2019 19th International Conference on Advanced Robotics (ICAR)
Belo Horizonte, Brasil, December 02-06, 2019

Towards a Virtual Reality Interface for Remote Robotic Teleoperation

Abdeldjallil Naceri, Dario Mazzanti, Joao Bimbo, Domenico Prattichizzo, Darwin G. Caldwell, Leonardo S. Mattos, and Nikhil Deshpande

***Abstract*— Intuitive interaction is the cornerstone of efficient and effective task execution in remote robotic teleoperation. It requires high-fidelity in control actions as well as perception (vision, haptic, and sensory feedback) from the remote environment. This paper presents *Vicarios*, a VR-based teleoperation interface with the aim of facilitating intuitive real-time remote teleoperation, while utilizing the inherent benefits of VR, including immersive visualization, freedom of viewpoint selection, and fluidity of interaction through natural action interfaces. The overall architecture of *Vicarios* is described, with its components and framework. A preliminary comparative user study, with a real-world tele-manipulation task, was conducted to quantify the effectiveness of *Vicarios*. It is shown that the VR-based interface is easy-to-use and learn, with an overall learning rate of 0.93 sec. per trial for the users. The smoothness metric shows that the users can improve on their task performance over time as well. *Vicarios* allows the intuitiveness and flexibility to maintain efficiency in tele-robotic tasks. It forms the basis for an immersive perception interface in remote robotic teleoperation.**

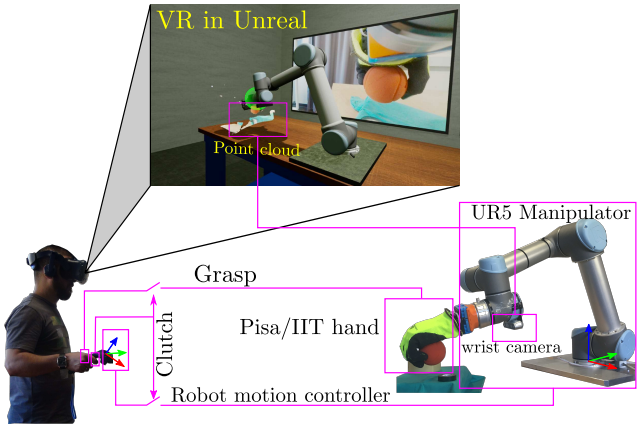


Fig. 1: *Vicarios* Interface: A user teleoperating a remote robotic arm using the HTC Vive Pro VR system. Unreal VR graphics engine provides immersive visualization. Remote camera provides video and depth feedback in real-time.

I. Introduction

State-of-the-art systems in remote robotic teleoperation are able to execute complex, yet dexterous, tasks in demanding environments, e.g., nuclear decommissioning, disaster response, search-n-rescue, surgery, etc. The traditional user interfaces though, relying mostly on stereo video feedback from the remote environment, suffer from limitations including accessibility, adaptability of viewpoint, improper mapping between operator and robot motions, unreliable video communications, two-dimensional displays, cybersickness due to inconsistent visual feedback, among others [1]. In contrast, modern virtual reality (VR) interfaces, owing to the video-gaming community, have seen important technological advances in graphics engines and devices [2] in terms of immersion, natural physical control devices, high-fidelity graphical renderings, and native viewpoint changes for near-natural visualizations. Utilizing VR in robotics, therefore, seems logical, and has been recognized as such for its inherent benefits [3]–[5].

Building on the above characteristics, this article presents a systematic approach for utilizing VR-based user interfaces in modern telerobotic applications. *Vicarios*, a VR-based interface for remote robotic teleoperation, facilitates real-time teleoperation of robotic platforms, while exploiting the immersiveness and high-fidelity of VR for real-time feedback, in an intuitive interface. The key idea involves: (i) the stereoscopic virtual rendering of the real remote scene, including models of the remote robotic platforms, coupled with (ii) the real-time video streaming feedback from the remote environment, and (iii) mapping of the user gestures/actions to the remote devices. This provides situational awareness to the operator allowing seamless execution of remote tasks. The integrated *Vicarios* interface, seen in Fig. 1, forms the basis of a framework that is extensible and adaptable to different teleoperation scenarios. The following sections outline the contribution of this research in detail.

*This research is funded by the Italian Workers' Compensation Authority (INAIL).

All authors are with Advanced Robotics, Istituto Italiano di Tecnologia (IIT), Via Morego, 30, 16163 Genova, Italy. abdeldjallil.naceri@iit.it

II. Related work

Researchers have long seen the advantages of using 3D graphics to overcome limitations of 2D displays, to better assist in robot motion planning and help operator training [5], [6]. The more recent investigations in improved human-robot interaction have focused on using affordable VR devices (e.g., [3]), integrating the VR graphics engine softwares (e.g., Unity3D, Unreal) with compatible robotic hardware, with the robotic software of choice being the robot operating system (ROS) [7]. Below, the most recent works in this field are discussed.

Peppoloni et al. [4], were one of the first groups to propose the concept of combining ROS-compatible robotic platforms and high performance VR interfaces for teleoperation. The authors combined the platforms with the Oculus Rift and the Leap Motion controller at the operator station, and the Kinect sensor for 3D point-cloud feedback from the

remote scene. Subsequently, several research groups have explored VR-based teleoperation interfaces. A similar approach was adopted by [8] for a VR-teleoperation interface, using Unity3D, ROS, and Rosbridge for inter-communication. Several robot models were imported in the VR environment and interaction was implemented through gesture-tracking using the Leap Motion controller, mapping the motion in virtual and real world. Lipton et al. [3] proposed a similar approach for bimanual manipulation, based on the Baxter robot, Unity3D, Razer Hydra hand trackers, and ROS. The user uses the VR world as the primary interface for interaction, and uses the real-time video stream from the real-world scene (shown in a customized window) for guidance. Another VR interface was introduced by [9] to allow users to interact with industrial robots via gestures, for trajectory planning. The authors allowed users to define the robot trajectory in VR, in offline mode, to be executed by the real robot later, in a more supervisory form of interface. A more integrated approach was recently presented by Whitney et al. [10], where the imported robot meshes are rendered through the VR-headset (Unity3D), whereas the remote scene and objects are rendered as a point-cloud from the remote depth camera. As noted, combining ROS-enabled robotic platforms with VR graphics engines and command controllers is quickly becoming a standard approach. The research in this paper builds on such principles adopted by [3], [4], [8], [10], to advance the state-of-the-art in VR-based interfaces for teleoperation tasks. In this work, the *Vicarios* VR interface for remote robotic teleoperation is proposed, with the following contributions:

- One-to-one rendering of the remote environment in VR for immersive visualization,
- Fluid mapping between operator and remote robot motion, independent of operator viewpoint,
- Real-time point-cloud visualization for improved depth perception of remote scene,
- Real-time video streaming feedback for first-person view of the scene

The proposed interface is evaluated through a preliminary comparative user study, quantifying the effects of introducing such an interface on participants' performance in representative remote teleoperation tasks.

III. *Vicarios* PLATFORM DESCRIPTION

The overall *Vicarios* setup, seen in Figure 1, follows the general teleoperation setup, which includes a user, a remote environment, and a visualization interface. The proposed framework, as shown in Figure 2, is divided into three major parts: the **operator site**, the **remote environment**, and a **communication network** between them. The gesture/motion controllers at the operator site convey the commands to the remote robots. The remote robots and the environment are rendered virtually in the VR interface.The communication network allows real-time data exchange between the operator site and remote environment, i.e., sending commands, receiving remote robot status, and receiving real-time video and point-cloud information. The different components are described below.

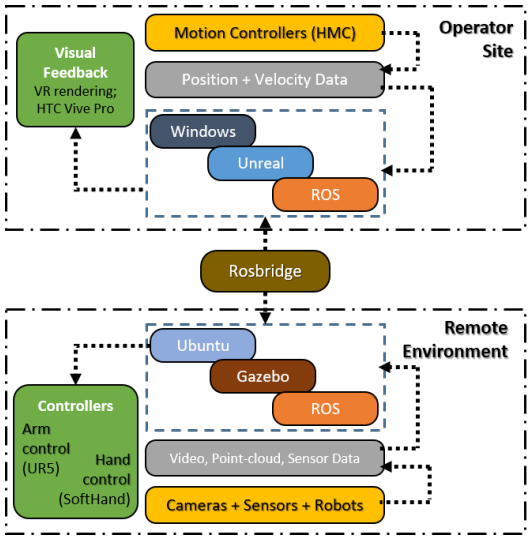


Fig. 2: Schema shows an overview of the *Vicarios* software and hardware architecture.

A. *OPERATOR Site*

1) The HTC Vive Pro VR module: consists of head-mounted display (HMD) unit, with a display resolution of 2880 x 1600 with a 110-degree field-of-view and 90 Hz refresh rate. It includes its own proprietary Lighthouse tracking system for accurate head-motion tracking.

2) The Unreal graphics engine: is a state-of-the-art framework for 3D environments and interactive experiences. It facilitates the inclusion of custom development libraries, and includes a visual programming tool to develop complex behaviours. The engine facilitates creating dedicated environments and display unconventional visual elements with custom shaders, and many other features that helps users comfort and immersion within VR. The engine is used on the Windows 10 operating system.

3) Command interfaces: are the HTC Vive Pro motion controllers (HMC), giving the user the flexibility to teleoperate the remote robots in real-time using an ungrounded motion controller interface.

The HMC are tracked in space allowing the exploration of the VR environment as well as natural interaction with the remote robots. Three buttons on the HMC are used for:

a) triggering the clutch action to engage / disengage the motion between the HMC and the remote robot. This overcomes the range limitation of the HMC as compared to that of the remote robot,
b) triggering the grasp action to open/close the remote robot hand end-effector
c) triggering the viewpoint changes as desired by the operator.

4) Rendering virtual robots and environment: The Unreal engine displays the virtual robot 3D models and the remote environment. Unreal allows to import, display, and animate

the robot models, complete with their links and joints hierarchy. Robot models are imported as URDF (Universal Robot Description Framework) files, based on an open C++ library [11]. A dedicated robot message controller parses messages between the remote environment and the operator site, allowing real-time, one-to-one mapping between the motions of the real and the virtual robots.

B. *REMOTE Environment*

1) ROS: The Robot Operating System (ROS) [7] is the defacto standard for robot programming and its node-based message-passing structure is well-suited to the client-server architecture which is native to any teleoperation setup. ROS serves as the framework between the operator VR interface and the real remote robots, and it is implemented in the Ubuntu 16.04 operating system.

2) Real-world environment: At this stage, it consists of the UR5 robot and the Pisa/IIT SoftHand [12]. The UR5 is a 6 degrees-of-freedom (DOF) manipulator, able to move at up to 1 m/s and each joint at 180 deg/s. It has a payload of 5 kg and a repeatability of 0.1 mm. The Pisa/IIT SoftHand is an underactuated soft robotic hand with 19 joints, but uses only one actuator to activate its adaptive synergy [13], [14]. The actuator is driven by commanding a grasping variable $g_r \in [0, 1]$, which drives the hand to open or close.

3) Perception devices: Cameras mounted in the remote environment stream real-time video and point-cloud data to the VR interface. At this stage, the Intel Realsense RGB-D camera was used for this purpose. The Realsense includes an RGB camera (1920 x 1080 @ 30 Hz) and an IR sensor for depth estimation (1280 x 720 @ 90Hz; depth distance of 0.2 - 10 m). Virtual screens positioned within the VR environment provide the video-feed.

C. *COMMUNICATION Network*

Within the *Vicarios* interface, the VR graphics engine on Windows communicates with the remote environment on Linux via Rosbridge [15] (by means of ROS messages) and UDP sockets. Within Unreal, starting from the open source C++ library [11], the inherent ROS architecture was encapsulated in dedicated classes and Unreal Blueprints, making it more convenient to connect to a ROS server, to add ROS publishers or subscribers, and add/modify ROS messages and types.

The UDP sockets allow the streaming of video-feed, depth data, and other sensory information from the remote environment. A distinct advantage of using VR as an interface is that it does not require more communication bandwidth than its traditional counterparts. The heavy-lifting of the scene-rendering and meshes is done at the operator site itself, without relying on any communication channel.

IV. *Vicarios* - VISUALIZATION AND MAPPING

The *Vicarios* interface consists of the rendered models of the remote robots and environment, the real-time video and point-cloud stream, and the motion control devices to command the remote robots.

A. *Visualization*

The visualization in the HMD of the remote scene is shown in Figure 1. The flexibility of the VR environment is evident in the arrangement of the room, the placement of the camera-feed and the point-cloud data, and the ability to change viewpoint as desired. The VR scene closely replicates the remote environment, either with prior knowledge of the environment, or through real-time object pose information using ROS messages, or through real-time capturing and rendering of the 3D remote environment. The user can focus on the task execution, and only refer to the virtual screens (video-feed) intermittently as required for assistance.
The point-cloud data is presented in an intuitive manner, projected in the virtual scene, registered to the real pose of the remote camera. A form of *augmented virtuality*, this allows users to infer the relative pose of the objects in the remote environment with respect to the robot end-effector.

B. *Motion mapping*

The user motion commands from the HMC control the motion of the remote robots by triggering the respective control algorithms for the robots (e.g., Jacobian, Inverse Kinematics), which compute the joint angles, velocities, and accelerations to be sent to the remote robotic platforms. The joint-states of the remote robots are in-turn communicated back to the Unreal engine to update the positions of the rendered robot models. This integration implies that the user actually commands the motion of the virtual robot and the real robot's motion is mapped one-to-one to this motion.

Since the kinematics of the remote robot and the operator controllers are non-homothetic, the 6-DOF pose of the robot is commanded using velocity-control, i.e., the velocity of the HMC is mapped to the robot. The robot pose is commanded by finding the position error $\boldsymbol{e} \in SE(3)$ between the current and the desired robot pose. Then, the damped least-squares solution to the problem in equation (1) iteratively finds the change in joint angles $\Delta\boldsymbol{q}$ that minimizes the error $\boldsymbol{e}$ [16].

$$\Delta\boldsymbol{q} = (\boldsymbol{J}^T\boldsymbol{J} + \lambda \cdot \boldsymbol{I})^{-1}\boldsymbol{J}^T\boldsymbol{e}, \tag{1}$$

J is the 6-DOF jacobian of the robot, and $\lambda \in R$ is a non-zero damping constant. Finally, a proportional controller $\dot{\boldsymbol{q}} = K_p \cdot \Delta\boldsymbol{q}$ sets the joint velocities to reach the desired pose. For the implementation, λ was set to 0.001 and K_p to 0.6.

C. *Viewpoint-independent Motion Mapping*

The aforementioned approach allows viewpoint-independent mapping between the gestures of the HMC and the motion of the remote robots. This is an important capability facilitated by the *Vicarios* interface, to let the operator freely change the viewpoint, without worrying about re-mapping their gestures. The ability to observe the remote scene from different angles improves the accuracy and efficiency when doing complex remote tele-manipulation tasks.

To allow such mapping, the following method is adopted. When the *clutch* button of the HMC is pressed, i.e., the

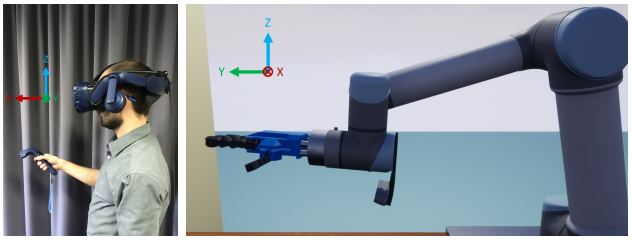
motion between the HMC and the remote robot is engaged, the pose of the HMC, $\boldsymbol{h}(t_0)$, and the robot, $\boldsymbol{r}(t_0)$, are saved. When the user moves their hand to new a pose $\boldsymbol{h}(t_1)$, the difference between the $\boldsymbol{h}(t_0)$ and $\boldsymbol{h}(t_1)$, i.e., $\Delta\boldsymbol{h}$, is calculated. $\Delta\boldsymbol{h}$ is then transformed to the robot base frame, scaled, and added to $\boldsymbol{r}(t_0)$ to obtain the desired robot pose. The equations are as follows:

$$\begin{aligned} \Delta\boldsymbol{h}^o &= \boldsymbol{h}^o(t_1)\cdot\boldsymbol{h}^o(t_0)^{-1} \\ \Delta\boldsymbol{h}^p &= \boldsymbol{h}^p(t_1) - \boldsymbol{h}^p(t_0) \end{aligned} \quad , \tag{2}$$

where $\{\cdot\}^o \in \mathbb{R}^{3\times3}$ is the orientation matrix and $\{\cdot\}^p \in \mathbb{R}^{3\times1}$ is the position vector. The mapping to the robot frame is done by adding the desired transformation ${}^W\Delta\boldsymbol{h}$ in the global frame W to the robot's current pose $\boldsymbol{r}(t_0)$, obtaining the new desired pose $\boldsymbol{r}(t_1)$:

$$\begin{aligned} \boldsymbol{r}^o(t_1) &= {}^W\boldsymbol{R}_R^{-1}\cdot{}^W\Delta\boldsymbol{h}\cdot{}^W\boldsymbol{R}_R\cdot\boldsymbol{r}(t_0) \\ \boldsymbol{r}^p(t_1) &= \boldsymbol{r}(t_0)\cdot\left({}^W\boldsymbol{R}_R\cdot\boldsymbol{r}(t_0)\right)\cdot{}^W\Delta\boldsymbol{h}^p \end{aligned} \quad , \tag{3}$$

where ${}^W\boldsymbol{R}_R$ is the rotation matrix from the world frame W to the robot frame R. Figure 3 demonstrates the concept with sample user motions mapped to the robot according to the chosen viewpoint, without requiring the user to re-orient their gestures. The viewpoints can have different positions and orientations in 3D space.



(a) User Motions (b) Side View

(c) Front View (d) Top View

Fig. 3: Viewpoint-independent motion mapping - user motions are mapped to the robot according to the chosen viewpoint.

Furthermore, the coordinate frames in ROS and Unreal are not aligned and therefore require a transformation. For positions, the right-handed coordinate frame of ROS is transformed to a left-handed one in Unreal, with a scaling factor of 100 for dimensional conformity. For orientations, the $x-$ and $z-$ axes of ROS are inverted for Unreal.

Although velocity-based teleoperation allows real-time control between non-homothetic devices, it limits the range of motion to that of the motion control device. This issue is easily solved by using a *clutch*-based system, mentioned earlier. The teleoperation control between the master controllers and the remote robots can be coupled and decoupled as required to change the workspace of the user. This allows the user to pause and resume the robot control as desired, to avoid complicated gestures and allow arm posture comfort.

V. Preliminary Evaluation

The proposed system, thus becomes suitable for different teleoperation scenarios. A preliminary evaluation of *Vicarios* was conducted in order to understand its suitability for remote teleoperation tasks. Through a comparative user study, the aim is to quantify users' task execution times and efficiency, when working with the *Vicarios* interface.

The *Vicarios* platform was tested in a real-world pick-n-place task (grasp and replace a tennis ball). The remote environment consisted of:

- a bench with the tennis ball and two cylindrical bases for pick-up and replacement;
- the UR5 robot with the Pisa/IIT SoftHand end-effector, mounted on the bench;
- The Intel RealSense RGB-D camera attached to the UR5 wrist, close to the end-effector.

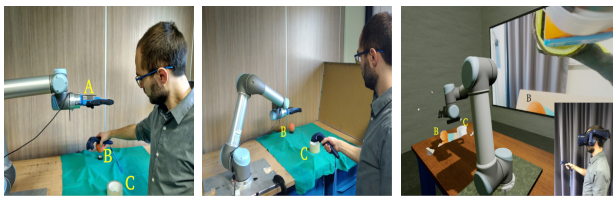
For the evaluation, the remote environment was rendered in VR, maintaining the poses of the bench, the UR5 robot with end-effector, and the camera. The ball and the cylindrical bases, to be visualized in the video and point-cloud feeds, were not rendered in VR. The task procedure is described below. When using the *Vicarios* interface, users were encouraged to change their visualization viewpoint while performing the task.

A. Participants

Six male participants (Age: 29 ± 4) took part in our evaluation study. All participants were right-handed and five were naive to the study, having no prior experience with the *Vicarios* platform.

The participants were instructed to use the three buttons on the HMC: (i) button 1 as the *clutch*; (ii) button 2 to open and close the hand (*grasp*); and (iii) button 3 for changing the *viewpoint* in the VR scene.

B. Task Procedure



(a) Manual (b) Teleop (c) VR-teleop

Fig. 4: Experimental scenes in different conditions. A: starting location. B: picking location. C: placing location

At the "Go" signal, starting from point "A", the participant picked up the tennis ball located at point "B" and placed it inside a cylindrical base located at point "C" (Refer Fig. 4).

In order to understand the impact of the *Vicarios* interface in teleoperation tasks, a comparative evaluation was done, where the task was performed in three different conditions:

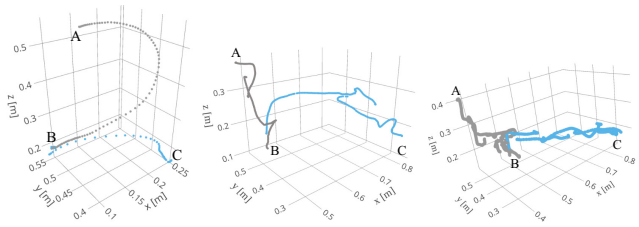
1) *Manual*: (baseline condition), participants performed the task using their hand (neither the robot nor the *Vicarios* interface was involved). They were asked to place their hand at point "A" (Fig. 4), as a starting point for each trial. The hand of the participants was tracked using the HTC Vive trackers (Fig. 4a).
2) *Teleop*: participants were asked to perform the task, teleoperating the robot while looking at it directly, being in the same environment (the *Vicarios* interface was not involved here). The participants used the HMC to command the robot, using buttons 1 & 2 only.
3) *VR-teleop*: participants performed the task while teleoperating the robot and visualizing the scene through the *Vicarios* interface, using all 3 buttons on the HMC.

The order of the experimental conditions was randomized for each participant to avoid the learning effect across conditions. Each participant conducted five trials in each condition.

C. Analysis Metrics

We quantified the robot end-effector trajectories and task completion times as dependent variables of the experiment in the *Teleop* and *VR-teleop* conditions.

Figure 5 shows movement trajectories during the task from a representative subject. To distinguish among the trajectory profiles in the three conditions, a trajectory smoothness measure is introduced here. We compute the 3-dimensional motion speeds for all participants in all conditions, and then count the number of peaks in each profile. Each peak would represent one motion gesture from the user, start-to-stop. For instance, in the *Manual* condition, as seen in Fig. 6, if a participant is able to complete the task in a single smooth gesture, from point "A" to "B", and then point "B" to point "C", without jitter or stopping in between, the corresponding speed profile would show a simple bell-shape curve with two peaks for the two motions. More bells in the profile, i.e., more peaks, would imply more start-stop motions by the user (Fig. 6). Therefore, the number of peaks in the speed profile gives a measure of smoothness of the trajectory in performing the task.



(a) Manual (b) Teleop (c) VR-teleop

Fig. 5: Representative user trajectories in the three experimental conditions. Blue dots represent data when the tennis ball is grasped, otherwise, Gray dots.

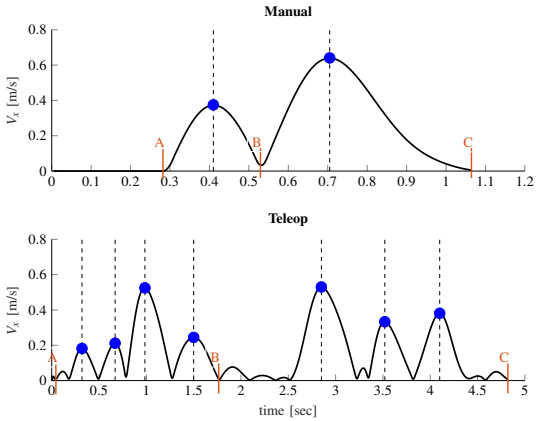


Fig. 6: Speed profile for a representative participant, demonstrating the different peaks in *Manual* and *Teleop* conditions.

VI. Results

Results of the user trials revealed that participants successfully achieved the pick-n-place task in both the dependent conditions, i.e., *Teleop* and *VR-teleop*.

For the completion times seen in Fig. 7, participants recorded 13.69 ± 2.33 sec. (Mean $\pm$ confidence intervals) during *VR-teleop* condition, 8.66 ± 0.99 sec. in the *Teleop* condition, and 3.64 ± 1.05 sec. in the *Manual* condition. We evaluated participants' learning rate across trials for completion times: in *Teleop* condition, the average rate was 0.15 sec per trial, while in *VR-teleop* condition, the average rate was 0.93 sec per trial.

For the smoothness measure, Fig. 8 shows number of peaks in each condition in $x-$, $y-$, and $z-axes$. We recorded a higher number of sub-movements in the *VR-teleop* condition, as compared to both the *Teleop* and *Manual* condition. Participants used more start-stop gestures, and therefore, did not have smooth trajectories in the *VR-teleop* condition.

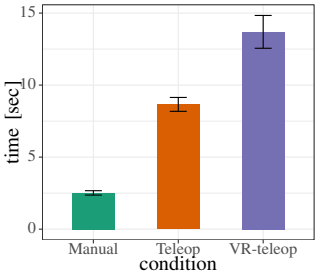


Fig. 7: Completion time in different conditions. Error bars represent confidence intervals.

VII. Discussion and Future Work

In this paper, the *Vicarios* VR interface was presented, forming the basis for an immersive and intuitive remote robotic teleoperation interface. It integrates: (i) the Unreal graphics engine with the HTC Vive Pro VR system; (ii) robotic platforms for manipulation in remote environments,

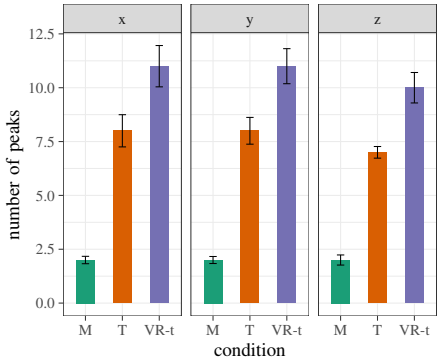


Fig. 8: Barplot for overall number of peaks in each condition for the three axes. Error bars indicate confidence intervals. M: Manual, T: Teleop and VR-t: VR-teleop.

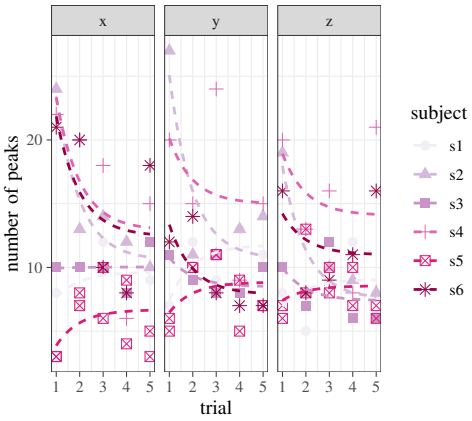


Fig. 9: Number of peaks per trial per participant in the VR-teleop condition, with exponential fitting for the learning rate

with RGB-D cameras for real-time perception; (iii) the robot operating system (ROS) as the control software, with Rosbridge as the inter-communication layer; and (iv) natural gesture controllers for commanding the remote robots in real-time.

The pertinent question therefore becomes, "how does the teleoperation interface itself impact the user performance?" Researchers note that the performance achieved by sophisticated remote tele-robotic systems is poorer and slower as compared to direct human manual work, on the order of ten-to-hundreds of times [17]. The results of this user study showed that, in comparison with the *Manual* condition, the participants were twice slower in *Teleop* condition, and four times slower in *VR-teleop* condition, but without any failed attempt. The different learning rates demonstrate that the users needed to familiarize with the manipulation DOFs as well as the new components within the *Vicarios* interface. In terms of efficiency, the users executed more tortuous trajectories to complete the task when teleoperating through *Vicarios* (Fig. 8). The users have made more gestures to achieve the goal, requiring more mental effort while observing the VR scene from different viewpoints before completing the task. It is evident that users require time in their earliest trials to get accustomed to new interfaces. Yet, as seen in Fig. 9, on average, the number of peaks reduced over time, showing that users were able to learn and improve their trajectories, using fewer segments, as they got more skilled with the interface.

The interface presented here lays the groundwork for future research. As next steps, *Vicarios* shall be evaluated with respect to the communication latencies in each of its components (e.g., ROS, Unreal VR rendering, network communications, robot controller loops, etc.). Further user trials shall be conducted, in order to analyse the impact of the different *Vicarios* components, i.e., point-cloud, video-feed, stereo view, etc., to better estimate the learning with the interface.

REFERENCES

[1] J. Y. C. Chen, E. C. Haas, and M. J. Barnes, "Human Performance Issues and User Interface Design for Teleoperated Robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, pp. 1231–1245, Nov 2007.

[2] B. Miguel, S. Andrew, C. Brian, S. Trey, and V. Rodrigo, "HTC Vive: Analysis and Accuracy Improvement," in *IEEE/RSJ IROS 2018*, pp. 2610–2615, 2018.

[3] J. I. Lipton, A. J. Fay, and D. Rus, "Baxter's Homunculus: Virtual Reality Spaces for Teleoperation in Manufacturing," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 179–186, 2018.

[4] L. Peppoloni, F. Brizzi, C. A. Avizzano, and E. Ruffaldi, "Immersive ROS-integrated Framework for Robot Teleoperation," in *2015 IEEE Symposium on 3D User Interfaces (3DUI)*, pp. 177–178, 2015.

[5] A. K. Bejczy, "Virtual reality in Robotics," in *1996 IEEE Conference on Emerging Technologies and Factory Automation. ETFA '96*, pp. 7–15 vol.1, 1996.

[6] P. Milgram and J. Ballantyne, "Real World Teleoperation via Virtual Environment Modeling," in *Intl. Conf. on Artificial Reality and Tele-existence (ICAT'97)*, Dec. 3-5 1997.

[7] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *IEEE ICRA Workshop on Open Source Software*, IEEE, 2009.

[8] D. Krupke, S. Starke, L. Einig, J. Zhang, and F. Steinicke, "Prototyping of Immersive HRI Scenarios," in *Proc. 20th Intl. Conf. on CLAWAR 2017*, pp. 537–544, Oct. 2017.

[9] M. Theofanidis, S. I. Sayed, A. Lioulemes, and F. Makedon, "VARM: Using Virtual Reality to Program Robotic Manipulators," in *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments - PETRA 2017*, pp. 215–221, 2017.

[10] 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 *Proc. IEEE IROS 2018*, pp. 5018–5025, Oct. 1-5 2018.

[11] "RobCog-IAI: UE4 based Robot Simulator." Accessed on 28-Feb-2019. [Online] Available: github.com/robcog-iai/URoboSim.

[12] "Pisa/IIT SoftHand Software Package." Accessed on 28-Feb-2019. [Online] Available: github.com/CentroEPiaggio/pisa-iit-soft-hand.

[13] A. Bicchi, M. Gabiccini, and M. Santello, "Modelling Natural and Artificial Hands with Synergies," *Philos. Trans. R. Soc. Lond. B Biol. Sci.*, vol. 366, pp. 3153–3161, Nov. 2011.

[14] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive Synergies for the Design and Control of the Pisa/IIT SoftHand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 768–782, 2014.

[15] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: ROS for non-ROS users," in *in Proceedings of the 15th International Symposium on Robotics Research (ISRR*, 2011.

[16] S. R. Buss, "Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods," tech. rep., University of California, San Diego, USA, 2004.

[17] J. P. Trevelyan, S.-C. Kang, and W. R. Hamel, "Robotics in hazardous applications," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), pp. 1101–1126, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.