WFH-VR: Teleoperating a Robot Arm to set a Dining Table across the Globe via Virtual Reality

Lai Sum Yim[1,*], Quang TN Vo[2,*], Chi-Ruei Wang[1], Po-Lin Li[1],
Hsueh-Cheng Wang[1], Haikun Huang[2], and Lap-Fai Yu[2]

***Abstract*— We present Work-from-Home Virtual Reality (WFH-VR), an easy-to-deploy, virtual reality-based teleoperation system for controlling a robot arm. Our system is constructed using a consumer-grade virtual reality device (an Oculus Quest 2) and a low-cost robot arm (a LoCoBot), so it can be easily replicated and set up. Using our system, the user uses a VR controller to control a virtual robot arm to manipulate virtual objects in virtual reality. The joint orientations of the virtual robot arm are sent to the real robot arm located remotely to update its pose for picking up real objects. On the other hand, our system continuously runs deep object pose estimation on the robot side to estimate the pose parameters of the real objects, which are sent to the virtual reality side to update the poses of the virtual objects. Through these processes, our system synchronizes the robot arm and objects in virtual reality and the real environment.**

We verified that our system can be applied across the globe for VR robot teleoperation with very little latency. We also performed user study experiments to validate the effectiveness of our VR teleoperation system compared to alternative approaches such as keyboard-based teleoperation on a 2D screen and automated deep grasping. The results show that users can perform different object manipulation tasks, such as dining table setting tasks, efficiently and efficaciously using our system.

I. INTRODUCTION

The recent COVID-19 epidemic has inspired global efforts to develop robots to facilitate remote work. In daily scenarios such as a restaurant, the interaction between people is almost inevitable, which increases the difficulty of social distancing and epidemic prevention [1], [2]. In view of such challenges, we devise an intuitive virtual reality-based teleoperation system for controlling a robot arm to perform tasks such as setting a dining table as Figure 1 shows. Akin to recent efforts on VR teleoperation [3], [4], [5], we devise such a system using a consumer-grade virtual reality headset (e.g., Oculus Quest 2) and a low-cost robot arm (e.g., LoCoBot), which are commonly available and can be deployed at scale.

To teleoperate a robot arm, the user sits on a chair, wears a VR headset, and holds a VR controller. Through the VR headset, the user sees a virtual environment with virtual objects and a virtual robot arm which resemble the real ones on the robot side located remotely (e.g., at the other side of the globe). The user uses the VR controller to control the robot arm to perform actions such as grasping. On the other hand, at the robot side, the system constantly runs object pose estimation to estimate the poses of the real objects, which are transmitted to the VR side for updating the virtual environment to synchronize with the real scene to facilitate visual perception and teleoperation.

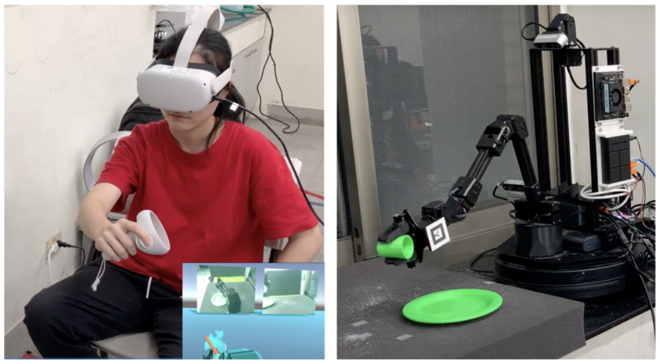
Fig. 1. A user controls a LoCoBot located remotely to set up a dining table in virtual reality.

There has been significant attention and competitions [6] on service robots' grasping and manipulation tasks such as setting up a dining table. We also evaluated our system based on dining table setting tasks. Through virtual reality, the users controlled a robot arm to arrange tableware (e.g., plates, forks) to match target positions and to perform tasks (e.g., pouring water). We recorded the success rates and time for performing different tasks, and the users' feedback about using our system. Our major contributions include:

- Proposing a hands-on, easy-to-deploy VR robot teleoperation system based on a consumer-grade VR headset and a low-cost robot arm to facilitate human-robot collaboration. We will release our toolkit to facilitate its adoption and extension.
- Testing the VR robot teleoperation system across the globe and verifying that the teleoperation can run successfully with little latency.
- Evaluating the VR robot teleoperation system using different dining table tasks and comparing it with alternative approaches such as learning-based grasping [7] and keyboard control via a 2D screen.

[*]L.S. Yim and Q.T. Vo contributed equally to this work.
[1]Department of Electrical and Computer Engineering, National Yang Ming Chiao Tung University, Taiwan.
[2]Department of Computer Science, George Mason University, USA.
Corresponding author email: hchengwang@g2.nctu.edu.tw
The research was supported by Taiwan's Ministry of Science and Technology (grants 109-2321-B-009-006, 109-2224-E-007-004, and 109-2221-E-009-074). This work was funded in part by Qualcomm through the Taiwan University Research Collaboration Project. This work was funded by an NSF CAREER Award (award number: 1942531).

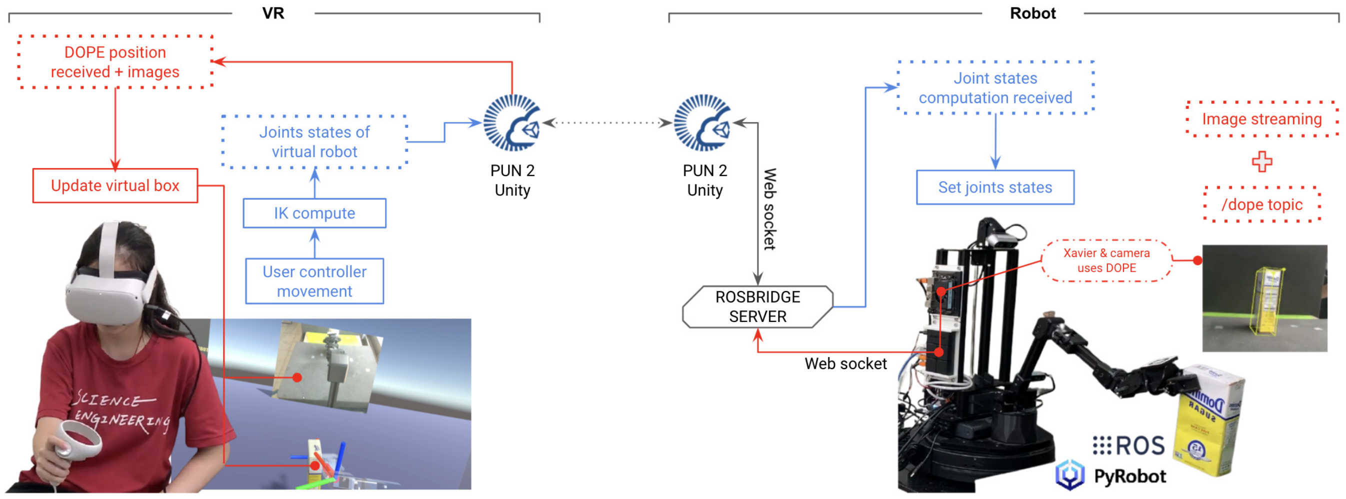

Fig. 2. An overview of the VR teleoperation system. It consists of a virtual reality interface, a robot control component, and a network component. Blue refers to the communication from the VR side to the robot side. Red refers to the communication from the robot side to the VR side.

II. RELATED WORK

A. VR Teleoperation

There have been several attempts of VR teleoperation systems to control robot end effectors in 3D space instead of relying on computer monitors and joysticks/keyboards. Earlier work in [8] used an Oculus Rift to control an industrial robot manipulator. A ROS RViz plugin for Rift was introduced [9] to enable fully immersive control of a PR2 robot. Subsequently [10] used a VR-teleoperation PR2 to collect human demonstrations for imitation learning. Nevertheless, those systems were not designed for long distance control. Recently the ROS Reality [4] was designed to overcome the latency issues over the setup not in a local network by deconstructing/reconstructing point cloud and image data. [3], [5] further evaluated motion intents and cup-stacking manipulation tasks using ROS Reality with a hand tracker implementation.

Based on recent VR devices (e.g., Oculus Quest 2) that come with robust hand controller tracking, our system differs from the above systems in various aspects. The recent cloud framework (Photon Unity Network 2) provides real-time hosting for multiplayer games on Unity, making long distance control across the global possible. Without sending point cloud nor raw RGB-D images from robot side, our system also benefits from the recent success of deep network on object pose estimation [11] and the progress of GPU computing units. We can run only the state-of-the-art algorithms on a low-cost mobile manipulation platform (LoCoBot); only the estimated object poses computed locally on the robot side are sent to the remote VR side.

B. Low-cost Robotic Manipulation Platforms

There have been limited research using low-cost robots and manipulators. Earlier work in [12] developed a reinforcement learning-based self-adaptive manipulator systems, although potentially imprecise, that enable interactions with human or environments. [13] developed a low-cost ($5,000 USD) dual-arm system called Blue, which demonstrated a VR-based interface for teleoperating data collection for human demonstrations. However, the Blue system is not publicly available yet. Recently, [7] assembled a low cost mobile manipulator (LoCoBot) under 3K USD, and have scaled and deployed over 30 of such systems to universities globally. [7] developed visual grasping policies learnt from data collected in 9 real-world home environments. The policies were formulated as planar grasping described in [14], which adopted a convolutional neural network to predict grasping positions and orientations on input images end-to-end using self-supervising data collection. Subsequently a high-level robot-agnostic PyRobot API [15] was built on top of ROS middleware and low-level actuators that supports LoCoBot and other platforms. PyRobot also supported various simulation frameworks such as Habitat [16] and Mojoco [17].

Our work uses LoCoBot which can be easily replicated and set up. PyRobot did not by default support Unity. Our proposed framework is similar to ROS Reality [4], but making more use of PyRobot high-level APIs on top of ROS. Only a side camera was added to track object poses to circumvent occlusion problems as possible. The visual grasping policies Robust-Grasp in [7] were used as our autonomous baseline approach for comparison.

C. Manipulation Tasks for VR

Robotic competitions have pushed forward progresses of the state-of-the-art methods. [5] suggested in the DARPA Robotics Challenge 2013 [18] that many completing tasks that require grasping and manipulation were too difficulty for autonomous solutions, suggesting that human teleoperation was a more practical alternative. On the other hand, autonomous robotic grasping methods [19], [20] in logistic domains were very successful in the Amazon Picking/Robotics Challenges. Other grasping methods such as the DexNet [21], [22] achieve high success rates. Zeng et al. [23] used reinforcement learning to decide whether to push or separate adjacent objects or to pick up objects. Therefore grasping-only tasks may not require teleoperation.

Recent competitions in major robotics conferences usually used the standard YCB benchmark objects [24]. A competition in [6] designed tasks such as setting up a dining table for service robots. Those tasks posed challenges not yet well-addressed including dexterous manipulation skills or rearrangement/placements using object initial/target poses. Recent VR-teleoperation work [4] designed 20 tasks of different movements that require either one or two manipulators to complete, allowing 5 VR-teleoperation attempts for each task. We devise a VR-based teleoperation system for controlling a robot arm by a human operator, which is particularly useful for performing complex and uncertain tasks that need human judgement in the manipulation process.

III. OVERVIEW

Figure 2 shows an overview of our approach. The user and the physical robot are located remotely from each other. The system consists of two major operations flows: control the robot arm and synchronize the virtual objects in virtual reality with real-world objects. The Unity PUN2 is responsible for the communication. An Oculus VR headset will visualize the virtual environment for the user while the controllers will be used to control the robot's arm. In the virtual interface, the objects are replicates (by 3D reconstruction or CAD modeling) of the real-world objects at the physical workplace. To control the real robot arm and gripper, the user will guide a virtual robot's arm identical to the arm at the physical workplace, having the same joints and dimensions. Also, the interface includes a video streamed from the workplace to the virtual reality scene via PUN2.

The control system consists of the user's controller movement in virtual reality and synchronizes that movement over the network using Unity PUN2. Based on the virtual end effector's position, our system computes the joints states of the virtual robot arm, which, upon confirmation by the user, are published to the physical workplace's Rosbridge server to update the orientations of the real robot arm. To synchronize the virtual objects with the real-world objects, pose estimation is continuously performed by capturing RGB images with the camera attached to the base of the real robot using Deep Object Pose Estimation (DOPE). DOPE estimates the pose parameters (positions and orientations) of the real-world objects for updating the virtual objects.

With this system, the user can see the virtual reality scene synchronized with the object settings and the real world at the physical workplace. The user can remotely control the robot arm to manipulate the real-world objects. Note that all the data transmitted (robot joint orientations, object poses) is minimal in size so that instant control and synchronization between the real and virtual scene is achieved, making the system more intuitive and convenient to use.

IV. TECHNICAL APPROACH

A. Hardware

Our teleoperation platform is based on a consumer-grade VR device, the Oculus Quest 2, which costed about \$420, and a low-cost robot, LoCoBot, which costed about \$5,000. For VR devices, the Oculus Quest 2 system provides a headset for head-mounted display, which is a singular fast-switch LCD panel with a $1,832 \times 1,920$ per eye resolution and runs at a refresh rate of up to 120 Hz. It also comes with two Oculus Touch hand controllers, each with 6 DoF pose tracking with infrared LEDs, which allows the controllers to be fully tracked in 3D space by the Oculus Quest 2 constellation system.

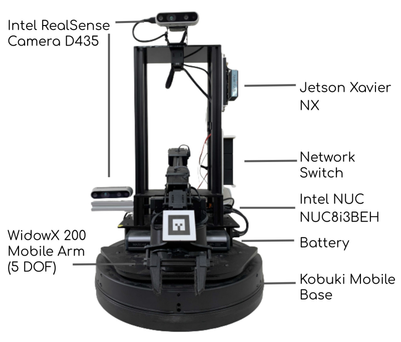

Fig. 3. Our robot setup.

Figure 3 depicts our robot. We used the LoCoBot, a low-cost mobile manipulator robot, for our purpose. This robot comprises 5 main components crucial for the control system: Intel RealSense RGB-D Camera D435, a Jetson Xavier NX, WidowX 200 Mobile Arm (5 DOF), Intel NUC, Kobuki Base. In addition to the original camera at the robot's top, we attached another Intel RealSense RGB-D Camera D435 near the arm base. The camera at the base is used to capture RGB images for pose estimation of objects by DOPE. Note that we do not use the original camera at the top of the robot to capture images for running DOPE due to the high possibility of robot arm occlusions, which can cause pose estimation failures. Our teleoperation system is developed in Unity, a 3D game engine that supports major VR headsets such as the Oculus.

B. Pose Estimation

Researchers started to apply deep neural networks for 3D object detection and pose estimation. For example, Tremblay et al. [11] introduced the deep object pose estimation (DOPE) method for robotic grasping of household objects. In our system, we applied DOPE for estimating object poses at the robot side. Rather than for robotic grasping, the estimated object poses at the robot side are used to update the corresponding virtual objects' poses at the VR side located remotely to synchronize the real working environment (at the robot side) with the virtual working environment (at the VR side) to facilitate remote teleoperation.

The physical LoCoBot is equipped with two cameras at the top and bottom right of the base as shown in Figure 3 which we have the exact position and orientation at any given moment. We utilize the top camera for real-time image streaming from robot to virtual reality. The side camera is response for capturing images for pose estimation of the objects. The robot is also equipped with a NUC computer responsible for the communication between the robot and Unity and Xavier NX for running DOPE calculation. The NUC is connected to Rosbridge Master IP, an open-source program that converts JSON API to ROS functionality, which has the cameras' coordination topic and DOPE's object topic readily to listen by Unity.

In the robot's Unity program, we use the ROS# package, open-source software libraries and tools in C# for communicating with ROS from .NET applications, connecting with ROS Master IP and listening to the related topic of the robot's image streaming and DOPE object's orientation in Unity. For object rendering (e.g., a Domino sugar box), we leverage the popular YCB object model for user reference.

C. Robot Arm Control in Virtual Reality

1) Visual Interface: To effectively control the robot in virtual reality, the user must have a good reference about the surrounding at the robot side. After the Unity system at the robot side has received the position and orientation data from DOPE through the local Rosbridge connection, it sets the virtual objects accordingly to synchronize with the real-world settings. The object's poses will be synchronized with the VR side over RPC protocol from PUN2. However, self-occlusion sometimes occurs when the robot arm is picking up objects, causing DOPE to fail to estimate the objects' poses. To mitigate this problem, our visual interface at the VR side uses a rigid body to represent the picked-up virtual object when DOPE fails to estimate objects' poses. The rigid body is attached to the robot arm so that its virtual object's pose is updated following the movement of the virtual robot arm controlled by the user. In other words, the visualization provided by the rigid body acts as a backup plan in case of self-occlusion. In this case, our system renders the object as mild transparent, indicating the object pose uncertainty.

To further guide the user in VR, we use the robot's top camera to stream the RGB video of the robot side (i.e. the tabletop) to a screen in the virtual working environment. This will enable the user to have a better understanding and reference of where the objects are and what is happening at the real robot side. The image streaming uses PUN2's RPC protocol to synchronize the streaming data from the robot side to VR. However, due to the size of each frame being sent over the network, there is 1-2 second latency.

The user will guide a virtual robot arm (in VR) before confirming the desired positions and orientations of the joints. We take advantage of the open-source LoCoBot's URDF model by Trossen Robotics to recreate the virtual robot model inside Unity. The Oculus Quest's right controller positions and orientation will be mapped into the virtual robot arm's model, allowing the user to move the arm freely to guide the movement of the virtual robot arm, which is visualized in real time.

2) Control Interface: To minimize the latency in communication and avoid expensive computation time at the robot side which uses the PyRobot's inverse kinematic solver, we instead use the Unity's kinematic chain and IK solver to compute the physical robot's joints' positions and orientations based on the end effector of the virtual LoCoBot. Once the joints' positions and constraints have been set, Unity uses the FABRIK algorithm [25] to compute the joints' movement based on the end effector of the virtual robot, which is mapped into the user's controller by updating its position and orientation to be the same as those of the user's right controller. The computation happens three times per frame, allowing the user to have a smooth movement without any delay. The real robot arm movement is synchronized with the virtual robot arm's movement using the Unity PUN2's RPC protocol. Once the user confirms the joints' desired positions and orientations by pressing the controller's grip button, the system invokes the ROS# publisher method over the Unity program at the real robot side to publish the virtual robot's joints states to its local Rosbridge server, which then triggers a handler script to read the virtual joints topic and uses PyRobot API [15] at the robot's NUC to set the joints of the physical robot arm accurately. This communication avoids the PyRobot's IK computation latency and sets the real robot joints right away, which takes less than 1 second from the user's movement to the robot's movement.

D. Network

Photon Unity Network 2 (PUN2) is a real-time cloud framework that hosts online multiplayer games for Unity developers. We leverage PUN2 by using its Remote Procedure Calls protocol to send and receive data remotely with minimum latency. Using this system, VR users will send their controllers' inputs and movements to the server, which will be received by the robot side to control the robot accordingly. Vice versa, the robot side will send video streaming and object's pose data estimated by DOPE to the VR side, which will then synchronize the virtual working environment with the real world status. Since only the robot side needs to be connected to its local Rosbridge server, the VR user can control the robot anywhere with access to the internet.

V. VR Robot Teleoperation across the Globe

We investigated the user experience in teleoperating a robot arm (i.e. a LoCoBot) across the globe and also in performing some common dining table tasks. We tested the proposed VR teleoperation platform between two universities across the globe. The VR side was set up at George Mason University in Virginia, USA. The real robot side was set up at National Yang Ming Chiao Tung University in Taiwan.

The user in USA put on a VR headset and controlled a LoCoBot robot arm in Taiwan. The user was instructed to pick up a box and place it at a target position. The user saw a virtual box and a virtual robot arm which resembled the real box and the real robot arm in Taiwan. In addition, a live video showing what was happening at the robot side was streamed to a virtual screen in the virtual working environment to facilitate teleoperation.

The user reported no awareness of the latency of the robot arm control. The user commented that the pose update of the virtual box (synchronized with the real box) helped him a lot in manipulating the object as he could intuitively judge the current status of the box as he teleoperated the robot. There was about one second of latency in streaming the video, yet the latency did not adversely affect the user's judgement and performance as the manipulation motion was not done at a very fast pace. The results showed that our system could

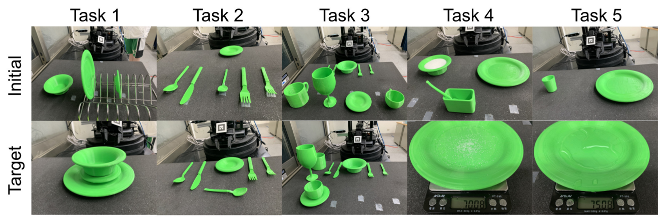

Fig. 4. The five tasks of the dining table experiment.

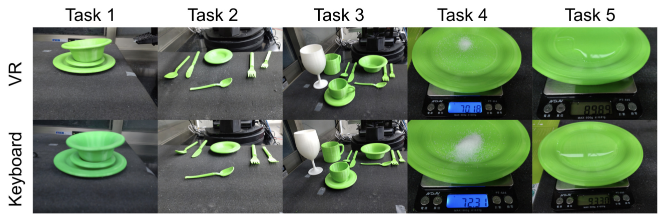

Fig. 5. Target states reached by the researcher using VR and keyboard.

support across the globe manipulation with minimum latency and the DOPE pose estimation facilitated teleoperation.

VI. Experiments: Set a Dining Table

A. Task Feasibility

1) Direct manipulation: We conducted a feasibility evaluation of various tasks. The commonly used direct manipulation (**DM**) as in [4] procedure was done by two experimenters, who physically grabbed the arm and moved the joints while the robot was set to teaching mode. DM was used to test if the 5-DOF arm and gripper were physically capable of performing certain tasks.

The experiment tasks are inspired by the multi-year IROS Robotic Grasping and Manipulation Competition - Service Robot Track. The objects we used in the experiments are also inspired by the YCB objects and common objects on a dining table. Due to the hardware payload and gripper limitations, we chose to scale down the objects by modeling CAD models followed by 3D printing. Such modifications also considered easy replication of our work using LoCoBot hardware. All CAD models will be publicly available. Figure 4 depicts the five tasks. We chose a range of tasks and modified them to fit with our robot setup.

Task 1: stack ***plates and a bowl****.* This task required the robot to pick up plates and a bowl from an initial position and then stack them up in order. This is challenging because it is easy to knock down an object placed previously. We evaluate the performance by counting the scores as below **rules**: **0)** Both bowl and small plate are within the big plate (100 pts); **1)** if the bowl touches the big plate (-20 pts); **2)** if the bowl or the small plate is out of the big plate from the top view (-20 pts).

Task 2: pick and place ***tablewares****.* This task required the robot to pick up the tablewares and rearrange them to match the target state. This is challenging because the tablewares are flat. The user also needed to surround a small plate with the tablewares. We evaluate the performance by counting the scores as below **rules**: **0)** if all the tablewares are arranged in the right relative positions (100 pts); **1)** the tablewares should be arranged vertical to the table edge (if the angle error is more than 5 degree, -5 pts for each tableware); **2)** the tablewares should be arranged into the specific poses (if not, -5 pts for each tableware).

Task 3: rearrange ***glasses and cups****.* This task required the robot to pick up a tea cup, a wine glass, a mug, and a small plate from their initial positions and rearrange them to match the target. It is challenging as the robot could easily knock down an object placed previously. The tea cup also needed to be put on a small plate. We evaluate the performance by counting the scores as below **rules**: **0)** all the cups are arranged in the right relative positions (100 pts); **1)** if one object isn't arranged in the right relative positions (-25 pts); **2)** if the teacup is not on the small plate (-5 pts); **3)** if the positions of the tablewares are changed during the task (-5 pts).

Task 4: transport ***sugar grains****.* This task aimed to grasp a small spoon to scoop sugar grains from the bowl and place the grains on a big plate. The scooping action may be limited to the 5 DOF joints, but still doable. We evaluate the performance by counting the scores as below **rules**: **0)** more than 5g (100 pts); **1)** 4g to 5g (90 pts); **2)** 3g to 4g (80 pts); **3)** 1g to 3g (60 pts). For VR-1 and KM Baseline-1 experiments, the targets of all rules would be divided by 2 because the task was modified to execute only one scooping action.

Task 5: transport ***liquid****.* The task aims to grasp a small water cup to pour water (representing wine/sauce) onto a plate. Similar to Task 4, the pouring action may be limited but still doable. We evaluate the performance by counting the scores as below **rules**: **0)** more than 13g (100 pts); **1)** 12g to 13g (90 pts); **2)** 11g to 12g (80 pts); **3)** 10g to 11g (60 pts). For VR-1 and KM Baseline-1 experiments, the targets of all rules would be added by 10g because the task was modified to use fully filled water cups.

There were other tasks designed in the competition, but we found them infeasible for the LoCoBot setup. The **ice cubes** task requires the manipulator to pick up a pair of tongs from a tray, hold it, and use it to pick and place three ice cubes, and drop the cubes into the water glass. Picking up a pair of tongs and use it to indirectly manipulate objects may require more dexterous skills and touch sensors. Our LoCoBot gripper was currently set to only open/close modes, and therefore was unable to perform such tasks. The **sugar packet** task requires to pick up a deformable packet of sugar, tear it open, and pour half of the packet into a tea cup. Such task may need dual-arm manipulation and handle deformable objects, which were currently infeasible by our hardware setup.

2) Learning-based Grasping Policies (Robust-Grasp): We further evaluated if the tasks could be simply completed by existing grasping policies provided by PyRobot [15]. If the autonomous grasping methods work well out-of-the-box, the VR teleoperation may only be used for error recovery or re-attempts. Here we carried out the Robust-Grasp algorithm twice: one with object located at the initial position and the other at the target position. We recorded the trajectories

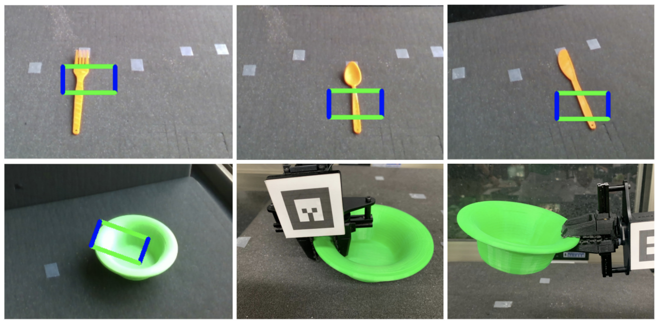

Fig. 6. Top three: gripping points predicted by Robust-Grasp used for successfully picking up the objects. Bottom left: gripping point predicted by Robust-Grasp. Bottom middle: gripping performed by the LoCoBot. It failed to grip the rim of the bowl due to its special shape. Bottom right: gripping the bowl by VR teleoperation.

of arm movements for both grasps. Then we connected the recorded trajectories of the first grasp, and the reversal trajectories of the second grasp. Finally we replayed the entire connected trajectories together to perform the tasks.

Given that the Robust-Grasp was formulated as planar grasping, we found it feasible for *Task 2: pick and place tablewares*, but not other tasks. The problem was that Robust-Grasp did not consider 6 DoF grasps such as motion primitives to grasp from the side or 45 degrees. Figure 6 demonstrated the success and failure cases.

B. User Evaluation

We recruited 10 users to use our VR teleoperation system for the 5 tasks feasible for our hardware setup. All participants were first-time users. After an instruction the participants worn the VR head mount and used the hand controller to interact with the objects. They were not instructed to grasp the objects in a specific way, and therefore all possible motion primitives, such as pushing, grasping, placing, were allowed. In Tasks 1 to 3, there was a 5-minute time limit for each task, but there was no limit in terms of number of attempts until the task was completed. In Task 4 and 5, we only allowed one attempt to scoop sugar grains or transport the fully-filled water cup. We also reported a baseline using keyboard and monitor (KM) by a researcher, who experienced with our VR teleoperation, to perform 5 trials of each tasks. Figure 5 shows this researcher's results. Table I shows the average scores and time spent of using VR, keyboard and monitor (KM), and the Robust-Grasp method. Overall both VR and KM methods were able to complete all tasks. However, the execution time by using VR was shorter than that by KM. We found that performance on Task 2 was comparable among the 3 methods, which showed that Task 2 was relatively easy to complete by planar grasping. Nevertheless, we observed that the VR inferface allows different **motion primitives** like pushing or fine-grained adjustment of the positions or orientations of the knife and fork while the pick-and-place was not aligned. VR also showed superior performance in Tasks 4 and 5 than KM by better **fluency in 6 DoF grasping**. During the transporting, the sugar grains or water tended to drop or splash out by using KM due to a sequence of noncontinuous strokes. Using

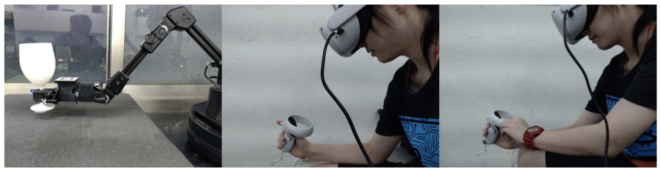

Fig. 7. Left: the end effector needed to be perpendicular to the wine glass leg to grasp. Middle: the VR hand controller needed to be perpendicular to the human wrist to control the end effector. Right: another hand helped the VR hand controller to stabilize and reach a desired pose.

TABLE I

RESULTS OF AVERAGE SCORE AND TIME SPENT IN EACH TASK.

	VR-10-Novice Score/Sec	VR-1-Expert Score/Sec	KM-1-Expert Score/Sec	R-Grasp Score/Sec
1: plates/bowl	62 / 139.6	80 / 102.75	84 / 197.25	-
2: tablewares	87 / 135.5	98 / 92.7	90 / 143.5	85 / 271*
3: glasses/cups	91.5 / 169.3	97 / 110.75	98 / 260.75	-
4: sugar	85 / 156.1	92 / 50.50	48 / 86.25	-
62 5: liquid	83 / 77.4	100 / 50.25	28 / 61.75	-

* The execution time of R-Grasp (Robut-Grasp) depends on the replay frame rate of the recorded trajectories and settings in PyRobot APIs.

VR led to smoother and more intuitive control than KM. VR is also useful in Task 4: as the spoon reaches into the sugar grain container, the scooping required **dexterous manipulation** to successfully retrieve the desired amount of sugar grain. Such skills were difficult to perform using the KM method.

We also found some **misalignment between human and robot joints** that prohibited efficient teleoperation. All participants were first-time users to our system, and were not familiar with joint limits and the degree of freedom of the LoCoBot arm. Some target poses of the end effector assigned by humans may not be reachable. On the other hand, a human's wrist joint angle has limitations so that some desired trajectories could not be achieved by just one pick-and-place, and therefore regrasping by multiple pick-and-place actions were required. Such case happened more often while rearranging the wine glass in Task 3. Finally, we observed that the high sensitivity of VR hand controller may cause noisy data due to minor human hand shaking without intention. Some participants used another hand to stabilize the VR hand controller for task 3 as shown in Figure 7.

VII. CONCLUSION

We presented a handy approach for VR teleoperation of a robot arm. We will release our toolkit to facilitate adoption and future extension. Current limitations and possible extensions include: a) possible pose estimation failures due to self-occlusion, which may be addressed by using additional cameras; b) limited flexibility due to the use of only one robot arm with a gripper hand, which could be addressed by using an additional arm and a five-finger hand for performing more sophisticated manipulations; c) lack of considerations of deformable objects (e.g., clothes), which may be addressed by estimating the 3D geometry of objects in real time.

References

[1] Z. Li, P. Moran, Q. Dong, R. J. Shaw, and K. Hauser, “Development of a tele-nursing mobile manipulator for remote care-giving in quarantine areas,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3581–3586.

[2] J. Li, Z. Li, and K. Hauser, “A study of bidirectionally telepresent tele-action during robot-mediated handover,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2890–2896.

[3] E. Rosen, D. Whitney, E. Phillips, G. Chien, J. Tompkin, G. Konidaris, and S. Tellex, “Communicating and controlling robot arm motion intent through mixed-reality head-mounted displays,” *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1513–1526, 2019.

[4] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, “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, 2018, pp. 1–9.

[5] D. Whitney, E. Rosen, E. Phillips, G. Konidaris, and S. Tellex, “Comparing robot grasping teleoperation across desktop and virtual reality with ros reality,” in *Robotics Research*. Springer, 2020, pp. 335–350.

[6] Z. Liu, W. Liu, Y. Qin, F. Xiang, S. Xin, M. A. Roa, B. Calli, H. Su, Y. Sun, and P. Tan, “Ocrtoc: A cloud-based competition and benchmark for robotic grasping and manipulation,” *arXiv preprint arXiv:2104.11446*, 2021.

[7] A. Gupta, A. Murali, D. Gandhi, and L. Pinto, “Robot learning in homes: Improving generalization and reducing dataset bias,” *arXiv preprint arXiv:1807.07049*, 2018.

[8] IVRE – an immersive virtual robotics environment. [Online]. Available: https://cirl.lcsr.jhu.edu/research/human-machine-collaborative-systems/ivre/

[9] PR2 Surrogate. [Online]. Available: http://wiki.ros.org/pr2_surrogate

[10] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, “Deep imitation learning for complex manipulation tasks from virtual reality teleoperation,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5628–5635.

[11] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, “Deep object pose estimation for semantic robotic grasping of household objects,” *arXiv preprint arXiv:1809.10790*, 2018.

[12] M. P. Deisenroth, C. E. Rasmussen, and D. Fox, “Learning to control a low-cost manipulator using data-efficient reinforcement learning,” in *Robotics: Science and Systems VII*, vol. 7, 2011, pp. 57–64.

[13] D. V. Gealy, S. McKinley, B. Yi, P. Wu, P. R. Downey, G. Balke, A. Zhao, M. Guo, R. Thomasson, A. Sinclair *et al.*, “Quasi-direct drive for low-cost compliant robotic manipulation,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 437–443.

[14] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3406–3413.

[15] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, S. Gupta, and A. Gupta, “Pyrobot: An open-source robotics framework for research and benchmarking,” *CoRR*, vol. abs/1906.08236, 2019. [Online]. Available: http://arxiv.org/abs/1906.08236

[16] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, “Habitat: A platform for embodied ai research,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9339–9347.

[17] E. Todorov, T. Erez, and Y. Tassa, “Mujoco: A physics engine for model-based control,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[18] G. Pratt and J. Manzo, “The darpa robotics challenge [competitions],” *IEEE Robotics & Automation Magazine*, vol. 20, no. 2, pp. 10–12, 2013.

[19] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, N. Fazeli, F. Alet, N. C. Dafle, R. Holladay, I. Morona, P. Q. Nair, D. Green, I. Taylor, W. Liu, T. Funkhouser, and A. Rodriguez, “Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching,” in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2018.

[20] C. Hernandez, M. Bharatheesha, W. Ko, H. Gaiser, J. Tan, K. van Deurzen, M. de Vries, B. Van Mil, J. van Egmond, R. Burger *et al.*, “Team delft’s robot winner of the amazon picking challenge 2016,” in *Robot World Cup*. Springer, 2016, pp. 613–624.

[21] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, “Dex-net 3.0: Computing robust robot vacuum suction grasp targets in point clouds using a new analytic model and deep learning,” *arXiv preprint arXiv:1709.06670*, 2017.

[22] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, “Learning ambidextrous robot grasping policies,” *Science Robotics*, vol. 4, no. 26, 2019.

[23] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, “Learning synergies between pushing and grasping with self-supervised deep reinforcement learning,” *arXiv preprint arXiv:1803.09956*, 2018.

[24] B. Calli, A. Singh, A. Walsman, S. Srinivasa, P. Abbeel, and A. M. Dollar, “The YCB object and model set: Towards common benchmarks for manipulation research,” in *2015 international conference on advanced robotics (ICAR)*. IEEE, 2015, pp. 510–517.

[25] A. Aristidou and J. Lasenby, “FABRIK: A fast, iterative solver for the inverse kinematics problem,” *Graph. Models*, vol. 73, no. 5, pp. 243–260, Sep. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.gmod.2011.05.003