Toward robot teleoperation with virtual reality: remote scene exploration and object grasping

Bukeikhan Omarali[1,2] Brice Denoun[1], Kaspar Althoefer[1], Lorenzo Jamone[1], Maurizio Valle[2]
Ildar Farkhatdinov[1,3]
b.omarali,i.farkhatdinov@qmul.ac.uk;maurizio.valle@unige.it
[1]Centre for Advanced Robotics, Centre for Intelligent Sensing,
School of Electronic Engineering and Computer Science, Queen Mary University of London, London, UK
[2]Cosmic Lab, The Electrical, Electronics and Telecommunication Engineering and Naval Architecture Department,
University of Genoa, Genoa, Italy
[3]Department of Bioengineering, Imperial College of Science, Technology and Medicine, London, UK

ABSTRACT

The state-of-the-art robot teleoperation through Virtual Reality relies on setups that use external static RGB-D camera(s) to reconstruct the remote environment. The resultant point cloud of objects of interest might be distorted depending on the lighting conditions or occluded by other objects and the robot. This complicates the grasping and manipulation of these objects as the operator cannot refine the visualization of distorted and occluded parts. We suggest that mounting the RGB-D camera on the robot allows the operator to visualize and understand the remote scene better, resulting in higher teleoperation performance. Since the in-hand camera is likely to be closer to the area of interest than an external static camera we supplement the visualization with the OctoMap to improve the overview of the remote scene. We encourage readers to view the video demo at: https://youtu.be/3vZaEykMS_E.

1 INTRODUCTION

As consumer-grade Virtual Reality (VR) technologies have become more accessible in the last decade, more researchers have started designing robot teleoperation interfaces around them. VR enables more immersive and intuitive human-robot interaction making remote task completion more efficient and safe. Currently applications of VR teleoperation include: disaster relief [14], surgery [1], exploration [16], and learning by demonstration [18].

VR teleoperation interfaces work efficiently when used with RGB-D cameras in the slave site providing point cloud representation of the robot's workspace [8, 10, 11]. The VR headset provides the operator with the depth perception through binocular vision. This is commonly achieved by rendering the point cloud for each eye separately.

The majority of VR teleoperation systems use a single external static RGB-D camera directed at the robot's workspace [6, 17, 18]. In such systems the teleoperation performance suffers from incomplete and imperfect visual reconstruction of the remote environment. Depending on objects' reflectivity, geometry and overall lighting conditions some objects may appear distorted in the point cloud - we advise the reader to consult the [2] for an overview. Furthermore all objects are partially self-occluded (since they are only seen from one side) and some objects can be occluded by other objects in a clutter or by the robot.

One solution to these problems is to mount an additional RGB camera onto the robot's end-effector (EE). Whitney et al, [17] used RGB camera mounted on EE to view objects that are distorted or occluded by the robot. The video stream was rendered on a virtual surface attached to the virtual representation of the robot. In this solution the grasp would have to be performed relying on the video stream rather than the point cloud, which reduces the benefits of the VR. Alternatively, Kohn et al [5] used multiple static RGB-D cameras to reconstruct the robot's workspace with less potential occlusions. Although it greatly reduces potential occlusions it does not eliminate them altogether. Furthermore having multiple external cameras is not always practically feasible.

In this workshop submission, we extend existing VR based teleoperation interfaces and propose a VR teleoperation framework with dynamic field-of-view control. We argue that in-hand RGB-D combined with OctoMap can allow the operator to explore the remote scene more efficiently compared to a single static external RGB-D camera. It should be noted that EE mounted RGB-D camera might not be able to register the point cloud of the grasp if it occurs very close to the camera. Therefore we propose a few grasping methods suitable for our framework. Details of the proposed VR-based teleoperation framework can be found in section 2. Sections 3 presents the experimental used to evaluate proposed methods. Conclusions and future work are discussed in Section 4.

2 PROPOSED FRAMEWORK

We propose a few augmentations to existing VR teleoperation frameworks that allow the operator to: 1) dynamically control of the operator's field of view with an RGB-D camera mounted on the robot's EE; 2) manipulate the virtual representation of the remote environment using gestures; 3) grasp and manipulate objects in the remote environment both through direct teleoperation and offline grasp-generation.

2.1 Dynamic field-of-view control

We provide the operator with the ability to control the field-of-view of the remote camera to reduce visual occlusions and distortions and improve the operator's situational awareness. We suggest that the RGB-D camera should be mounted on a controllable link close to the EE of the slave robot. Although EE mounted RGB visual feedback was used in various robotics applications [7, 13, 17], to our knowledge, there has been no research that use EE mounted RGB-D cameras as a part of the VR teleoperation interface. Hence we generate both the video stream that is viewed in a dedicated

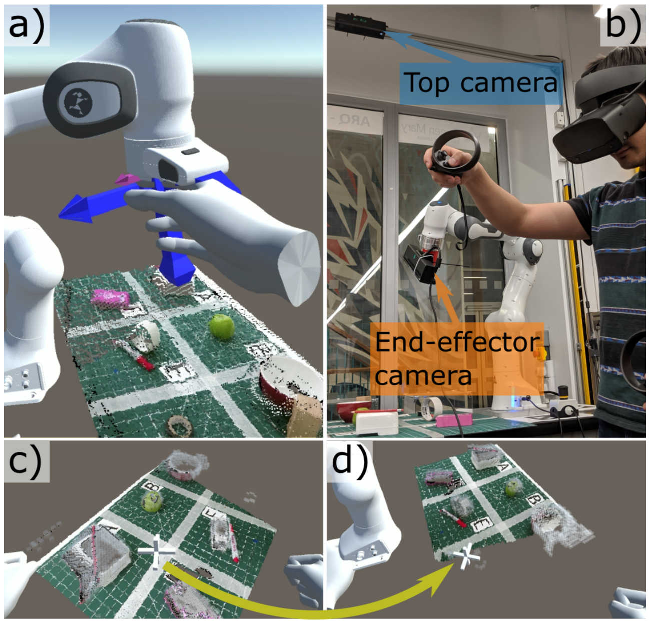

Figure 1: a) Robot is directly teleoperated in VR by dragging the end-effector (double camera mode); b) The experimental setup for the visualization study; c) and d) operator translates, rotates and scales the virtual representation of the robot's workspace with gestures (end-effector camera with OctoMap mode).

plane in the virtual environment and the point cloud from the same dynamic camera.

The major differences between the EE mounted camera and external cameras are the level of detail and the size of the captured area. External cameras are usually placed to maximize the overview of the remote scene, resulting in a large but low detailed reconstruction. By comparison the EE camera provides a more detailed view of a smaller area. We suggest that a mapping techniques such as OctoMap [3] can be used to continuously generate a lightweight occupancy map of the robot's workspace to compensate for the lack of overview in EE camera mode. In VR occupied nodes are displayed as transparent boxes, layered over the point cloud, see Fig. 1.{c,d}. The map is generated continuously as the operator explores the remote environment.

2.2 Gestures-based manipulation of the virtual scene

We propose a gesture-based human-VR-robot interface for more efficient remote environment exploration. The interface relies on tracking the operator's hands and use their relative position and orientation to manipulate the position, orientation and the scale of the virtual representation of the robot's workspace. This helps the operator to navigate the virtual environment, if the operator is uncomfortable with physically moving in VR (walking or crouching) or prefers to operate sitting (to reduce physical exertion).

Gestures are inspired by VR 3D drawing tools like Tilt Brush[1]. For example, to rotate the scene the operator should press both buttons of the left and right handheld wireless controllers and perform rotation as if he/she is rotating a physical steering wheel. In this case, the center of rotation is fixed to the middle point between the operators' left and right hands. Translating and scaling is implemented similarly. Pulling and pushing the center of rotation will pull and push all objects in the scene. Bringing arms closer or further apart will zoom in or zoom out the area of interest. Translation, rotation and scaling interfaces can operate simultaneously. The gesture-based manipulation of the virtual scene is illustrated in Fig. 1{c,d}.

2.3 Grasping methods for robotics setups with end-effector mounted RGB-D camera

The grasping with EE mounted RGB-D camera is more complicated compared to grasping with an external camera, since RGB-D cameras require a minimum distance to an object to register it as a point cloud. If the EE mounted camera is too close to the gripper the operator will have to grasp nearly blind. We propose three solutions that vary in the operator's involvement in the process, illustrated in Table 1. We encourage the readers to view the video demo at: https://youtu.be/3vZaEykMS_E.

2.3.1 Direct grasping. This solution is designed for direct teleoperation in which the operator drags the virtual EE to control the robot in real-time. The core idea for this solution is to perform the grasp on a persistent segmented clone of the object of interest rather than on the "live" point cloud. We perform a naive segmentation by separate OctoMap chunks - i.e. all occupied contiguous OctoMap nodes - to isolate the object. The operator indicates the object of interest by pointing at it with a ray that extends from the operator's hand, illustrated in Fig. 2.a. All points contained in the isolated OctoMap chunk are then segmented and cloned to persistent local memory, see Fig. 2.b If the partial clone of the object is insufficient the operator can reposition the camera and rescan adding more points to the existing clone, as shown in Fig. 2.c,d,e. As the operator approaches for the grasp, the "live" point cloud of the object disappears as it is too close to the camera, but the clone persists and can be grasped, see Fig. 2.f.

2.3.2 Supervised grasping: grasp on pose. Our second solution is to plan the grasp on the "live" point cloud offline. It is arguably less physically demanding as the operator does not have to manually drag the robot to the grasp pose. In Fig. 3.a the operator specifies the position of the grasp on the "live" point cloud. The desired grasp pose is then sent to the slave controller which proposes the grasp

[1]https://www.tiltbrush.com

Table 1: Task distributions across grasping methods

	Grasp object	Grasp pose	Grasp trajectory
Direct	Operator	Operator	Operator
Grasp pose	Operator	Operator	Robot
Point and click	Operator	Robot	Robot

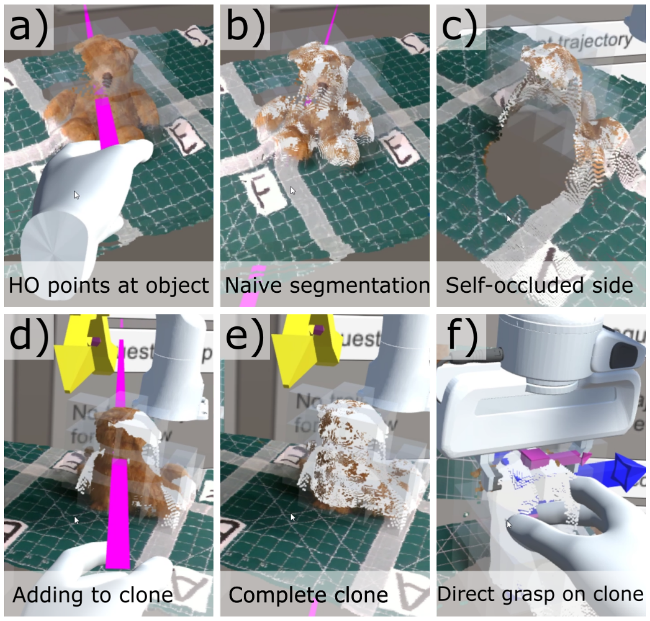

Figure 2: Naive segmentation and cloning of the object of interest's point cloud for direct grasp: a) human operator (HO) requests the segmentation; b) the segmented clone is created (white); c) the self-occluded part of the clone; d) operator moves the camera to add missing points; e) complete segmented clone; f) operator drags the robot to the grasp pose;

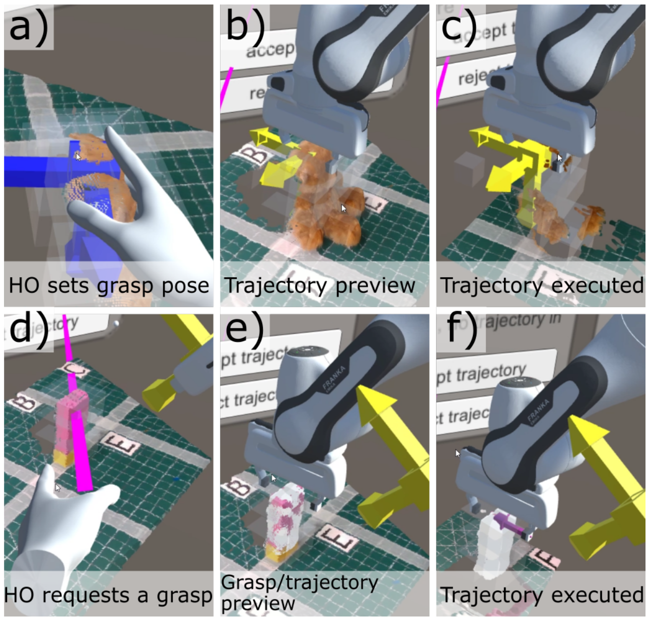

Figure 3: Supervised grasping on pose and point and click grasping: g) operator specifies the grasp position; h) robot proposes a grasp trajectory; i) robot executes the trajectory; j) the operator requests a grasp on the object; k) robot proposes grasp pose and trajectory; l) the robot executes the trajectory.

trajectory previewed in Fig. 3.b. Finally the operator decides to reject or accept the trajectory, shown in Fig. 3.c.

2.3.3 Supervised grasping: point and click. In our third solution we propose to use automated grasp methods to further reduce the teleoperation task load. The idea is only to point at an object of interest and let the robot propose the grasp pose and the trajectory. We use the same pointing and segmentation method as in direct grasping, shown in Fig. 3.d except that the segmented point cloud is then fed to the grasp pose generator. The resulting grasp pose is then passed to the motion planner similar to the previous supervised method. The operator previews the proposed grasp pose and trajectory, see 3.e, and rejects/accepts them, see 3.f.

3 FRAMEWORK IMPLEMENTATION

The experimental setup consisted of: the Franka Emika's Panda robot, two Microsoft Kinect2 RGBD cameras (for the visualization demonstration), Intel d415i RGB-D camera (for the grasping demonstration), Oculus Rift VR headset with Oculus Touch controllers, master PC, slave PC and a local Ethernet network. The general view of the setup for the visualization demonstration is shown in Fig. 1.b and its schematic diagram is shown in Fig. 4. The first Kinect2 camera was placed two meters above the robot. The second Kinect2 or Intel d415i were attached to the robot's EE and pointed along the EE.

We ran the ROS-bridge on the slave PC to publish ROS messages to a WebSocket and ROS-sharp on the master to read them. We separated the point cloud into a dedicated UDP channel outside the ROS-bridge. In our tests the dedicated UDP channel proved to be faster than the ROS-bridge. The dedicated UDP channel only streamed the XYZRGB part of the Pointcloud2 message.

In the visualization demonstration we controlled the robot in a "gantry mode", similar to [12, 17]. The operator moved the robot by dragging and rotating the virtual EE gizmo (an axes mesh attached to the virtual robot's EE), see Fig. 1.b. The gizmo's change of pose was then published in the Interoperable Teleoperation Protocol (ITP) format, see [4],[9]. The slave controller ran the Panda robot in the Cartesian impedance mode. A Proportional-Differential (PD) controller created a desired Cartesian force and torque necessary to match the current and the desired EE poses. The desired Cartesian force and torque were then converted into joint torques using the Jacobian pseudo-inverse. The gains of the PD-controller for the robot were adjusted to achieve a critically-damped response (response time of approximately 1 second). This control mode is also used for the direct grasping.

Furthermore we added a supervised control mode (for "grasp on pose" and "point and click" grasping) that controlled the robot using the *MoveIt* with RRTConnect planner. We define three types of motion requests in the corresponding master GUI: "move to pose", "move to grasp pose", "propose grasp on point cloud and move". The "move to pose" generates a trajectory to a desired EE pose. The "move to grasp pose" is similar to "move to pose" except it

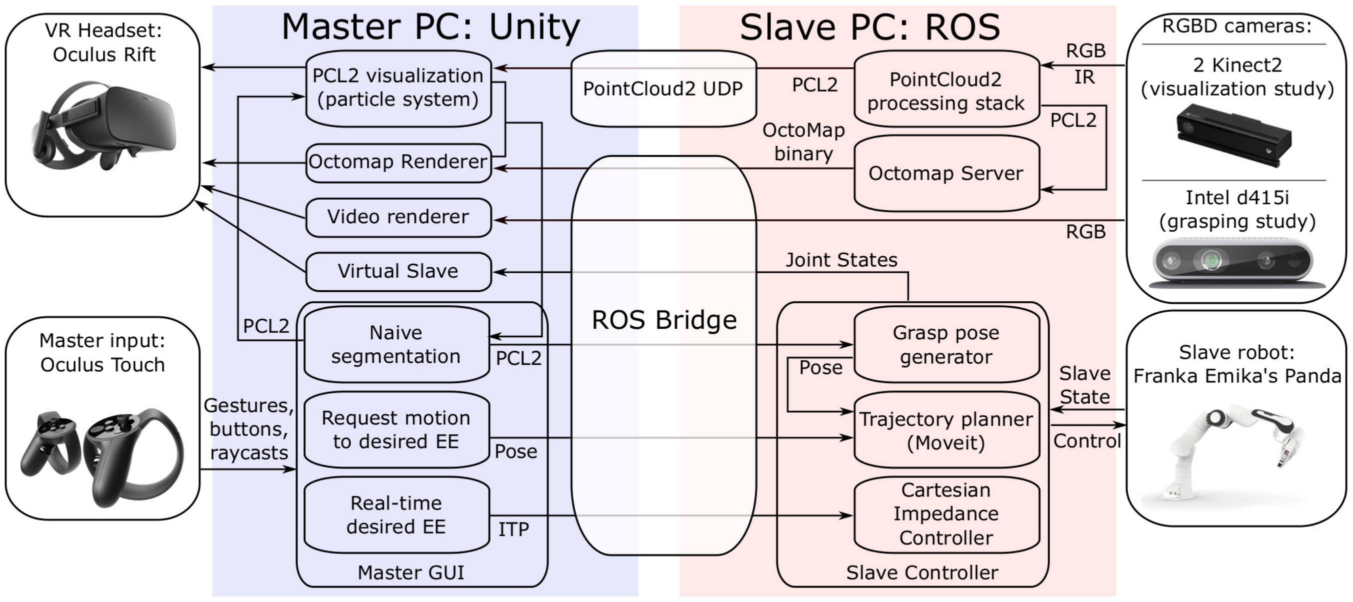

Figure 4: The schematic diagram of the experimental setup.

queues an additional pre-grasp pose that is offset from the grasp pose along the grasp's z-axis. The "propose grasp on point cloud and move" uses the point cloud clone of the object of interest to generate a grasp pose using the Principal Component Analysis [15]. The generated grasp pose is then processed as in "move to grasp pose".

The virtual representation of the robot was added to the Unity using the ROS-sharp's URDF import feature and animated using either the actual robot's joint angle values (default setting) or trajectory proposed by *MoveIt*. The Master GUI is set such that whenever a trajectory message is received the virtual robot visualization is toggled into preview mode. We animate the preview trajectory state by state rather than a smooth animation. The virtual robot animation is toggled back to real robot if the operator rejects or accepts/ executes the trajectory.

Both Kinect2 point clouds were generated using the standard definition - 512X424, the Intel d415i - 640X480. We ran the XYZRGB registration on the slave PC. This simplified the point cloud and OctoMap generation compared to registering on master [17], although it does use more bandwidth. To reduce the bandwidth consumption we cropped the point cloud to the area of interest - 0.9m X 0.6m X 0.3m and removed all points on which registration failed. Furthermore we repacked original point clouds to 15 bytes per point (standard Kinect2 registration uses 32 bytes, Intel d415i - 24 bytes).

We visualized the point cloud using the Unity's particle system. Particles were animated and colored based on received point cloud data. The segmented clone is also represented using a particle system. We kept the segmented clone white to make it visible when layered over the "live" point cloud.

We used the standard ROS implementation of OctoMap at and sent the map to the master in a binary format. The OctoMap was produced with the same point cloud that was sent to the master in the EE frame. Note that we only detected objects that are above the desk. For the visualization study the OctoMap resolution was set to 0.02m. We designed a custom parser and renderer for the binary OctoMap, that inherits from the ROS-sharp's subscriber class. The renderer only visualizes occupied OctoMap nodes as transparent boxes. We did not visualize or made a distinction between free and unknown nodes since the operator can infer that information himself/herself. We also included the RGB video stream for the grasping demonstration. The video stream was used to monitor the success of the grasp. We did not include the video stream to the visualization demonstration as we were interested in the operator's ability to understand the remote scene from the point cloud only.

4 CONCLUSION AND FUTURE WORK

We have proposed a Virtual Reality (VR) teleoperation framework with a dynamic field of view control and corresponding grasping methods. Our demonstration, see Fig. 5 have shown that compared to external static RGB-D camera the end-effector (EE) mounted RGB-D camera presents a more detailed visual feedback to the user that suffers less from occlusions and point cloud distortions. By including the OctoMap mapping we provide the operator with an overview of the remote environment similar to additional external RGB-D camera at much lower communication bandwidth consumption compared to the dual-camera setup.

The grasping with EE camera is more complicated compared to grasping with an external camera, since RGB-D cameras require a minimum distance to object for it to be registrable as a point cloud. If the EE mounted camera is too close to the gripper the operator will have to grasp in blind. We propose three solutions to this problem that vary in the level of the operator's involvement. In the first solution we segment the point cloud of the object of interest and keep it in the persistent memory, such that the grasp

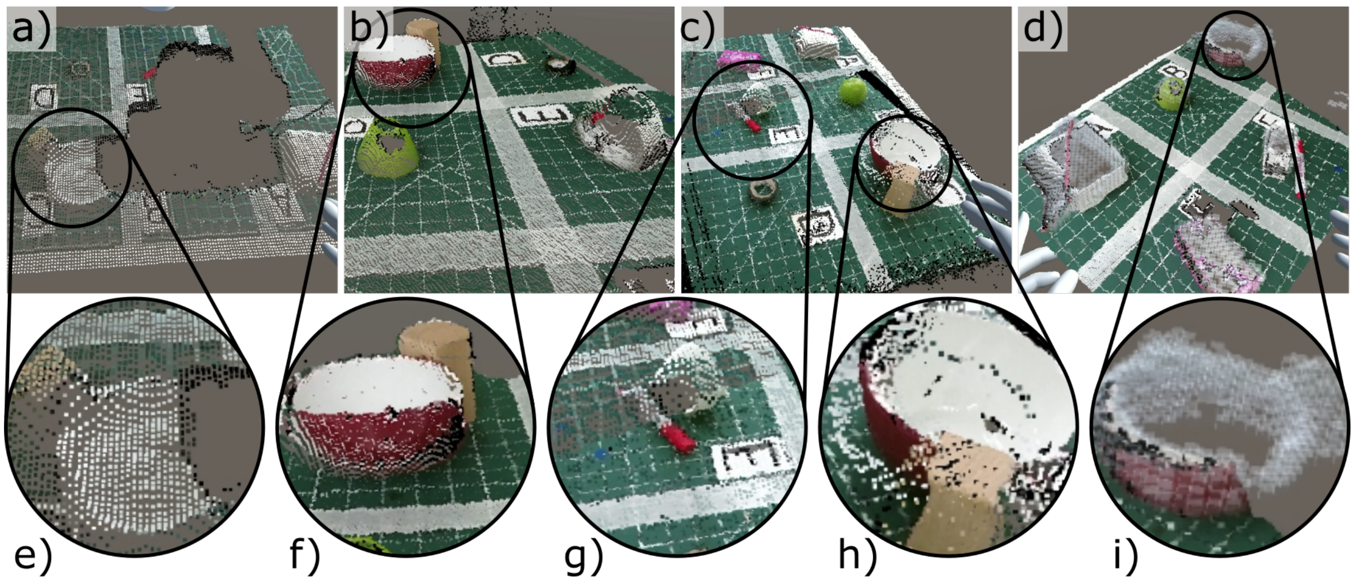

Figure 5: Operator's views in different modes: a) and e) top camera mode - note how the robot occludes a part of the view and the bowl shape is deformed; b) and f) end-effector (EE) camera view - the same bowl is much more recognizable; c) and h) double camera mode - the EE camera provides a better representation of the bowl, compared to the top camera g) the edge between the EE camera view and top-camera view, notice the difference in resolution; d) and i) EE camera mode with OctoMap - although the bowl is no longer in the field of view, the OctoMap maintains the accurate geometrical representation.

can be performed on the segmented clone, rather than on the "live" point cloud. In the second solution we plan the grasp on the "live" point cloud, by specifying the desired grasp pose, which is used to generate and preview the trajectory offline before the execution. Our third solution builds on previous ones - instead of manually selecting the grasp pose, we use automated grasp generation on the segmented point cloud of the object of interest. This reduces the physical load on the operator as he only requires to point on the object of interest to grasp it.

There are a few challenges that need to be addressed in future work. Although all three proposed grasping methods can bring the robot to the grasp position the success of the grasp can only be viewed from the video stream. It is not always obvious if the grasp was successful. We are planning to add tactile sensors to monitor the grasp. Manipulation of the grasped object needs to be addressed as well, as it also cannot be viewed as a point cloud. Two potential solutions to this problem are: use the segmented clone as a stand-in for the object in grasp; move the RGB-D camera further from the gripper such that the object can be viewed in grasp. Both solutions come with certain disadvantages - the segmented clone might not accurately represent the grasped object, especially if the object is not rigid, meanwhile moving the camera farther from the gripper might complicate the online trajectory generation as it needs to be included into the Inverse Kinematics problem as a self-collision obstacle.

5 ACKNOWLEDGEMENTS

This work was funded by EPSRC NCNR EP/R02572X/1, MAN^3, EP/S00453X/1, and QMUL-Genoa PhD program.

REFERENCES

[1] Mark Draelos, Brenton Keller, Cynthia Toth, Anthony Kuo, Kris Hauser, and Joseph Izatt. 2017. Teleoperating robots from arbitrary viewpoints in surgical contexts. *IEEE International Conference on Intelligent Robots and Systems* 2017-Septe (2017), 2549–2555. https://doi.org/10.1109/IROS.2017.8206076

[2] Yu He and Shengyong Chen. 2019. Recent Advances in 3D Data Acquisition and Processing by Time-of-Flight Camera. *IEEE Access* 7 (2019), 12495–12510. https://doi.org/10.1109/ACCESS.2019.2891693

[3] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. 2013. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots* 34, 3 (2013), 189–206. https://doi.org/10.1007/s10514-012-9321-0

[4] H Hawkeye King, Blake Hannaford, Ka-Wai Kwok, Guang-Zhong Yang, Paul Griffiths, Allison Okamura, Ildar Farkhatdinov, Jee-Hwan Ryu, Ganesh Sankaranarayanan, Venkata Arikatla, Kotaro Tadano, Kenji Kawashima, Angelika Peer, Thomas Schauss, Martin Buss, Levi Miller, Daniel Glozman, Jacob Rosen, and Thomas Low. 2010. Plugfest 2009: Global interoperability in Telerobotics and telemedicine. In *2010 IEEE International Conference on Robotics and Automation*. IEEE, 1733–1738. https://doi.org/10.1109/ROBOT.2010.5509422

[5] Sebastian Kohn, Andreas Blank, David Puljiz, Lothar Zenkel, Oswald Bieber, Bjorn Hein, and Jorg Franke. 2018. Towards a Real-Time Environment Reconstruction for VR-Based Teleoperation Through Model Segmentation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1–9. https://doi.org/10.1109/IROS.2018.8594053

[6] Donghyeon Lee and Young Soo Park. 2018. Implementation of Augmented Teleoperation System Based on Robot Operating System (ROS). In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vol. 2018-Janua. IEEE, 5497–5502. https://doi.org/10.1109/IROS.2018.8594482

[7] Jeffrey I Lipton, Aidan J Fay, and Daniela Rus. 2018. Baxter's Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing. *IEEE Robotics and Automation Letters* 3, 1 (jan 2018), 179–186. https://doi.org/10.1109/LRA.2017.2737046 arXiv:1703.01270

[8] John O Neill, Sebastien Ourselin, Tom Vercauteren, Lyndon Cruz, and Christos Bergeles. [n.d.]. Virtual Reality to Enhance Surgical Robot Development and Control. ([n. d.]).

[9] Bukeikhan Omarali, Francesca Palermo, Maurizio Valle, Stefan Poslad, Kaspar Althoefer, and Ildar Farkhatdinov. 2019. Position and Velocity Control for Telemanipulation with Interoperability Protocol. 316–324. https://doi.org/10.1007/978-3-030-23807-0_26

[10] Luis Pérez, Eduardo Diez, Rubén Usamentiaga, and Daniel F. García. 2019. Industrial robot control and operator training using virtual reality interfaces. *Computers in Industry* 109 (2019), 114–120. https://doi.org/10.1016/j.compind.2019.05.001

[11] Vitalii Pruks, Ildar Farkhatdinov, and Jee-Hwan Ryu. 2018. Preliminary study on real-time interactive virtual fixture generation method for shared teleoperation in unstructured environments. In *International Conference on Human Haptic Sensing and Touch Enabled Computer Applications*. Springer, 648–659.

[12] Daniel Rakita. 2017. Methods for Effective Mimicry-based Teleoperation of Robot Arms. *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction - HRI '17* (2017), 371–372. https://doi.org/10.1145/3029798.3034812

[13] Daniel Rakita, Bilge Mutlu, and Michael Gleicher. 2018. An Autonomous Dynamic Camera Method for Effective Remote Teleoperation. In *Proceedings of the 2018 ACM/IEEE International Conference on Human-Robot Interaction - HRI '18*. ACM Press, New York, New York, USA, 325–333. https://doi.org/10.1145/3171221.3171279

[14] Tobias Rodehutskors, Max Schwarz, and Sven Behnke. 2015. Intuitive bimanual telemanipulation under communication restrictions by immersive 3D visualization and motion tracking. *IEEE-RAS International Conference on Humanoid Robots* 2015-Decem (2015), 276–283. https://doi.org/10.1109/HUMANOIDS.2015.7363547

[15] Toshitaka Suzuki and Tetsushi Oka. 2016. Grasping of unknown objects on a planar surface using a single depth image. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM* 2016-Septe (2016), 572–577. https://doi.org/10.1109/AIM.2016.7576829

[16] John Thomason, Photchara Ratsamee, Jason Orlosky, Kiyoshi Kiyokawa, Tomohiro Mashita, Yuki Uranishi, and Haruo Takemura. 2019. A Comparison of Adaptive View Techniques for Exploratory 3D Drone Teleoperation. *ACM Transactions on Interactive Intelligent Systems* 9, 2-3 (2019), 1–19. https://doi.org/10.1145/3232232

[17] David Whitney, Eric Rosen, Daniel Ullman, Elizabeth Phillips, and Stefanie Tellex. 2018. ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 1–9. https://doi.org/10.1109/IROS.2018.8593513

[18] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. 2018. Deep Imitation Learning for Complex Manipulation Tasks from Virtual Reality Teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1–8. https://doi.org/10.1109/ICRA.2018.8461249 arXiv:1710.04615