Intuitive Bare-Hand Teleoperation of a Robotic Manipulator Using Virtual Reality and Leap Motion

Inmo Jang(✉), Joaquin Carrasco, Andrew Weightman, and Barry Lennox

The University of Manchester, Manchester M13 9PL, UK
{inmo.jang,joaquin.carrasco,andrew.weightman,
barry.lennox}@manchester.ac.uk

Abstract. Despite various existing works on intuitive human-robot interaction (HRI) for teleoperation of robotic manipulators, to the best of our knowledge, the following research question has not been investigated yet: Can we have a teleoperated robotic manipulator that simply copies a human operator's bare hand posture and gesture in a real-time manner without having any hand-held devices? This paper presents a novel teleoperation system that attempts to address this question. Firstly, we detail how to set up the system practically by using a Universal Robots UR5, a Robotiq 3-finger gripper, and a Leap Motion based on Unity and ROS, and describe specifically what information is communicated between each other. Furthermore, we provide the details of the ROS nodes developed for controlling the robotic arm and gripper, given the information of a human's bare hands sensed by the Leap Motion. Then, we demonstrate our system executing a simple pick-and-place task, and discuss possible benefits and costs of this HRI concept.

Keywords: Human-robot interaction · Teleoperation · Virtual Reality · Leap Motion

1 Introduction

Teleoperation of robotic manipulators have been widely studied for various domains [7,14]. Recently, this technology is attracting interests and considered to be promising for tasks in extreme environments, for example, glovebox operations [1,15]. Use of teleoperated robots will minimise the need for human workers to be exposed to radioactive hazardous materials and thus improve safety and reduce the operational costs in a long-term perspective.

To this end, as one of the stepping stones, this paper presents an intuitive human-robot interaction concept to teleoperate a robotic manipulator by using

This project has been supported by the RAIN Hub, which is funded by the Industrial Strategy Challenge Fund, part of the government's modern Industrial Strategy. The fund is delivered by UK Research and Innovation and managed by EPSRC [EP/R026084/1].

K. Althoefer et al. (Eds.): TAROS 2019, LNAI 11650, pp. 283–294, 2019.
https://doi.org/10.1007/978-3-030-25332-5_25

Virtual Reality and a Leap Motion (a hand tracking system). Our system does not require a human operator to have any hand-held devices, but the robotic manipulator simply and directly follows the posture and gesture of his/her bare hands. To the best of our knowledge (see more details in Sect. 2), this is the first attempt to teleoperate a robotic manipulator using bare hands in real-time manner without having neither virtual targets nor predefined hand gestures.

In this paper, we show how to practically setup the system using a 6-DOF robotic arm (Universal Robots UR5), a 3-finger gripper (Robotiq), a Leap Motion, Virtual Reality based on Unity (3D game engine) and ROS (Robot Operating System), and describe specifically what information is communicated between each other. Furthermore, we provide the details of the ROS nodes developed for controlling the robotic arm and gripper, given the information of a human's bare hands sensed by the Leap Motion. Then, we demonstrate our system executing a simple pick-and-place task, and discuss about possible benefits and costs of this HRI concept.

2 Related Work

Over the past decade, many researches have proposed human-robot interface concepts using Virtual Reality (VR) for teleoperation of robotic manipulators. Many of the works have utilised default hand-held devices [12,19], which may cause considerable long-term workload on a user's arms for time-consuming tasks. As alternative interfaces without hand-held devices, human motion capture systems such as *Kinect* (using vision) [13] and/or *Myo* (using electromyography) [11] have been also popularly used. However, the former needs a large space to capture the whole body of a user, and the location of such a sensor relative to VR should be calibrated carefully. The latter, as a wearable device, is convenient to use, but reportedly provide less accuracy than Kinect, requiring sensor fusion [3,11].

Recently, *Leap Motion* (LM), i.e. vision-based hand motion/gesture capture system, has started to be utilised along with VR for robotic teleoperation. In [2,4,18], predefined hand gestures are used to control a robotic manipulator. This HRI concept requires a human operator to map those input gestures to the desired output robot behaviours, although it is shown to be at least more efficient than using a default interface device for a robot given by its manufacturer (e.g. Teach pendant for *Universal Robots*) [18]. For better intuitiveness, some works use virtual objects [9,10] or waypoints [16], which are the spatial targets that the end effector of a robotic manipulator has to follow and reach, and a human can simply pick up and place such a virtual target within a virtual space. It is presented in [10] that manipulating virtual objects can reduce task completion time, compared with using *Moveit* interactive markers in Rviz. However, all the works use plan-and-execution concepts, which inevitably induce latency for every command.

Our work was inspired by the following research question: Can we have a teleoperated robotic manipulator that simply copies a human operator's hand posture and gesture in a real-time manner, without having neither virtual targets

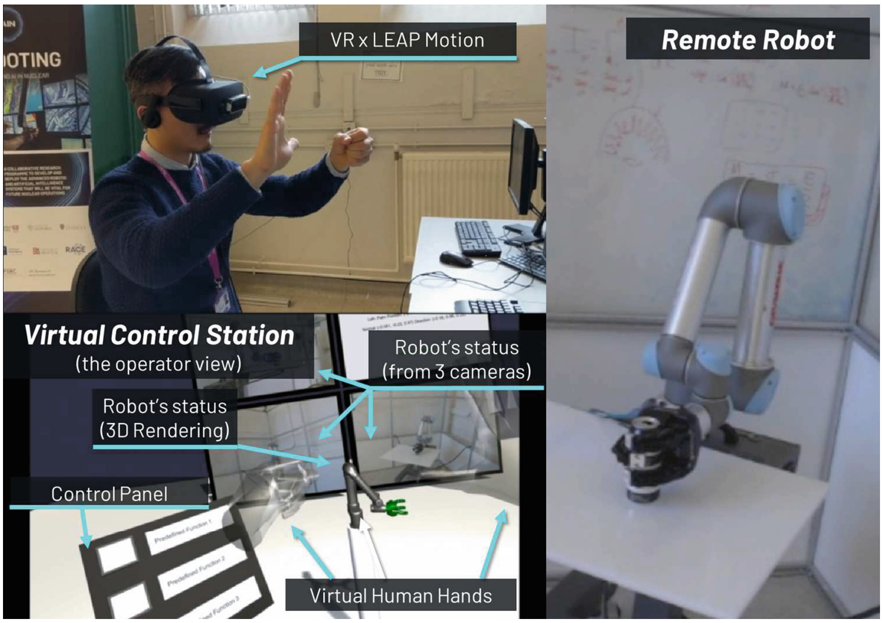


Fig. 1. The proposed teleoperation system using Virtual Reality and a Leap Motion (Up left: a user wearing a VR goggle with a Leap Motion; Bottom Left: the virtual control room; Right: the robot in a remote site)

nor predefined hand gestures nor hand-held devices? The following sections show our teleoperation system that attempts to answer this question.

3 The Proposed System

In this paper, we present a novel HRI concept where a human operator can teleoperate a robotic manipulator using his/her bare hands intuitively. In our system, as shown in Fig. 1, a human operator can be seated in a virtual control room, where there are three-view displays and a virtual robot model rendering the actual robot's status, and control panels by which the operator can give any predefined commands. The human's hands can be sensed and intuitively interact with the control panels by touching them, without any hand-held devices, via the LM attached to the outer surface of the VR goggle. More importantly, depending on control modes, i.e. *Reaching mode* or *Manipulating/Grasping mode* (see Sect. 3.2), the robot manipulator can follow the operator's bare hands sensed by the LM, which provides a high degree of intuitiveness.

3.1 System Architecture

Figure 2 shows hardware/software components comprised of our proposed teleoperation system, which uses a 6-DOF robotic manipulator (UR5), a 3-finger gripper (Robotiq), and three webcams for the slave side in a remote site, and uses a Leap Motion and a VR system (Oculus Rift) for the master side. All the devices in the remote site are communicated via *ROS* (kinetic version) running

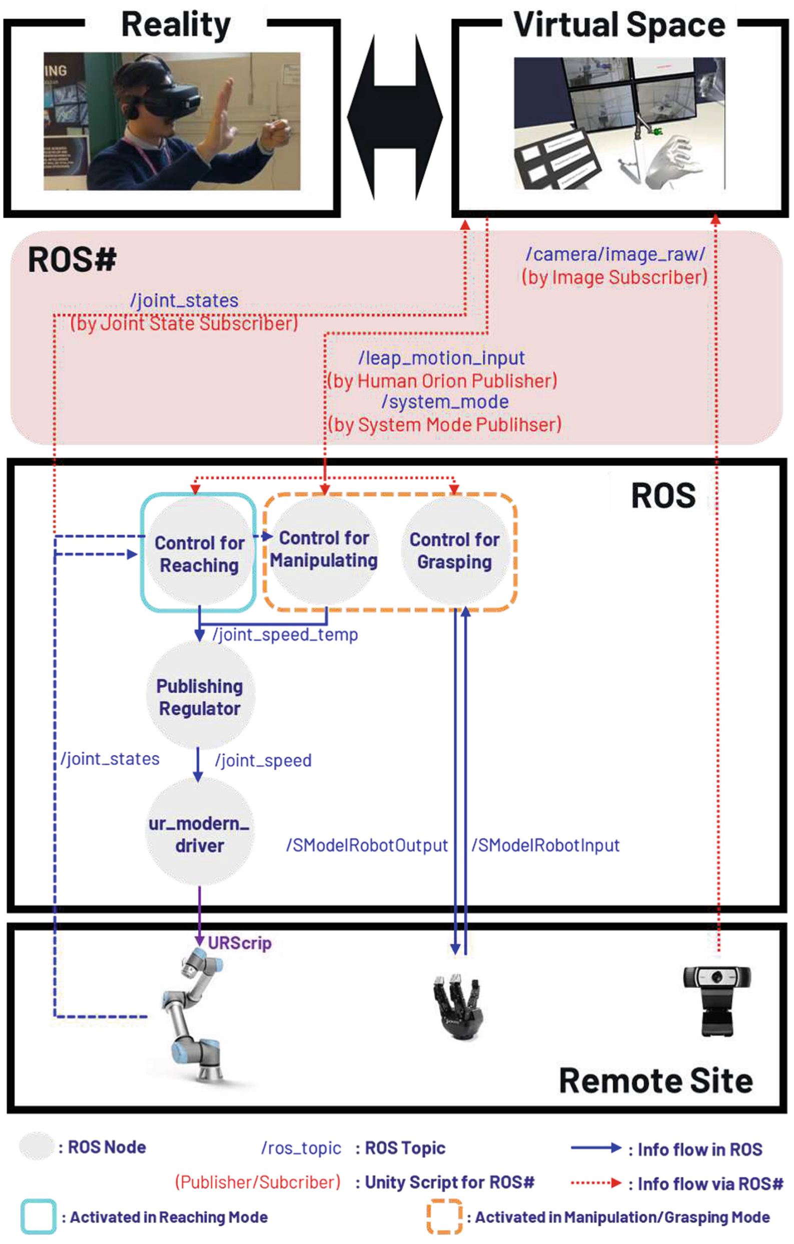


Fig. 2. System architecture of the proposed teleoperated robotic manipulator

on Ubuntu 16.04. User interface devices such as VR and LM are connected to *Unity3D* on Windows 10, and then ROS and Unity3D are linked by *ROS#*[1], which is based on *Rosbridge*[5]. Refer to [17] for more details of ROS# usage.

Unity-Side Setup: We use *Orion Beta* for LM, the recent software providing significantly better performance compared with the previous one, *V2*[2], but only available on Windows for the moment. Importantly, when Orion is used with VR, LM enables an immersive interface without hand-held devices. Compared with use of LM via ROS directly (which uses V2), Orion also gives additional information about hands such as a palm's normal and directional vectors, which are used in this work to map a user hand to the robot hand.

Furthermore, we utilise LM's Unity SDK such as *Interaction Engine*[3] to enable a user to interact with virtual objects such as the virtual control panels by simply touching them in a virtual space.

In the current teleoperation system, a user's right hand gestures are used to select a control mode, e.g. *Reaching mode* or *Manipulating/Grasping mode*, by a thumbs-up or stretching out all fingers, respectively. To this end, LM's *Detection Example*[4] is also imported.

ROS-Side Setup: ROS packages for UR5[5] and the gripper[6] are used to communicate with the devices via ROS topics such as `joint_speed` (`trajectory_msgs/JointTrajectory.msg` (http://docs.ros.org/melodic/api/trajectory_msgs/html/msg/JointTrajectory.html), i.e. the reference velocity of each joint), `joint_states` (`sensor_msgs/JointState.msg` (http://docs.ros.org/melodic/api/sensor_msgs/html/msg/JointState.html), i.e. the current status of each joint) for UR5, and `SModelRobotOutput` (`robotiq_s_model_control/SModel_robot_output.msg` (http://docs.ros.org/hydro/api/robotiq_s_model_control/html/msg/SModel_robot_output.html), i.e. gripper function registers) for the gripper.

For UR5, `ur_modern_driver` package[7] is used as recommended for newer system versions (v3.x and up). The driver receives `joint_speed` topic as an input, then transforms it to the corresponding *URScript*, which is the programming language that controls the robot at a script level. According to the URScript manual[8], "the robot must be controlled a frequency of 125 Hz, or in other words, it must be told what to do every 0.008 s".

As shown in Fig. 2, we have two ROS nodes for controlling the UR5 and one node for the gripper, and they are activated or deactivated depending on the input signal `system_node` from the Unity side, which contains a single string value indicating the control mode selected by a user.

[1] https://github.com/siemens/ros-sharp/.
[2] See comparison in https://youtu.be/7HnfG0a6Gfg.
[3] https://leapmotion.github.io/UnityModules/interaction-engine.html.
[4] https://gallery.leapmotion.com/detection-example/.
[5] https://github.com/ros-industrial/universal_robot.
[6] https://github.com/ros-industrial/robotiq.
[7] https://github.com/ros-industrial/ur_modern_driver.
[8] The URScript Programming Language, ver 3.5.4, April 12, 2018.

Despite the selected control mode, either of the two nodes for UR5 generates `joint_speed_temp` topic, which has the same message type as `joint_speed` but needs to be through the publishing regulator node because ROS does not guarantee real-time capabilities. Particularly, since the two control nodes have processing burden such as inverse and forward kinematics computation within them, it is not straightforward to keep the computation time for each processing loop consistently. Due to this fact, when `joint_speed_temp` was directly input to `ur_modern_driver` node, it was easily experienced that the movement of the UR5 was not smooth but starting and stopping repeatedly. To address this undesirable behaviour, we included the publishing regulator node. Basically, this node simply subscribes `joint_speed_temp` and then publishes it as `joint_speed`, but if there is no new topic received within 0.008 sec after the previous topic, it instead publishes the previous topic again. Although this does not provide proper real-time capability as well, at least the movement of the UR5 has become much smoother.

For the gripper, the control node for grasping subscribes `leap_motion_input`, which contains the user hand's grabbing strength sensed by the LM, i.e. `grab_strength` $\in [0, 1]$, where the value of 1 represents fully-closed hand. Depending on the grabbing strength and its rate of change, `rPRA`, `rPRB`, and `rPRC` (i.e. the robotic gripper's each finger position request) and `rSPA`, `rSPB`, `rSPC` (i.e. speed request) are chosen and comprised of `SModelRobotOutput`, which is then published to the gripper.

ROS# Setup: ROS# publishes the user's input such as `system_mode` and `leap_motion_input` from the Unity side to the ROS side, while subscribing the status information from the remote site such as `joint_states` and `camera`. Here, `leap_motion_input` contains ROS message `Human_Orion.msg`, which is based on `Human.msg` of the existing ROS package for LM but modified by ourselves to have new information such as a user palm's normal and directional vectors. To implement the publishing and subscribing processing between the Unity and ROS sides, we also created the C# scripts such as `Human Orion Publisher` and `System Mode Publisher` and utilised existing ones such as `Joint State Subscriber` and `Image Subscriber`.

3.2 How Bare Hands Control the Robotic Manipulator?

As mentioned previously, there are two control modes in the current system: *Reaching mode* and *Manipulating/Grasping mode*. In Reaching mode, the three-dimensional Cartesian position of the robot end effector only follows the operator's palm position sensed via the Leap Motion only during the time when the hand is closing, which activates the movement of the robot. This activation functionality is intended to reduce excessive human concentration load during the entire operation, preventing undesirable accidents caused by the operator' possible unconscious hand gestures or movements. This section describes more details of each control mode.

Algorithm 1. ROS node: Control for Reaching

```
1: while ROS is not shutdown do
2:     if grab_strengh $\geq c_1$ and control_mode = ‘reaching’ then
3:         $\theta_k \leftarrow$ joint_states
4:         ${}^6_0\mathbf{B}_k \leftarrow$ kin.forward($\theta_k$)
5:         $\Delta\mathbf{p} \leftarrow$ palm_position$_k$ - palm_position$_{k-1}$
6:         $\mathbf{p}^d_{k+1} \leftarrow \mathbf{p}_k + \Delta\mathbf{p}$
7:         Construct ${}^6_0\mathbf{B}^d_{k+1}$ using $\mathbf{p}^d_{k+1}$
8:         $\theta^d_{k+1} \leftarrow$ kin.inverse(${}^6_0\mathbf{B}^d_{k+1}$)
9:         $\dot{\theta}^d_{k+1} = (\theta^d_{k+1} - \theta_k)/\Delta t$
10:        joint_speed $\leftarrow \dot{\theta}^d_{k+1}$
11:        Publish joint_speed and sleep $\Delta t$.
12:    end if
13: end while
```

Reaching Mode: The description of this control mode is shown as Algorithm 1. Basically, the main loop runs at every computation time instant k after each sampling time $\Delta t = 0.008\,\mathrm{s}$, if a user's hand is closing (i.e. `grab_strength` from topic `leap_motion_input` is larger than a certain threshold constant c_1) as well as Reaching mode is selected (i.e. `control_mode` from topic `system_mode`). Note that we set c_1 to 0.9 because `grab_strength` sometimes becomes less than 1 due to the sensor noise even when a human hand is fully closed. At first, the ROS node obtains the current status of the robot arm in joint space, i.e. `joint_states`. It is then, by using the forward kinematics method for UR5 [6][9], transformed to the corresponding position and orientation of the end effector in Cartesian space, ${}^6_0\mathbf{B}_k \in \mathbb{R}_{4\times4}$, which is defined as

$$ {}^6_0\mathbf{B}_k = \begin{bmatrix} \mathbf{E}_k & \mathbf{p}_k \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix}. \tag{1} $$

$\mathbf{E}_k = [\mathbf{u}_{X_6}, \mathbf{u}_{Y_6}, \mathbf{u}_{Z_6}] \in \mathbb{R}_{3\times3}$ indicates the orientation of the end effector, where $\mathbf{u}_{X_6}$, $\mathbf{u}_{Y_6}$, $\mathbf{u}_{Z_6} \in \mathbb{R}_{3\times1}$ are unit vectors representing each orthogonal direction of the end effector coordination frame with respect to the robot arm's base frame (i.e. the shoulder). $\mathbf{p}_k \in \mathbb{R}_{1\times3}$ is the position of the end effector in task space.

Then, it obtains the desired amount of the end effector translation, $\Delta\mathbf{p}$, using the user's palm position from topic `leap_motion_input`. It updates the desired position $\mathbf{p}^d_{k+1}$ and construct ${}^6_0\mathbf{B}^d_{k+1}$, which is then transformed to the desired joint states for the next instant θ^d_{k+1}. Finally, the desired joint speed for each joint $\dot{\theta}^d_{k+1}$ is computed and published.

Manipulating/Grasping Mode: In Manipulating/Grasping mode, not only the position of the end effector but also its orientation follows those of a human hand. In this mode, closing-hand gesture instead triggers grasping behaviour of the robotic gripper.

[9] http://wiki.ros.org/ur_kin_py.

Algorithm 2. ROS node: Control for Manipulating

```
1: while ROS is not shutdown do
2:     if control_mode = 'manipulating/grasping' then
3:         // Get the current status
4:         $\theta_k \leftarrow$ joint_states
5:         ${}^6_0\mathbf{B}_k \leftarrow$ kin.forward($\theta_k$)
6:         // Position control
7:         $\Delta\mathbf{p} \leftarrow$ palm_position$_k$ - palm_position$_{k-1}$
8:         $\mathbf{p}^d_{k+1} \leftarrow \mathbf{p}_k + \Delta\mathbf{p}$
9:         // Orientation control
10:        thumb_direction$_k \leftarrow$ palm_normal$_k$ $\times$ palm_direction$_k$ (for left hand)
11:        $\mathbf{H}_k =$ [palm_normal$_k$, palm_direction$_k$, thumb_direction$_k$]
12:        Get the transformation matrix $\mathbf{T}_{\mathbf{H}_{k-1}\rightarrow\mathbf{H}_k} \leftarrow \mathbf{H}_k \cdot \mathbf{H}^{-1}_{k-1}$
13:        $\mathbf{E}^d_{k+1} \leftarrow \mathbf{T}_{\mathbf{H}_{k-1}\rightarrow\mathbf{H}_k} \cdot \mathbf{E}_k$
14:        // Command
15:        Construct ${}^6_0\mathbf{B}^d_{k+1}$ using $\mathbf{p}^d_{k+1}$ and $\mathbf{E}^d_{k+1}$
16:        $\theta^d_{k+1} \leftarrow$ kin.inverse(${}^6_0\mathbf{B}^d_{k+1}$)
17:        $\dot{\theta}^d_{k+1} = (\theta^d_{k+1} - \theta_k)/\Delta t$
18:        joint_speed $\leftarrow \dot{\theta}^d_{k+1}$
19:        Publish joint_speed and sleep $\Delta t$.
20:    end if
21: end while
```

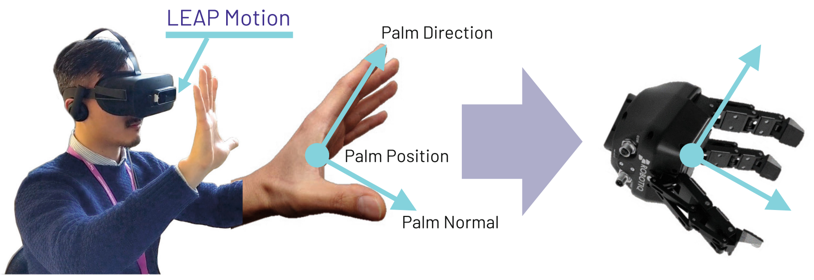
Algorithm 2 shows how the ROS node for the robotic manipulator works in more details. Compared with Algorithm 1, additionally included are Lines 10–13 for orientation control. At first, it obtains the user hand's current orientation $\mathbf{H}_k \in \mathbb{R}_{3\times3}$ using `palm_normal` and `palm_direction` from topic `leap_motion_input`, and `thumb_direction`, which is the cross product between the two vectors. Then, it calculates $\mathbf{T}_{\mathbf{H}_{k-1}\rightarrow\mathbf{H}_k}$, the transformation matrix from the previous hand's orientation $\mathbf{H}_{k-1}$ to the current one $\mathbf{H}_k$. Using the transformation matrix, the desired orientation of the end effector $\mathbf{E}^d_{k+1}$ can be obtained. ${}^6_0\mathbf{B}^d_{k+1}$ is then constructed using $\mathbf{E}^d_{k+1}$ along with the desired position $\mathbf{p}^d_{k+1}$, and eventually transformed to the desired joint speed for each joint $\dot{\theta}^d_{k+1}$, which is then published to the robotic manipulator.

The ROS node for grasping is described in Algorithm 3. It computes the time difference of the hand grabbing position `grab_speed`$_k$ using the current `grab_strength`$_k$ from topic `leap_motion_input` and the one at the previous time instant. Then, `grab_speed`$_k$ and `grab_strength`$_k$ are used to set the gripper's desired position and speed (i.e. `rPRA`, `rSPA`, etc.), respectively. Since the values of `rPRX` and `rSPX` are integers from 0 to 255, it is required for `grab_speed`$_k$ and `grab_strength`$_k$ to be mapped appropriately, for which function f_P and f_S are used. Finally, those values are comprised of topic `SModelRobotOutput`, which is then published to the gripper.

Algorithm 3. ROS node: Control for Grasping

```
1:  while ROS is not shutdown do
2:      if control_mode = ‘manipulating/grasping’ then
3:          grab_speed$_k$ ← abs(grab_strength$_k$ − grab_strength$_{k-1}$)
4:          if grab_strength$_k$ ≥ $c_2$ then
5:              rPRA, rPRB, rPRC ← $f_P$(grab_strength$_k$, $c_2$)
6:              rSPA, rSPB, rSPC ← $f_S$(grab_speed$_k$)
7:          else
8:              rPRA, rPRB, rPRC, rSPA, rSPB, rSPC ← 0
9:          end if
10:         Construct and publish SModelRobotOutput and sleep $\Delta t$.
11:     end if
12: end while
```

In summary, the hand gestures and their resultant commands towards the robot are shown in Fig. 3.



Control Mode \ Hand gesture	Closing Hand	Moving Hand
Reaching Mode	Movement activated	Gripper moves (when activated)
Manipulating/Grasping Mode	Gripper closes	Gripper moves

Fig. 3. How to control the robotic manipulator through hand tracking

4 Demonstration and Discussion

As shown in Fig. 4, we successfully demonstrated the proposed teleoperation system to pick up an empty aluminium canister[10]. Hand position and gestures can be sensed very well, which provides a high degree of intuitiveness. Thanks to this intuitiveness, it was mentioned by the test operator that the multiple plane displays were enough to perceive the remote situation and accomplish the task.

[10] Watch also the video: https://youtu.be/lu-0yrl9J5g.

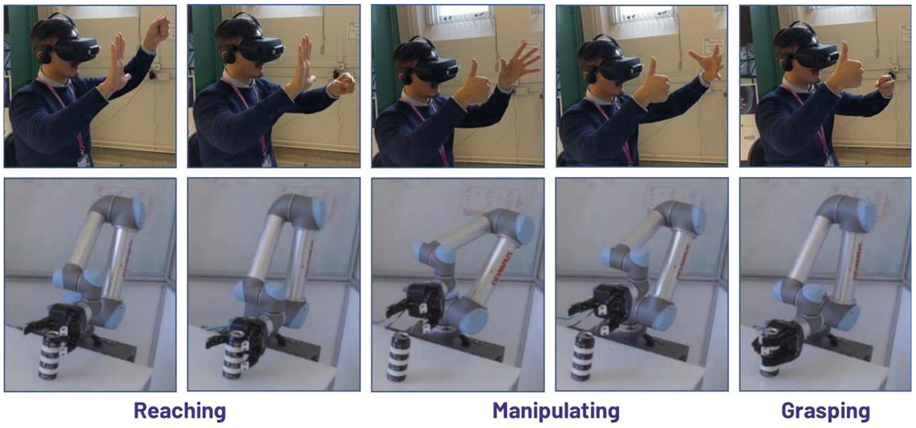


Fig. 4. Demonstration of the proposed teleoperation system: the first two subfigures present Reaching mode; the rest subfigures show Manipulating/Grasping mode

In this context, as future work, we will evaluate this HRI concept by participating through human-case studies, compared with the existing one using a virtual target object. One of our hypotheses is that, relying on multiple plane displays only, the proposed HRI concept will provide more benefits as we experienced in this demonstration.

Obviously, the current system based on LM does not inherently provide any haptic feedback, which is considered as an important feature for teleoperation in many cases. Even though we tried to use a commercial-grade exoskeleton haptic-force-feedback glove (i.e. *CyberGrasp*), according to our test, a hand wearing the haptic glove or occluded by even a small object is not able to be sensed by a LM. In this context, it will be a valuable future work to explore how to provide haptic feedback to LM users. Firstly, we might be able to use multiple LMs as in [8], where two sensors are located to view orthogonal aspects of a human hand. However, even use of this approach is challenging due to the fact that the haptic-force-feedback glove almost fully covers the back of a hand and thus the palm should be always oriented towards the auxiliary LM to be sensed clearly, which limits the user's operational range. Without such force-feedback gloves, we possibly could use virtual haptics based on visual or audio feedback. Alternatively, an artificial haptic device can be used such as *Ultrahaptics*[11], which uses ultrasound to provide mid-air haptics. Otherwise, it would be also an interesting research to design a wearable light-weight haptic device compatible to LM.

[11] https://www.ultrahaptics.com/products-programs/.

5 Conclusion

This paper presented a novel teleoperation system where a robotic manipulator simply and directly copies a human operator's bare hand posture and gesture in a real-time manner without having any hand-held devices. We showed how to practically setup the system using commercial-grade robots such as Universal Robots UR5 and Robotiq 3-finger gripper, and a Leap Motion and Virtual Reality, based on Unity and ROS. We demonstrated our system executing a simple pick-and-place task, and discussed about possible benefits and costs of this HRI concept.

References

1. Allspaw, J., Roche, J., Lemiesz, N., Yannuzzi, M., Yanco, H.A.: Remotely teleoperating a humanoid robot to perform fine motor tasks with virtual reality. In: Waste Management Symposium (WM 2018), Phoenix, AZ (2018)
2. Cancedda, L., Cannavò, A., Garofalo, G., Lamberti, F., Montuschi, P., Paravati, G.: Mixed reality-based user interaction feedback for a hand-controlled interface targeted to robot teleoperation. In: De Paolis, L.T., Bourdot, P., Mongelli, A. (eds.) AVR 2017. LNCS, vol. 10325, pp. 447–463. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-60928-7_38
3. Chae, J., Jin, Y., Sung, Y., Cho, K.: Genetic algorithm-based motion estimation method using orientations and EMGs for robot controls. Sensors **18**(2), 183 (2018). https://doi.org/10.3390/s18010183
4. Chen, S., Ma, H., Yang, C., Fu, M.: Hand gesture based robot control system using leap motion. In: Liu, H., Kubota, N., Zhu, X., Dillmann, R., Zhou, D. (eds.) ICIRA 2015. LNCS (LNAI), vol. 9244, pp. 581–591. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22879-2_53
5. Crick, C., Jay, G., Osentoski, S., Pitzer, B., Jenkins, O.C.: Rosbridge: ROS for non-ROS users. Springer Tracts Adv. Robot. **100**, 493–504 (2017)
6. Hawkins, K.P.: Analytic inverse kinematics for the universal robots UR-5/UR-10 arms. Technical report, Georgia Institute of Technology (2013). https://smartech.gatech.edu/bitstream/handle/1853/50782/ur_kin_tech_report_1.pdf
7. Hokayem, P.F., Spong, M.W.: Bilateral teleoperation: an historical survey. Automatica **42**(12), 2035–2057 (2006). https://doi.org/10.1016/j.automatica.2006.06.027
8. Jin, H., Chen, Q., Chen, Z., Hu, Y., Zhang, J.: Multi-LeapMotion sensor based demonstration for robotic refine tabletop object manipulation task. CAAI Trans. Intell. Technol. **1**(1), 104–113 (2016). https://doi.org/10.1016/j.trit.2016.03.010. https://linkinghub.elsevier.com/retrieve/pii/S2468232216000111
9. Krupke, D., Einig, L., Langbehn, E., Zhang, J., Steinicke, F.: Immersive remote grasping: realtime gripper control by a heterogenous robot control system. In: Proceedings of the ACM Symposium on Virtual Reality Software and Technology, VRST 02–04-Nove, pp. 337–338 (2016). https://doi.org/10.1145/2993369.2996345
10. Kruusamae, K., Pryor, M.: High-precision telerobot with human-centered variable perspective and scalable gestural interface. In: Proceedings - 2016 9th International Conference on Human System Interactions, HSI 2016, pp. 190–196 (2016). https://doi.org/10.1109/HSI.2016.7529630

11. Li, C., Yang, C., Wan, J., Annamalai, A.S.S., Cangelosi, A.: Teleoperation control of Baxter robot using Kalman filter-based sensor fusion. Syst. Sci. Control Eng. **5**(1), 156–167 (2017). https://doi.org/10.1080/21642583.2017.1300109
12. Lipton, J.I., Fay, A.J., Rus, D.: Baxter's homunculus: virtual reality spaces for teleoperation in manufacturing. IEEE Robot. Autom. Lett. **3**(1), 179–186 (2018). https://doi.org/10.1109/LRA.2017.2737046
13. Makris, S., et al.: Dual arm robot in cooperation with humans for flexible assembly. CIRP Ann. **66**(1), 13–16 (2017). https://doi.org/10.1016/j.cirp.2017.04.097
14. Nuño, E., Basañez, L., Ortega, R.: Passivity-based control for bilateral teleoperation: A tutorial. Automatica **47**(3), 485–495 (2011). https://doi.org/10.1016/j.automatica.2011.01.004
15. Pancake, D., et al.: A novel and cost effective approach to the decommissioning and decontamination of legacy glove boxes - minimizing TRU waste and maximizing LLW waste - 13634. In: Waste Management Symposium (WM 2013), Phoenix, AZ (2013)
16. Peppoloni, L., Brizzi, F., Avizzano, C.A., Ruffaldi, E.: Immersive ROS-integrated framework for robot teleoperation. In: 2015 IEEE Symposium on 3D User Interfaces (3DUI), pp. 177–178. IEEE, March 2015. https://doi.org/10.1109/3DUI.2015.7131758
17. Roldán, J.J., et al.: Multi-robot systems, virtual reality and ros: developing a new generation of operator interfaces. In: Koubaa, A. (ed.) Robot Operating System (ROS). SCI, vol. 778, pp. 29–64. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91590-6_2
18. Tang, G., Webb, P.: The design and evaluation of an ergonomic contactless gesture control system for industrial robots. J. Robot. (2018). https://doi.org/10.1155/2018/9791286
19. Whitney, D., Rosen, E., Phillips, E., Konidaris, G., Tellex, S.: Comparing robot grasping teleoperation across desktop and virtual reality with ROS reality. In: International Symposium on Robotics Research, pp. 1–16 (2017)