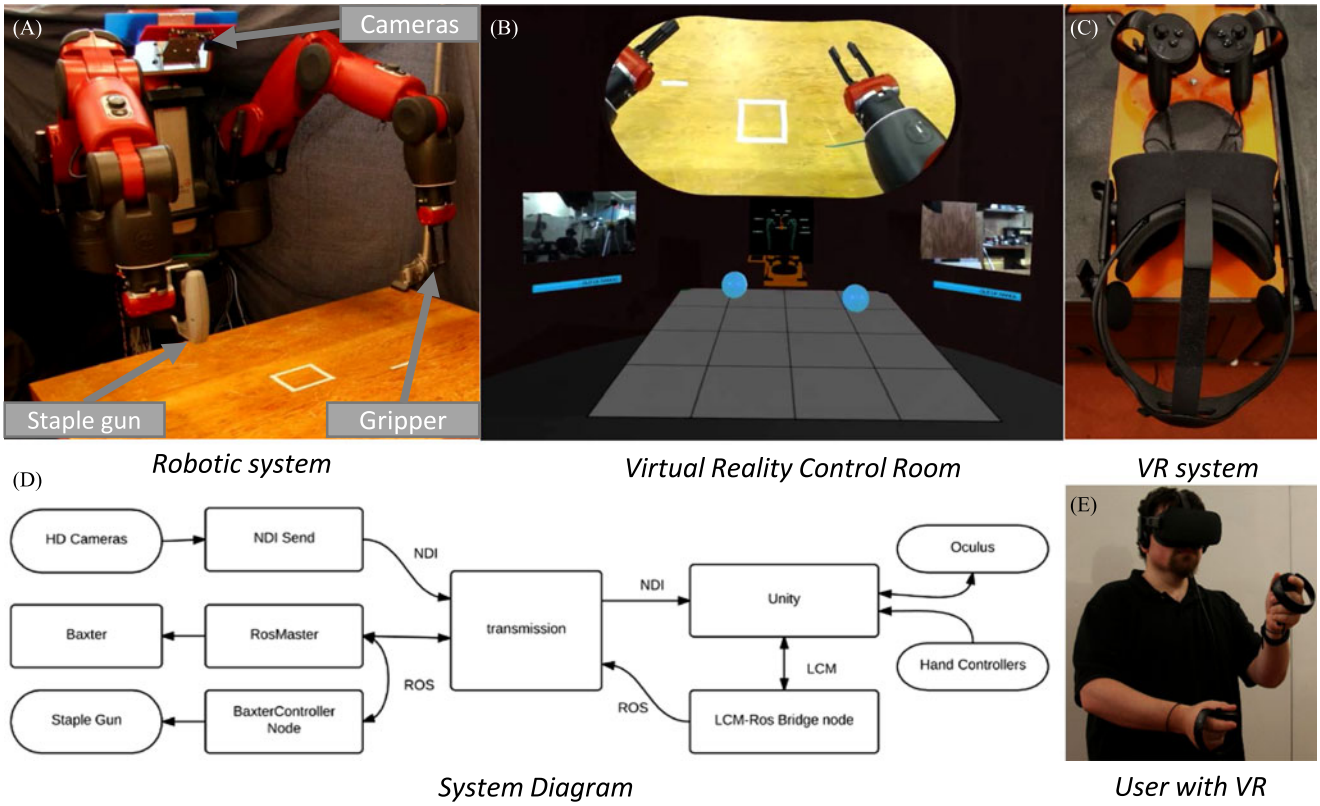

Fig. 2. Telerobotic system for Homunculus Model. A Baxter robot is outfitted with a stereo camera rig and various end effectors. In Unity, a virtual control room is established which uses an Oculus Rift CV1 headset and Razer Hydra hand trackers for inputs and displays. This allows the User to feel as if they are inside Baxter's head while operating it. Displays from each hand's sensors can be integrated into the control center. Only information from the cameras and commands, and state information need to be sent over the network. The transmission component can be non-existent, or a Canopy system for internetwork communications. The ROS nodes can be on the same machine, or different machines.

touch controllers can sense the grasp and finger locations of the user. The dedicated computer, represented by the unit box in Fig. 2(d), receives input on the user state from the Oculus and updates the Oculus display at a rate of 90 frames per second. The VRCR was configured with each hand camera feed displayed on a video screen and the range sensors displayed by a progress bar. This allowed a user to see through the hand camera, read the range sensor value, and see a stereoscopic view of an arm at the same time. A virtual grid space was placed in the VRCR to represent a table surface in front of the Baxter.

The user changes the mapping $\mathscr{M}_{HG}$ by changing how they held the controllers (See Algorithm 1). When the user grasps the controller with a fist, $\mathscr{M}_{HG}$ is position and single angle only. When the user grasps the controller while pointing their index finger, $\mathscr{M}_{HG}$ fully maps position and orientation. If the user grabbed the control while pressing a pre-assigned button, only the position would be transferred. By selecting a different transform, the user can switch from having the gripper match the hands position and orientation, to simple matching position, with orientation fixed into the downwards position. Users use pre-assigned buttons on the controller to command the hands to open and close, or a staple gun to fire.

The use of current commercial VR hardware presented a novel software architecture challenge. The Oculus Rift and other consumer VR head sets rely on Windows and can be easily developed for using C# and Unity. Since Baxter and a wide variety of other robots use ROS on Ubuntu, we needed a messaging system which could communicate between the systems. Lightweight Communications and Marshalling (LCM), developed by MIT and University of Michigan, proved a sufficient solution by being light-weight, cross-platform, and operating in

Algorithm 1: Control of Arm.

1: User selects type of model $\mathscr{M}_{HG}$ by positioning fingers
2: User grabs virtual gripper G_i^V with hand H_k^V
3: **if** TYPE($\mathscr{M}_{HG}$) is position and orientation **then**
4: $\quad \overrightarrow{r}_H^R \leftarrow \bar{T}_{RU}\,\overrightarrow{r}_H^U, Q_H^R \leftarrow Q_{RU} Q_H^U$
5: **else if** TYPE($\mathscr{M}_{HG}$) is position and single angle **then**
6: $\quad \overrightarrow{r}_H^R \leftarrow \bar{T}_{RU}\,\overrightarrow{r}_H^U, Q_H^R \leftarrow Q_{RU} Q^U(\hat{n}_o^U, \theta_H^U)$
7: **else if** TYPE($\mathscr{M}_{HG}$) is position only **then**
8: $\quad \overrightarrow{r}_H^R \leftarrow \bar{T}_{RU}\,\overrightarrow{r}_H^U, Q_H^R \leftarrow Q_{RU} Q_o^U$
9: **end if**
10: User releases G_i^V
11: Joints solution $A \leftarrow$ PLANNER($\overrightarrow{r}_H^R, Q_H^R$)
12: **if** A is valid **then**
13: $\quad \forall$ joints $J, J_i \leftarrow A_i$
14: $\quad \overrightarrow{r}_G^R \leftarrow \overrightarrow{r}_H^R$
15: $\quad Q_G^R \leftarrow Q_H^R$
16: **else**
17: $\quad$ COLOR($G_i^V, error$)
18: **end if**

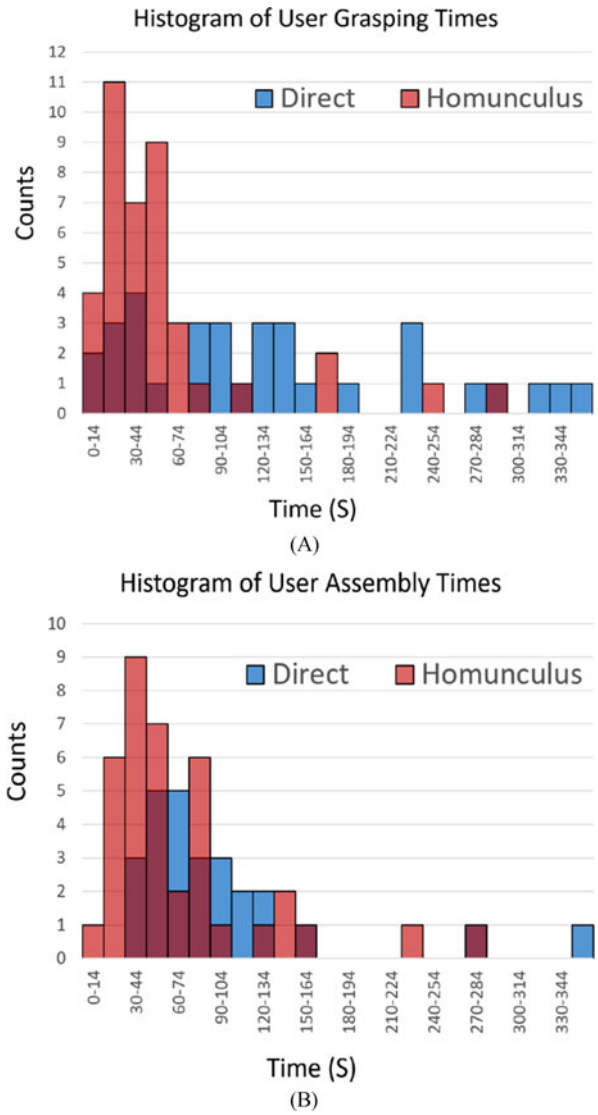

Fig. 3. User speeds for successful uses of the homunculus and mimicking systems. In (a) we see that the users were significantly faster with the homunculus system when compared with the mimicking system. In (b) we see evidence that users may be faster at assembly when using the homunculus system.

C# C++ and Python [18]. LCM Messages and ROS messages are converted into each other using a separate process on a ROS machine. For high definition video, the LCM/ROS infrastructure was insufficient because of the data rate required for high definition video. We used the NDI system by NewTek for transmitting and receiving a compressed HD video stream from the HD cameras. This architecture for the communication can be seen in Fig. 2(d). For this local network configuration requires 3 machines: a ROS computer, the dedicated Windows computer running unity for the Oculus, and a Windows computer operating the NDI software. The transmission element of Fig. 2(d) represents only the local router in this setup.

We expanded the system capability to include inter-network operations. For this we used the Canopy framework to relay ROS messages between different networks [19]. The Canopy framework subscribes to ROS topics, broadcasts them to server, and then transmits them to other ROS Master nodes that connect to the same server. This allows the same code that runs on the local network to operate across multiple networks by simply changing the channels the program subscribes to without changing network configurations. The robot side and the user side of the connection both call out to an intermediary service rather than having the user side know the address and ports for the robot network. It also avoids the problems of lost UDP packets over long network transmission between the user and robot networks by quickly converting them to ROS messages which are passed over TCP on the user's network. This is represented in Fig. 2(d) as the transmission element.

V. Results and Discussion

We conducted a user study where participants teleoperated the Baxter to perform an assembly task using either the homunculus system, or a direct system. The users would first grab a randomly placed block and then, stack it on top of another block, as seen in Fig. 4. We had the users receive a tutorial before using each system. They were timed in performing the task 5 times. If the user made an unrecoverable error, miss-aligned the blocks, or took longer than 6 minutes between each subtask, the attempt was recorded as a failure. The direct system was modeled on a system developed by UMass Lowell for use with a Baxter [10]. It fed only the images from the fixed cameras directly to the eyes of the user, and Baxter mimicked both the position and orientation of the user's hand. The same master side hardware was used for both systems. Both the homunculus system and direct system were connected over a local wired network, but the update rate of the cameras was set to 10 Hz, and Baxter would respond to position commands only once every 2 seconds in order to simulate limited system connectivity. We had eight users use the homunculus system and seven users for the mimicking system. The users had no prior experience with virtual reality or teleoperation.

As seen in Table I, the human users were more successful while using the homunculus system compared with the direct system. We performed a two proportion Z test, to determine if this is a significant result. We label the the probability of successfully grasping and then assembling a block $p(build)$, probability of successfully grasping the block $p(grasp)$, and the probability of successfully assembly given the block was already grasped as $p(assemble)$. With a score of $z = 2.58$, we can reject the null hypothesis that $p(build|H) = p(build|D)$, over the 95% confidence range. We can therefore state that the homunculus system does improve performance on grasping and then assembling. Similarly, with a Z score of 2.094, we can state with over 95% confidence that $p(grasp|H) \neq p(grasp|D)$. Along with being more successful, we can see in Table I and Fig. 3(a), that users who successfully grasped the blocks were consistently faster with the homunculus system. An ANOVA analysis of the grasping data gave F of 15.58, compared with $F_{critical}$ of 3.97, with a P value of 0.000183. Therefore we can definitively conclude that the homunculus system makes users faster at grasping objects from the table. However the z score is only 1.82, for $p(build|H) \neq p(build|D)$ so we only have an 93% confidence interval. Fig. 3(b) indicates that users of the homunculus system

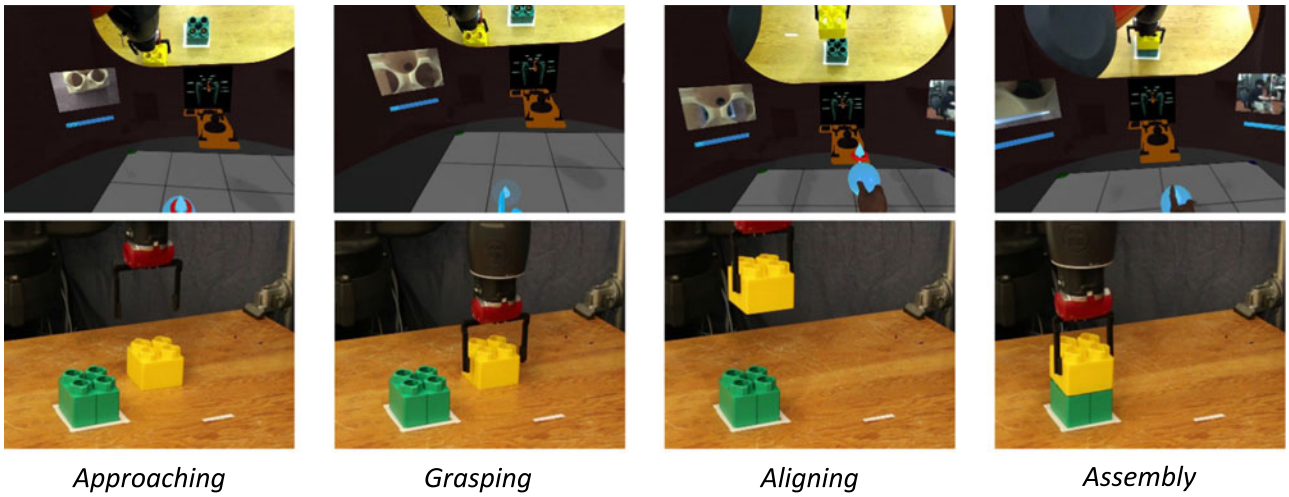

Fig. 4. Block Assembly. A series of blocks are grasped and stacked as a test of the system performance. The top row is the user view inside the VRCR, the bottom row is a view of the end effector manipulating the blocks.

TABLE I
RESULTS OF HOMUNCULUS VS DIRECT SYSTEM

	Homunculus	Direct Mimicking
Number of Trials	40	33
Number of successful builds	30	15
Number of successful grasps	38	26
Grasp success rate	95%	79%
Assembly after grasp success rate	79%	58%
Build success rate	75%	45%
Average time of successful grasp	57 s	133 s
Average time of successful assembly	70 s	94 s

were slightly faster than users with the mimicking system, however, an ANOVA analysis was inconclusive. We cannot show that a user who has successfully grasped an object, is faster or slower at assemble for either condition. This is in large part because the disparity in successful grasping limited the number of test of assembly with the mimicking system.

We believe that the ability to use the hand cameras, and the ability to vary $\mathscr{M}_{HG}$ aided the users in the grasping task with the Homunculus system. Baxter's parallel plate gripper seem to work best when grasping from above, and human hands are not typically used to grasp from above while standing. Varying $\mathscr{M}_{HG}$ allowed a user hand to be aligned horizontally while Baxter grasped from above, making the experience more comfortable than the mimicking system. The cameras in the Baxter's hands also aided the user in aligning the gripper with the target block, since they were able to directly judge distance and alignment from the hand, rather then relying on the point of view of the stereo cameras. Once the block was grasped, the hand camera could not be used for alignment when stacking. This would explain the wide disparity in performance on grasping and the narrow speed difference on assembly. Perhaps if more users had used the second hand as a positionable camera, the homunculus users would have been even faster. Researchers at MIT had Baxter perform the same assembly task with and without in-hand object localization (IOL) with the same hard parallel plate gripper [20]. They reported that with 100 trials there was a 100% success rate at grasping for both systems, a 44% success rate without IOL, and a 66% success rate with IOL. We use this as a comparison to determine the applicability human operation on the task. While vision based grasping algorithms were reported to be perfectly successful at grasping, the human users with the homunculus had a higher success rate for assembly tasks than either autonomous system. With a z score of 3.25, we are confident that $p(assemble|H) \neq p(assemble|withoutIOL)$. However with a z score of 1.47 we only have an 85% confidence interval over the system with In-Hand Object Localization. Given our competitiveness with automation, we believe this is a viable test to benchmark the system against.

For additional demonstrations of the system, we devised a series of tasks for the lead author of the letter to perform. We focused on fixture-less assembly, pick-and-place, and manufacturing tasks for the system. Fixture-less assembly seen in Fig. 5 is a variant of the stacking task. The user grabs an object in one hand, passes it to the other and then grabs a second block and inserts it onto the first block. Here the user switches between two different $\mathscr{M}_{HG}$ to complete a fixture-less assembly task. When grasping items from the table, the rotation information is fixed, but when assembling in free space or handing the items off between hands, the rotation of the users hands are directly mapped to the robots.

For pick-and-place operations, we tested the system's ability to handle objects of different shape and compliance. As seen in Fig. 6(a), the user was tasked with picking up each item, transferring them from the left hand to the right hand, and then depositing them into a bin. This required the user to identify locations to grasp the object from, align the hands, re-grasp, and transfer to the bin. This is a non-trivial set of operations for a robotic system since it must be able to predict the movements of the cloth in response to grasping, understand the presence of the overhangs on the reel, and plan to drag the reel into a position where it can be grasped from the side. By contrast, a user is able to quickly execute these tasks in the teleoperation framework by leveraging their intuition for the objects' response to stimuli. In the supplementary video we see that an expert user is able

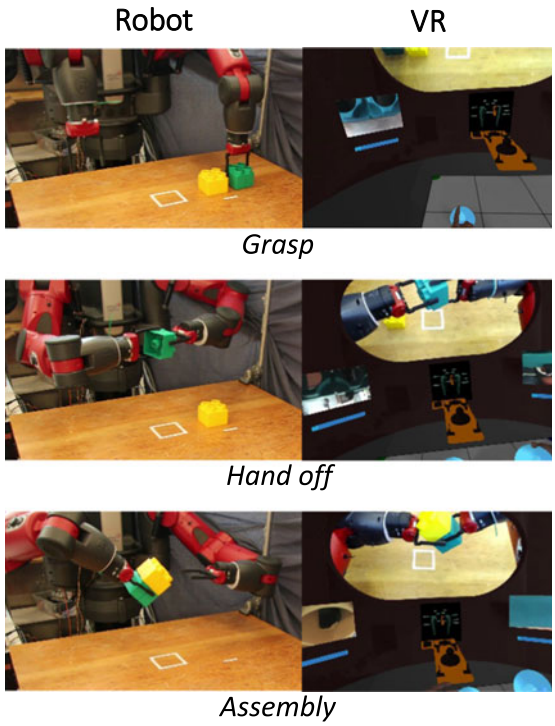

Fig. 5. Fixture-less Assembly by Teleoperation. A user is able to pick up a block, hand it off to the other hand, pick up a second block and stack them together. This requires the use of multiple $\mathscr{M}_{HG}$ to complete.

to use the hand cameras and virtual window to pick and place screws into a bin and lift and place a wire.

For manufacturing a task, we had the user staple a wire to a board. A staple gun was mounted to the side of the left arm of Baxter, as seen in Fig. 2(a). Wires were placed on a table, next to a wooden board. The user needed to be able to pick and place the wire onto the board with one arm, place a staple gun and fire the gun several times to lock the wire in place. As seen in Fig. 6(b), the user was able to perform the task and staple the wire in place. This tested the relative utility of the hand cameras, as the staple gun was attached to the side of the robot arm as an end-effector, preventing the use of the hand camera when aligning the staple gun. The user was able to pick up the wire while using the right hand camera, but without the aide of the left hand camera, because the alignment of the staple gun was inaccurate. The user needed 5 attempts to secure 2 staples. Future version of the system could include a pressure sensor and display to allow the user to know the force they are applying to the surface as well as a map of the pressure over the contact surface. This would allow the user to be sure the tool is lying evenly with enough force on the surface. As an additional task, we had the user place two soft rubber tubes through a vertical hole in a plastic rib. As seen in Fig. 6(c), the user used a soft gripper to grasp a tube and place it through the hole, and a hard gripper to pull it through the hole. This is similar to the task for installing ducking and wiring in large scale structures such as planes and buildings.

In order to test systems ability over multiple network architectures, we controlled the robot in several different network configurations. The tests above were done on a local network using a wired connection. We also controlled the robot in the

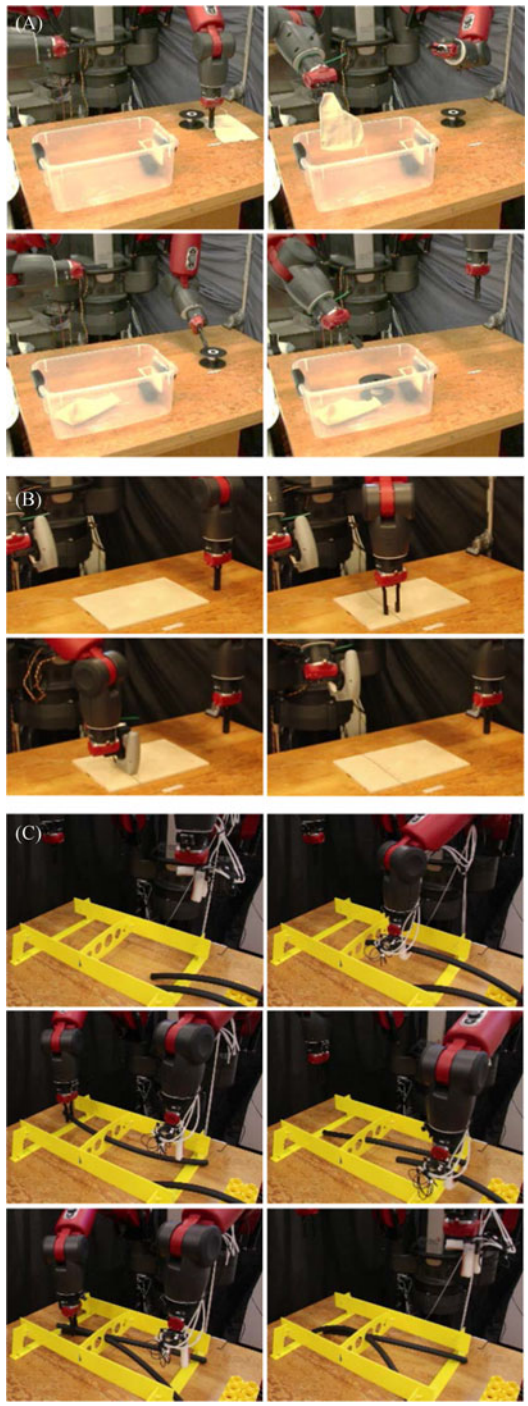

Fig. 6. Demonstrations of system capabilities. (a) A user grasps objects such as tubes, reels, and cloth. the user then passes it between the hands and into the bucket. (b) A wire is grasped with one arm, placed on a wooden board, and stapled in place with the other arm. (c) Using a soft gripper the user grasps and places two rubber tubes through a single hole, and pulls them through with a hard gripper.

same building over different networks via a wireless connection This required decreasing the update rate on the hand cameras to 5 Hz. As a test of long range communication, the system was set up at the Hyatt Regency Crystal City in Arlington, VA for the 2016 PI meeting of the National Robotics Initiative. In this condition there was a limit of 5 Mbps on the connection. to meet

this requirement, we limited the frame rate on the hand cameras to 1 Hz and the range update to 5 Hz and reducing the resolution of the stereo cameras to 800 px $\times$ 600 px. At the NRI PI meeting users were able to pilot a Baxter at MIT over the hotel's wireless Internet. Users were given visual movement queues at 90 Hz from the VRCR, so a frozen camera image looked simply like a 3D or 2D image on a wall. If this were a direct system, where the VR update rate was at 1 Hz, people would likely have gotten sick due to the slow update.

VI. Conclusion

In this letter we developed a system for VR enabled telerobotics based on the homunculus model of mind. The system is highly flexible across network architectures, bandwidth allotments, and tasks. By being able to rapidly change the mappings between human inputs and robot state, we can improve utility and ergonomics. We demonstrate our framework with existing consumer grade hardware and software and robot systems. This provides it an ability to scale into wider deployment. Teleoperated robotic systems will allow humans the ability to work at scales and in environments which they cannot accomplish today.

Acknowledgment

The authors would like to thank C. Choi for his assistance with the Baxter system and for sharing his data on assembly.

References

[1] R. L. Gregory and O. L. Zangwill, *The Oxford Companion to the Mind.* London, U.K.: Oxford Univ. Press, 1987.

[2] D. C. Dennett and M. Kinsbourne, "Time and the observer: The where and when of consciousness in the brain," *Behav. Brain Sci.*, vol. 15, no. 2, pp. 183–201, 1992.

[3] M. R. Mine, F. P. Brooks, Jr., and C. H. Sequin, "Moving objects in space: Exploiting proprioception in virtual-environment interaction," in *Proc. 24th Annu. Conf. Comput. Graph. Interact. Tech*. ACM Press/Addison-Wesley, 1997, pp. 19–26.

[4] X. Wang, C. Yang, H. Ma, and L. Cheng, "Shared control for teleoperation enhanced by autonomous obstacle avoidance of robot manipulator," in *Proc. 2015 IEEE/RSJ Int. Conf. Intell. Robots Syst.* IEEE, 2015, pp. 4575–4580.

[5] R. S. Kennedy and K. M. Stanney, "Postural instability induced by virtual reality exposure: Development of a certification protocol," *Int. J. Human-Comput. Interact.*, vol. 8, no. 1, pp. 25–47, 1996.

[6] E. Ackerman, "Oculus rift-based system brings true immersion to telepresence robots," *IEEE Spectrum*, 2015. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/robotics-hardware/upenn-dor a-platform

[7] M. Borg, S. S. Johansen, K. S. Krog, D. L. Thomsen, and M. Kraus, "Using a graphics turing test to evaluate the effect of frame rate and motion blur on telepresence of animated objects," in *Proc. 8th Int. Conf. Comput. Graph. Theory Appl.*, 2013, pp. 283–287.

[8] M. Stilman, K. Nishiwaki, and S. Kagami, "Humanoid teleoperation for whole body manipulation," in *Proc. IEEE Int. Conf. Robot. Autom*. IEEE, 2008, pp. 3175–3180.

[9] T. Rodehutskors, M. Schwarz, and S. Behnke, "Intuitive bimanual telemanipulation under communication restrictions by immersive 3D visualization and motion tracking," in *Proc. 2015 IEEE-RAS 15th Int. Conf. Humanoid Robots*. IEEE, 2015, pp. 276–283.

[10] U. L. R. Lab, "Baxter hydra rift teleop," Aug. 2014. [Online]. Available: http://sdk.rethinkrobotics.com/wiki/Baxter_Hydra_Rift_Teleop

[11] J. Kofman, X. Wu, T. J. Luu, and S. Verma, "Teleoperation of a robot manipulator using a vision-based human-robot interface," *IEEE Trans. Ind. Electron.*, vol. 52, no. 5, pp. 1206–1219, Oct. 2005.

[12] L. Fritsche, F. Unverzag, J. Peters, and R. Calandra, "First-person teleoperation of a humanoid robot," in *Proc. 2015 IEEE-RAS 15th Int. Conf. Humanoid Robots*. IEEE, 2015, pp. 997–1002.

[13] H. Reddivari, C. Yang, Z. Ju, P. Liang, Z. Li, and B. Xu, "Teleoperation control of Baxter robot using body motion tracking," in *Proc. 2014 Int. Conf. Multisensor Fusion Inf. Integr. Intell. Syst*. IEEE, 2014, pp. 1–6.

[14] P. Kremer *et al.*, "Multimodal telepresent control of DLR's Rollin'JUSTIN," in *Proc. IEEE Int. Conf. Robot. Autom*. IEEE, 2009, pp. 1601–1602.

[15] C. Passenberg, A. Peer, and M. Buss, "A survey of environment-, operator-, and task-adapted controllers for teleoperation systems," *Mechatronics*, vol. 20, no. 7, pp. 787–801, 2010.

[16] K. R. Guerin, S. D. Riedel, J. Bohren, and G. D. Hager, "Adjutant: A framework for flexible human-machine collaborative systems," in *Proc. 2014 IEEE/RSJ Int. Conf. Intell. Robots Syst*. IEEE, 2014, pp. 1392–1399.

[17] M. Antonov, "Asynchronous timewarp examined," Mar. 2015. [Online]. Available: https://developer3.oculus.com/blog/asynchronous-timewarp-examined/

[18] A. S. Huang, E. Olson, and D. C. Moore, "LCM: Lightweight communications and marshalling," in *Proc. 2010 IEEE/RSJ Int. Conf. Intell. Robots Syst*. IEEE, 2010, pp. 4057–4062.

[19] R. C. A. Wallar, "Canopy: The ROS cloud robotics framework and cloud computing platform service," 2017. [Online]. Available: https://github.com/canopy-ros

[20] C. Choi, J. DelPreto, and D. Rus, "Using vision for pre- and post-grasping object localization for soft hands," in *Proc. Int. Symp. Exp. Robot.*, 2016, pp. 601–612.