DOI 10.15622/ia.20.6.1

A. NAKAYAMA , D. RUELAS , J. SAVAGE , E. BRIBIESCA

TELEOPERATED SERVICE ROBOT WITH AN IMMERSIVE MIXED REALITY INTERFACE

Nakayama A., Ruelas D., Savage J., Bribiesca E. **Teleoperated Service Robot with an Immersive Mixed Reality Interface.**

Abstract. Teleoperated service robots can perform more complex and precise tasks as they combine robot skills and human expertise. Communication between the operator and the robot is essential for remote operation and strongly affects system efficiency. Immersive interfaces are being used to enhance teleoperation experience. However, latency or time delay can impair the performance of the robot operation. Since remote visualization involves transmitting a large amount of video data, the challenge is to decrease communication instability. Then, an efficient teleoperation system must have a suitable operation interface capable of visualizing the remote environment, controlling the robot, and having a fast response time. This work presents the development of a service robot teleoperation system with an immersive mixed reality operation interface where the operator can visualize the real remote environment or a virtual 3D environment representing it. The virtual environment aims to reduce the latency on communication by reducing the amount of information sent over the network and improve user experience. The robot can perform navigation and simple tasks autonomously or change to the teleoperated mode for more complex tasks. The system was developed using ROS, UNITY 3D, and sockets to be exported with ease to different platforms. The experiments suggest that having an immersive operation interface provides improved usability for the operator. The latency appears to improve when using the virtual environment. The user experience seems to benefit from the use of mixed reality techniques; this may lead to the broader use of teleoperated service robot systems.

Keywords: teleoperated robot, service robot, immersive operation interface, mixed reality interface, virtual reality environment

1. Introduction. Service robots can perform more complex and precise tasks by harnessing human experience through teleoperation. Teleoperation systems are intended to provide technical means to perform the desired task in a remote environment [1]. In an efficient teleoperation system, the operator must be ensured that the desired task is adequately performed in the remote environment. To this end, teleoperation systems must overcome a series of barriers such as distance, and time delay, among others. Furthermore, the relationship between the operator and the remote environment strongly affects the system efficiency. This means that system performance can be vastly improved by allowing the operator to understand the interaction with the remote environment intuitively and easily.

The feeling of presence refers to the operator's sensation of being present in the remote environment [2]. Ideally, the system should make the operator feel the remote environment as if it were the real world. There is a belief that presence or embodiment is correlated with task performance positively [3] so, by improving the feeling of presence, task performance is also improved.

In addition to improved performance, the feeling of presence also gives the operator the ability to perform the task in a more natural way, similar to real-world actions. This can reduce the need for the operator to train extensively to successfully operate the robot.

A typical method of providing the feeling of presence is by displaying video from the robot's camera [4, 5]. Recently, immersive interfaces have been used to induce the feeling of being inside the robot [6] and improve the teleoperation experience [7]. The feeling of presence is the result of immersion [8]. Therefore, immersive interfaces are often accompanied by Head-Mounted Displays (HMD) consisting of a display combined with a head-tracking system to improve immersion and control.

Communication between the operator and the robot is also essential for effective remote operation. Virtual reality (VR) has proven to be an excellent option for displaying information to the operator. VR enhances communication between a system and its users by displaying and visualizing complex data efficiently. VR systems are being used for robotic teleoperation tasks due to their ability to allow users to intuitively interact with 3D environments and concede fluid interaction in the physical world [9] . Gradecki defined VR as a "technology that allows a user to view a virtual environment from any point and angle, and interact with objects that make up that environment" [10].

By enhancing the teleoperation interface with virtual environments, the operator can experience better robot control, allow non-expert users to control the robot, and leverages the experts' experience in challenging domains [11]. However, the integration of robots with VR systems can become a challenging task, as several considerations must be taken into account. For instance, the lack of standard interfaces that connect ROS (Robot Operating System) with virtual reality platforms that use HMD [12], or VR sickness, which are the ailments such as headaches, dizziness, and nausea caused by misuse of the VR environments [13].

Latency or time delay is a key factor in teleoperation systems. The time delay between input and visual feedback response greatly afflicts the communication between distributed master and slave systems over a network [14]. This delay can vary from a few milliseconds to several hundred milliseconds, depending on different factors such as distance or communication infrastructure. Early experiments to test the effect of time delays on teleoperation have shown that latency impairs the performance of the robot operation [15]. More recent studies show that in systems with latency over 1 second, the operator often tries to compensate for the delay with a " move and wait " strategy [16] that makes the robot's operation suboptimal. Much effort has been made to reduce latency, such as algorithms for data compression and optimization.

For example, Bouzakaria and collaborators [17] are trying to reduce network latency and have obtained response times below 240 milliseconds on a local network.

The Internet is becoming the most common communication medium in teleoperation [18] since it offers several advantages such as ease of use, broad accessibility, low cost, and relative reliability. But despite these benefits, this communication technology has its drawbacks, such as latency, loss of data packets, and others [19]. As remote visualization involves transmitting a large amount of video data over the network, the challenge is to decrease communication instability, such as latency, which is one of the main problems in teleoperation [4].

In summary, an efficient teleoperation system must have a suitable operating interface that is not only capable of viewing the remote environment, controlling the robot, and providing a feeling of presence but also has a fast response time.

This work presents the development of a service robot teleoperation system with an immersive mixed reality operation interface where the operator can visualize the robot's actual environment or a virtual 3D environment representing it. Mixed Reality is the concept (not any particular technology) of combining real and virtual worlds [20]. It involves the merging of real and virtual worlds somewhere along the "virtuality continuum" which connects completely real to completely virtual environments [21].

The aim of adding a virtual environment is to reduce communication latency by reducing the amount of information sent over the network and improving the user experience. However, adding the virtual reality environment brought the need to ensure that the task was being performed correctly, so visualization of the real environment through the robot's cameras was also required. To overcome this, a mechanism was added to switch between real and virtual visualization modes.

The design and development of this teleoperation system required considering diverse aspects such as having a user-friendly operation interface, dealing with communication media (such as transmission, protocols, and time-delay problems), creating a custom virtual environment faithful to the real environment, selecting proper virtual reality devices, implementing emerging features, and testing with human operators.

The system should allow everyday users to perform navigation tasks without extensive training. Therefore, the robot should be controlled remotely through an operation interface, and the operator should be able to move or navigate the robot with freedom in the remote environment. In addition, the

operator should be able to view the remote environment through the robot's stereoscopic cameras with a 3D effect on the HMD.

To deal with latency issues and prevent the operator from having an unpleasant user experience, it is possible to change the visualization mode from real 3D to virtual 3D and continue to operate the robot in the VR environment with no difference from the real one. One of the goals of this dual visualization is that the user was able to navigate successfully in the remote environment by real or virtual mode indistinctly without further issues such as dizziness, collisions with objects, or damaging any component of the robot or the remote environment.

Another crucial point to consider was the lighting conditions. The robot had to be able to operate normally even when changing the lighting conditions. This means that if the remote environment goes dark, the real visualization would be inappropriate to continue operating as the operator would be unable to see the environment. When changing to the VR mode, this change in lighting shouldn't affect the visualization. However, the operator must still be able to perceive the new lighting conditions to preserve the feeling of presence and enhance the user experience.

The developed system contains an operation interface with two visualization modes: a real stereo video from the robot's cameras and a virtual 3D environment mapping the real environment. The interface is displayed in an HMD to put the operator in a virtual space that contains both visualization modes. This HMD also controls the movements of the robot's head by following the operator's head movements. For navigation and other features, a joystick is used. The service robot can perform navigation and simple tasks autonomously or it can switch to teleoperated mode when navigating in difficult places or when performing complex tasks.

The tests suggest that the immersive mixed reality interface for teleoperation provided improved usability for operators. On the one hand, the VR mode provided an environment to experiment naturally with the robot and remote location. On the other hand, visualization in real 3D was a great aid when users needed to ensure that the task was performed similarly in real and virtual environments. Communication latency appears to have improved when using virtual mode, as there is no need to visualize real video. Additionally, the virtual mode provided a certain level of realism, which is a key feature for training operators. Teleoperation was presumably successful in both environments without collisions or major problems. Switching between visualization modes appears to provide more control to the operator and generally improves the experience.

2. Related Work. Mixed reality techniques are a convenient instrument for teleoperation systems. In recent years the advances in VR technology have prompted the use of HMD's and 3D visualization. A variety of applications taking advantage of such technologies have been proposed to enhance teleoperation. For instance, Lipton and collaborators [22] developed Baxter's Homunculus, a telerobotic system that uses commercial VR technology and a Baxter robot. Baxter is a versatile manufacturing robot designed to perform repetitive assembly line tasks. This system consists of a VR Control Room presented to the user through an Oculus Rift (a commercial HMD). The Control Room has the shape of a robot head with a window showing the robot environment pretending the user was a minuscule human inside the robot controlling it. Unlike the proposed system, Baxter's Homunculus does not display the images from the robot cameras directly to the user's eyes but on a pair of 2D surfaces representing virtual windows within the Control Room which may cause a loss of situational awareness. Lipton and collaborators dealt with latency by adding the Control Room to hide the delay issues and leave the 3D interpretation task to the user's brain. The Homunculus system requires three computers to operate, one for the Oculus, one for the robot, and one for transmitting and receiving a video stream from the HD cameras. The proposed system requires only one non-specialized computer to control the robot and a mobile device or the Oculus Rift (with an additional computer) to visualize the user interface. Our system uses a joystick to operate the robot intuitively as it was inspired by video games, in contrast with the Homunculus, which requires the users to complete a tutorial before using the system.

Immersive teleoperation interfaces present a system that maps movements from the user to the teleoperated robot to perform them and visualize what the robot perceives through its cameras. Teleyes [23] is a system based on stereoscopic vision and head movement tracking to control Unmanned Vehicle Systems (UVS). Its goal is to reduce the visual distortion in remote environments by synchronizing human and machine vision. Teleyes provides the operator with the sensation of being on board the UVS by displaying the images from two cameras attached to the UVS on an HMD. The use of stereo cameras provides a better perception of the environment depth. The system synchronizes user head movement with the controlled UVS. Teleyes uses a VR Headset with an iPhone 6 plus as HMD. An inertia measurement unit (IMU) sensor was attached to the HMD for head tracking, and two cameras were installed on the UVS to capture the stereo video to transmit it to the user. Teleyes and the proposed system share the approach of enhancing the teleoperation experience by matching the stereo cameras and head movement with user visualization and control of the device. However, Teleyes was designed to work

in a local wi-fi network which reduces communication issues. Teleoperation from a distance or over the Internet is not under the scope of Teleyes. Its experiments aimed to prove the ability to interpret the remote environment from the sense of sight. Teleyes only provides real-video visualization of the remote location it does not use any virtual environment.

Virtual Environment for Tele-Operation (VETO) [24] is an immersive user interface for the teleoperation of a small robot. VETO was designed to operate in a fully observable environment scanned beforehand and automatically generates walls and rooms in a virtual environment for the user to explore them. It improves the raw visual feedback with additional information in the form of text and haptic feedback. The CrossSock Networking API was developed to work with VETO using the C++ programming language. CrossSock is a cross-platform, lightweight, and header-only high-level solution for developing client-server architectures. It uses Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) sockets for data transmission. VETO uses a Leap Motion Controller mounted on the HMD to generate a mesh representation of the user's hand position and finger movements in the virtual view. Its architecture uses the Unreal Engine 4 game engine for its user interface presented on an Oculus Rift VR headset. Similar to the Homunculus project, the VETO interface does not show video from the robot stereoscopic cameras directly to the user's eyes. Its perspective view is from a standing human facing the robot. VETO is an entirely virtual environment where all the displayed rooms, objects, and the robot itself are virtual representations of the real ones. Only when there is a need to identify an obstacle, images from the robot cameras are displayed on a pair of side-by-side windows-like 2D surfaces. This feature makes the need for video transmission occasional, and even when transmitting, the latency issues are hidden by the window-like visualization of the video. VETO aims to be a framework for robot platforms with immersive VR game interfaces. In their paper they present a user study that shows the benefits of using immersive virtual environments for teleoperation in comparison with traditional teleoperation methods.

In summary, a number of the existing VR teleoperation systems present real video visualization from the robot's cameras. Some have chosen to use VR environments to get better control of robots. A few have used mixed reality techniques and have real and virtual visualization working together. Nevertheless, none of them have managed to use a combination of real and virtual visualization to the point of using them interchangeably. The proposed system has an immersive dual visualization (real and virtual) which brings the best of both approaches. Each teleoperation system exists for a specific purpose; ours was to work with humanoid service robots. The visualization

mode can be switched from real to virtual or vice versa to suit user needs. The importance of visualizing the real environment is to make timely control decisions when performing daily life house tasks or interacting with humans. In addition, operating under poor lighting conditions is challenging for most of the existing teleoperation systems. It is tremendously difficult to visualize real video in dark surroundings. Our system overcomes this situation by using the virtual mode and the robot's sensors to continue operating. The lighting conditions are imitated in the virtual environment but preserving the ability to observe all scene elements; therefore, the feeling of presence is unaltered.

Lately, Internet-based robotic teleoperation has become popular. However, Internet's slow rate and connection instability restrict real-time control and feedback. The user may not be able to observe the remote environment real-time changes through video images due to network delays. Although latency is rarely discussed, it is a extremely important issue to overcome. One of this work's goals is to reduce latency to improve user experience. VR visualization use considerably diminishes latency issues since only small data packages containing robot information and commands need to be sent. This approach reduces system traffic compared to transmitting video images and allows proper robot control even when communication rates are slow. By adding a 3D VR environment to simulate the real environment, the proposed system provides the user with a "live" virtual representation of the remote environment instead of the delayed 3D video, which may also increase the efficiency of the user performance.

The discussed works suggest there is a growing trend towards immersive teleoperation interfaces enhanced with mixed reality techniques. Mixed reality techniques and versatile robot control techniques create a symbiosis that can break new ground in the fields of virtual reality and robotics.

3. System Design and Development

3.1. System overview. The aim of this work was the creation of a teleoperated service robot system with an immersive mixed reality operation interface. Besides, it was intended that such an operation interface would be easy to use, capable of work on different platforms, and consume as little computer resources as possible.

The proposed system (Figure 1) consists of two parts. For remote device, a service robot with a laptop attached to its back was used (Figure 1 LEFT). This laptop's purpose is to directly control the robot and needs an internet connection to communicate with the operator side. For the operator to control the robot, an immersive operation interface displayed on an HMD was created. Two hardware settings for the operation interface are possible. Setting A (Fig. 1 RIGHT TOP) is an HMD wired connected to a PC and a wired joystick.

Setting B (Figure 1 RIGTH BOTTOM) is an HMD that uses a smartphone as a display and a Bluetooth joystick. In both settings, a stable internet connection is needed.

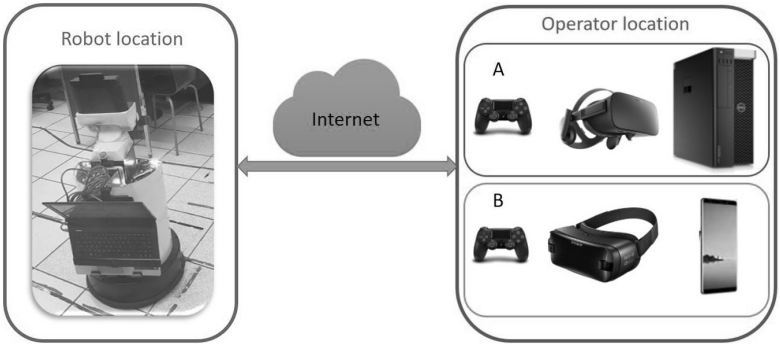

Fig. 1. General system overview. Robot and Operator locations are communicated via regular internet connection. LEFT, Real Service Robot with portable computer device mounted on its back. RIGHT, Operator hardware settings for the immersive experience with two possible configurations (A and B)

To deal with video transmission delay, a mix of visualization modes is proposed: real video mode and virtual environment mode. By having a virtual environment, the resources needed to communicate with the robot can be reduced. Besides, when in need of visualizing the real environment, like on regular teleoperated systems, visualization can be switched from virtual to real or vice versa. This duality allows having the best of both approaches.

The real environment was pre-mapped into the virtual environment, this means the real and virtual environment perfectly match. On the one hand, when using the virtual mode, only real-world robot location has to be transmitted. This reduces the amount of information to be sent and gains efficiency, faster control, and provides a better teleoperation experience. On the other hand, the real video mode can be activated if there is a need to display the robot's view. This mode allows the operator to visualize the real environment through the robot's cameras and determine if the robot is properly performing.

3.1.1. Hardware. The service robot used on this work was the Toyota's Human Support Robot (HSR) [25]. HSR is a service robot that has two main purposes: physical work and communication. Although this robot is currently only available for research, it has been used in many international robot competitions such as RoboCup [26] and it is planned to be commercially available soon. Combining 3 degrees of freedom (DOF) of its base, 4 DOF of its arm, and 1 DOF of its torso, the HSR features 8 DOF, allowing flexible

movements for navigation and object manipulation. The HSR also has features like sensors, laser, microphone, stereo and depth cameras, etc. (Figure 2).

Fig. 2. Toyota's Human Support Robot (HSR)

Head-Mounted Displays (HMD) were used to provide immerse sensation. HMDs consist of a screen divided in two to show a different image to each eye of the user with a pair of glasses. The field of view is widened so that the image appears positioned several meters ahead [27]. These virtual reality headsets allow to see a virtual world directly on their screens and allow an immersive sensation in the virtual environment.

In order to test the system performance on different platforms, two different HMDs were used: Setting A, an Oculus Rift connected to a PC (Figure 3 LEFT) and Setting B, a Samsung GearVR with a Samsung Galaxy Note 8 smartphone (Figure 3 RIGHT).

Fig. 3. Hardware Settings. LEFT, Setting A: PC and Oculus Rift. RIGHT, Setting B: Galaxy Note 8 and Gear VR

Oculus Rift [28] is a virtual reality headset developed and manufactured by Oculus VR. Its technical features make it ideal for immersiveness in this system. Samsung GearVR [29] is also a virtual reality headset created by Samsung Electronics in collaboration with Oculus VR and it is compatible with Samsung mobile devices. The screen of these devices is used to display the images, and their accelerometers and gyroscopes are used for rotation tracking. The ability of this headset to work with mobile devices allows them to be easily used in a domestic environment.

The PC used in Setting A has the following features:

- PC Dell Precision Tower 7810
- OS Window 10 64 bit
- Processor Intel(R) Xeon(R) CPU E5-2650 v4 2.2GHz
- RAM 16 GB
- GPU Quadro P400
- GPU Memory 8GB GDDR5
- CUDA Cores 1792

For controlling and interacting with the virtual environment a joystick was used, the Dualshock 4 [30]. This joystick is the fourth iteration in a line of gamepads developed by Sony Interactive Entertainment for the PlayStation family. It has various buttons for command input, as well as the use of Bluetooth and USB for connectivity with devices other than PlayStation. This feature allows it to be used with PCs and mobile devices with Android.

As it was shown, to make the system accessible to a broader audience, commercial not specialized hardware was used.

3.1.2. Software. This system uses ROS (Robot Operating System) for robot main control, and sockets to connect the interface with the robot. ROS [31] is a robotics middleware, a set of software libraries and tools to build robotic applications. The Kinetic version of ROS running on the Ubuntu Operating System (Linux) 16.04 LTS was used.

For autonomous navigation, complex tasks, and robot behaviors, a version of ViRbot (VIrtual and Real roBOt sysTem) adapted to Toyota's HSR was used. ViRbot [32] is a hybrid robotics architecture to operate autonomous robots. ViRbot system inter-operates with ROS to form an intelligent systems development platform for computer vision, digital signal processing, automatic planning, automatic control, and human-robot interaction. ViRbot was adapted to work with the HSR proprietary libraries and this adapted version has been tested in our HSR at international robot competitions such as RoboCup where it was awarded 2nd place in 2018.

Video game engines have been widely used for robot simulation due to their ease of content creation that allows focusing on more specific topics in

the field of simulation. Game engines include elements for the simulation of physics, lighting, artificial intelligence, rendering, network, etc. The quality that can be achieved with these engines allows creating a simulation convincing enough to be used on the interface virtual mode.

Unity 3D [33] is a video game engine that allows the creation of games for various systems and devices such as PC, Xbox Family, Play Station Family, Nintendo consoles, Android, iOS, Web, etc. Furthermore, Unity 3D uses C# and javascript as base languages to create scripts that model behaviors. As for 3D models, it is possible to use various formats such as .obj, .fbx, and .mb. All these features make Unity ideal for this work. The Unity 3D version used was 2018.2.21f1, running on Windows 10 Professional Operating System.

3.2. Local interface for teleoperation. Local interface refers to the operator control interface presented on the HMD. Communication between the local interface and the robot must be precise and fast. Information such as robot position, video from cameras, and control commands need to be transmitted from and to the robot.

To generate an immersive 3D environment, stereo video from the robot's stereoscopic cameras was transmitted to the local interface and displayed on the HMD. On the interface, the operator can visualize a 3D real environment (video mode) or switch to a 3D virtual environment (virtual mode). Additionally, the operator's head movement on the HMD was mapped to the robot's head movement, so that he can freely explore the remote environment by moving his head (Figure 4).

Fig. 4. Mapped head movements. When the user tilts his head up, the robot also tilts its head up

The head movement information was sent to ROS using update messages (detailed on section 3.3) to mimic this movement on the robot. Upon receiving

the update messages, they were published as ROS messages to be executed by the robot. Moreover, precision in the detection of headset movement had to be carefully considered. If not properly managed, even the slightest operator head movement could trigger a robot head movement and this could cause him dizziness. To address this, a threshold for motion detection was implemented. This means, when the movement was under the threshold, the information was not sent to avoid this unpleasant experience.

To control the remaining robot movements, the joystick's buttons and stick were mapped on Unity. By this mapping, Unity gathers information about the direction and speed of the joystick's movements. This information was sent to ROS to be converted into movement commands for the robot. The stick movements were sent as state update messages and the button information as event or command messages (detailed on section 3.3).

For the immersive 3D experience, the images from the robot's stereo cameras were displayed on the headset lenses corresponding to each eye of the user. To achieve this, a connection with the robot was established to receive the images on Unity. The robot can capture video with its frontal and stereo cameras but in image format. This means, the robot cannot capture video, instead, it captures static images. These images are quite large as stereo vision requires images from both stereo cameras (left and right). Thus, a large amount of data was required to be transmitted over the network.

A ROS library, web video server was used to this end. This library allows ROS to convert the robot video images into a stream of compressed mjpeg images. This stream is a continuous sequence of jpeg format images with no headers or footers. This stream was received asynchronously in Unity so that it would not cause slow processing nor affect the images displayed. The images received were assigned to textures and presented to each eye of the user to recreate the video from the real environment with a three-dimensional effect on the HMD.

To deal with data quality of the received video, image enhancement techniques were applied. Those techniques are commonly used to improve the visual appearance of images, in terms of sharpness, distortion, contrast, etc. Enhancement techniques in the spatial domain use the information from the image pixels to perform the enhancement. One of these techiniques is the Laplacioan Filter [34], which is a high-pass filter that highlights regions with rapid intensity changes and is used for edge detection. The Laplacian $L(x,y)$ of an image with pixel intensity values $I(x,y)$ is given by Eq.1. This can be calculated using a convolution filter. Given that the received images are discrete, then a discrete convolution kernel was used. By using this filter, the

edges on the image can be obtained and must be added to the original image to highlight the edges and improve the sharpness.

$$L(x,y) = \frac{\delta^2 I}{\delta x^2} + \frac{\delta^2 I}{\delta y^2}. \quad (1)$$

The images received as a stream of jpeg images were placed in Texture2D Unity objects and from them, a color matrix was obtained. A convolution was applied to perform filtering and enhancement of the picture. However, image processing can become a time-consuming task, especially when the images to be processed are large. Visualization of real or virtual environments could be considerably affected by this processing and could reduce the user experience fluidity. In such cases, image processing should be performed in parallel to avoid interference with the user experience. This process was made in Unity with an asynchronous task for image processing.

After finishing the image filtering, the main thread detects it and the new texture with the improved image is applied to the objects that show the video on the headset. Then, the filtering process begins anew with the latest received image. This process may cause frame loss, but the latest image is always preserved.

3.3. Robot-interface communication. Communications on virtual environments are categorized by Kessler [35] on three main types:

– **Events messages:** Transmit information that cannot be discarded.

– **Command messages:** Similar to event message but require a response.

– **State update messages:** Any transmission about the current state of the shared environment. A state message becomes stale when a new state message is generated for the same set unless an event or command has occurred since the last state update or the full history of state changes is required by the receiver.

Distributed virtual environments usually transmit a large amount of state update messages. A task can not allow losing time receiving old messages. In this work, even though the virtual environment is only on one side of the communication, the information from the real environment was abstracted by the robot. This allowed the information to be handled as if both environments were virtual and thus improve and simplify communication between the virtual interface and the robot.

Sockets were used for robot-interface communication because they provide a simple and reliable way for communication between completely different processes. For instance, they can be used in different systems programmed in

different programming languages running on different OS. This means sockets are suitable to communicate ROS with Unity 3D.

Sockets for trusted and untrusted communication can be implemented on ROS and Unity via C++ or C# code. Trusted communication refers to messages that always arrive at their destination and are implemented by TCP sockets. Trusted communication could be slow to ensure the message's arrival and in case of loss, they are asked to resend the message. This feature makes them ideal for command messages. Untrusted communication refers to messages that may not arrive at their destination and are implemented by UDP sockets. This communication is faster and even with the possible data loss, they are ideal for state update messages.

The laptop attached to the robot (Figure 5 LEFT) contains the ROS nodes and it is connected to the robot via Ethernet cable and to the Internet wirelessly to connect with the interface. Incoming communications on the corresponding ports must be redirected to this computer to ensure proper communication. The device holding the interface (Figure 5 RIGHT) can be connected to the Internet via ethernet, Wi-Fi network, or mobile network (in the case of Samsung Gear VR). Communication must be established from the device running the Unity interface to the computer running ROS. The ROS computer receives the data and converts it to information to be published in ROS for execution by the robot. When it receives information from the nodes, it sends it through the sockets to the connected Unity 3D device.

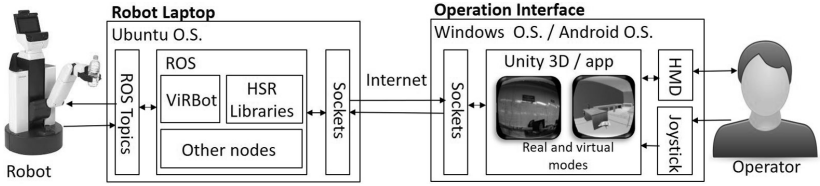

Fig. 5. Immersive Mixed Reality Teleoperation System. The robot laptop controls the robot and gathers the required information from the robot sensors and cameras. It contains all the ROS nodes running on a Ubuntu O.S. and communicates with the operation interface using sockets. The operation interface was created with Unity 3D. It can be executed from Windows for the Oculus (setting A), or from the generated Android app for the Samsung Gear VR (setting B)

3.4. Virtual environment. The operator had to be able to control or explore the environment interactively and fluent. This means, the virtual environment had to be able to emulate processes such as real environment and objects physics, graphics display, objects interaction, and artificial intelligence when required.

The Oculus Rift API for Unity 3D, allows to easily integrate the Oculus headset motion recognition, and the accelerometers and gyroscopes of the smartphone inside the Samsung Gear VR. Unity allowed creating the virtual environment for both platforms, Windows for Oculus, and Android for Samsung Gear VR. In the case of Samsung Gear VR, it was required to install the JDK (Java Development Kit) and to have Android SDK (Standard Development Kit) to compile.

The virtual environment was designed and developed to perform the teleoperation of the HSR. To this end, the real HSR was replicated on the virtual environment by an HSR Unity model (Figure 6) from SIGVerse simulator [36].

Fig. 6. Virtual HSR model

SIGVerse is a tridimensional Unity-based simulator created by Dr. Inamura and collaborators [37], which allows 3D environment interactions (perception and action) and was designed to simulate HSR in a domestic environment.

To abstract the real space into the virtual space, adequate corresponding scales were required. Thus, the robot model was used for scale reference to match real and virtual environments. This abstraction was achieved using the robot to map the space where it was going to navigate. Using the robot's laser system with ViRbot modules and its adaptations to HSR, a map of the real environment was obtained by navigating the robot in the real space (Figure 7).

Fig. 7. Biorobotics Laboratory map obtained from HSR using ViRbot

This map image was transferred to Unity 3D in an adequate position and scale (Figure 8 LEFT). With this image as the base floor, fixed objects such as walls, ceiling, floor, or large size furniture were represented on the virtual environment by 3D models on a fixed position corresponding with their real position (Figure 8 RIGHT). When adding a large number of complex structures to the virtual environment, the interface could become slow, thus, it was preferred to use simple 3D models to avoid overload.

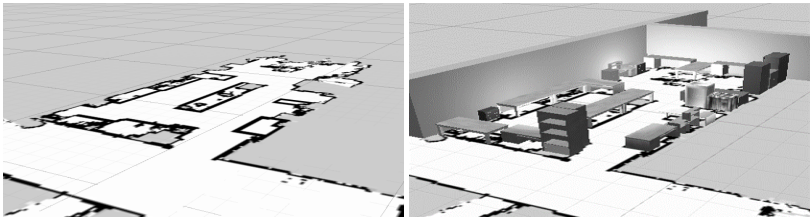

Fig. 8. Map from real space to virtual space

To locate the robot in the virtual space, the real robot used its sensors and ROS libraries to locate itself and transmit its position to the interface to replicate it in the virtual environment. The robot position relative to the map is requested continuously. Once the position was obtained, it was sent to the virtual environment through sockets. Status update messages were used since this information is constantly changing and only the latest available is used. In the virtual environment, the information was received and assigned to the robot model to position it on the corresponding location.

3.4.1. Lighting conditions. Most of the existing teleoperation systems face problems when operation under poor lighting conditions (e.g. with lights OFF) as real video visualization becomes hard to achieve. To address this, on the local interface the operator can switch to virtual mode to be able to properly visualize the environment and avoid collisions when moving. Nevertheless, when using teleoperation in virtual environments, the virtual environment must be a close representation of the real environment. This means a change in lighting conditions should be perceived by the operator.

Unity allows creating ambient lights with brightness levels within the virtual environment. These levels can be adjusted in real-time from the brightness information of the real environment. Since the images received from the robot are RGB images, they can be converted to HSV (Hue Saturation Value) space. The HSV color space is an alternative representation of the RGB model commonly used for rendering images. The Hue of this space represents the color, the Saturation represents the intensity of the color, and the Value

represents the brightness of the color. The average from each frame pixel's Value or luminosity was used to set the Unity brightness levels.

3.5. Implementation overview. As shown in Figure 5, the implementation of the system is divided into two main parts, Unity and ROS.

In Unity there is a Scene containing all the elements of the local interface. This Scene is presented in the HMD. A virtual camera mapped with the HMD is placed at an initial position inside the Scene (Figure 9).The camera is facing the virtual environment to visualize all the virtual elements on the scene. A couple of 2D planes are placed in the corresponding position with the HMD for each eye. Those planes will display the real video to the user. The images from the stereo cameras are used as a texture for these planes, which provides the 3D effect. When selecting the real visualization mode, the planes are enabled, and the video is displayed on the 2D planes. When switching to virtual mode, the planes are disabled, and the virtual environment can be observed.

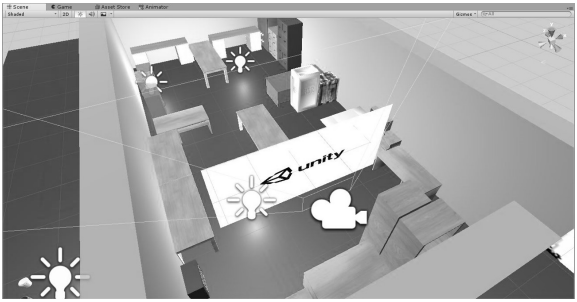

Fig. 9. Unity Scene. A virtual camera representing the user is placed at an initial position

To control the robot, the joystick is monitored by Unity to detect stick movement or button pressing and convert this to messages to be sent through sockets. A similar process applies to the head movement but monitoring the HMD movement from Unity. The generated messages are sent to ROS to be processed and executed, and in return, new information from the robot is received in Unity to update the virtual camera position and rotation accordingly to the real robot's base and head positions.

In ROS, the messages arrive through sockets then, they are converted to commands and sent to the robot for its execution. In return, the robot's updated information is converted to messages and returned to Unity via sockets.

For the video transmission, the web video server ROS package was used. In summary, web video server opens a local port and waits for incoming HTTP requests. When a video stream of a ROS image topic is requested via

HTTP, it subscribes to the corresponding topic and creates an instance of the video encoder. The encoded raw video packets are served to the client. Only a URL with some parameters is required to connect to the node [38].

A brief description of the Unity C# classes and their main methods, and ROS nodes and their functions is given below:

Player class: Main class, associated with the user. Controls all the functions of the local interface. Initializes and configures data for communications, robot control, visualization, etcetera. Contains HostAddress, HostPortUDP, and HostPortTCP for sockets connections, an array of objects of MPGJStream class associated with the planes that display the video, and other helpful elements. Its most relevant methods are: *Start()* - Establishes connections through ControllerUDP and ControllerTCP and initializes everything. *Update()* - Is continuously executed to update the scene and manages sets the robot position to the corresponding GameObject accordingly to robot state data. *moveVirtualHSR()* - Updates the robot state data (position and rotation) with the information received to be used by the Update method. *headReceived()* - To confirm the reception of head movement message from the real robot. *leftStick()* and *rightStick()* - To manage the joystick's stick movements and send them through ControllerUDP objects. *headRotation()* - Detects head movement from the HMD device and converts it to messages to be sent via UDP sockets. *buttonsDown()* - Manages joystick's buttons pressing and can send TCP or UDP messages configured for each button. Simple commands like close grip or complex commands like go to the kitchen can be assigned to buttons.

ControllerUDP class: Manages UDP sockets. Its most relevant methods are: *StartClient()* - Establishes UDP socket connection, receives messages to move robot base and head, and notifies Player object when receiving messages to execute *moveVirtualHSR* and *headReceived*. *sendMSG()* - Send message through the established socket.

ControllerTCP class: Manages TCP sockets. Its most relevant methods are: *StartClient()* - Establishes TCP/IP socket connection. *sendMSG()* - Send messages through the established socket. *TaskAsyncClient()* and *clientCall()* - Sends messages and receives a response asynchronously when required.

MJPGStream class: Manages everything related to video and its display on the planes corresponding to each eye on the HMD. Its most relevant methods are: *Start()* - It connects to the web video server URL to get the images from the robot stereo cameras. Two objects of this class are needed, one for each eye, therefore two URLs with different topics are used. It also initializes a CustomWebRequest object to manage the received images. *Update()* - Validates if the images received have finished being processed by the filters and

then applies the images as textures for the planes to visualize on the HMD. *changeTexture()* - when the CustomWebRequest object receives a complete image this method is executed. It calls the *filter* method from this class to enhance the image and the *changeLightValue* method from LightManager class to adjust the environmental lights. *filter()* - Applies the filters to enhance the images in an asynchronous process. In this method, different filters can be configured and use the one that suits best.

CustomWebRequest class: Manages the images received from the video web server. Its most relevant method is: *ReceiveData()* - This method executes when data is received. It validates if the information is new or different from the last received. When the complete image is received, it is sent to the *changeTexture* method of the *MJPGStream* class to process it.

LightManager class: Processes the received images to replicate the lighting in the virtual environment. Its most relevant methods are: *changeLightValue()* - Converts all image elements from RGB to HSV to obtain an average Value and assign this to the virtual lights. *resetLight()* - Resets virtual lights to its initial state.

Server node: Main ROS node. It contains all the functions needed to manage messages through sockets, convert messages to commands and vice versa, and send commands to robot. Its functions grouped by usability are described next.

Initialization functions: To configure ROS topics to subscribe or publish, initialize elements such as threads for TCP and UDP sockets and other utilities.

ServerTCP functions: To create TCP sockets, establish communication, receive messages, and send them o the CommandsConverter functions.

ServerUDP functions: Same as ServerTCP but for UDP sockets. Additionally, they can send messages when the head moves.

CommandsConverter functions: To convert received messages into commands and send them to the CommandsToRobot functions for execution and convert robot information to messages to be sent via sockets back to Unity.

CommandsToRobot functions: To send commands to robot for execution. When the command is simple, like move base or head, a ROS message is published to the corresponding ROS topic for execution by the robot. When the command is complex, like “go to the kitchen”, the command is sent to ViRbot (HSR version) to be processed, planned, and executed autonomously by the robot.

4. Experiments and Results

4.1. Local interface tests. To test the performance of the system, a series of experiments were conducted. First, the experimental environment had to be settled. Our University's BioRobotics Laboratory was selected to act as the real environment space. The laboratory was mapped into the virtual space using the procedure described in the previous section. Furthermore, hardware settings A and B for the interface and the robot were properly set, the required connections were successfully established, and the software was loaded to begin the tests. A compact set of users not experts in the field of VR and teleoperation tested the system. All users received identical instructions as described below.

The interface tests were divided into three types: real mode only, virtual mode only, and mode mixing freely.In all tests, users had to wear the HMD and grab the joystick. Subsequently, the interface was started and connected to the robot to begin the visualization. In real mode tests (Figure 10), the video received from the robot's stereo cameras was displayed on the HMD with a 3D effect. Such visualization provided the users with the immersive sensation of being inside the robot; in other words, they were "seeing what the robot was seeing". The users were requested to move their heads to explore the remote environment. Regular movements were mimicked by the robot as expected, and the users could control the robot's head with their heads. Next, the users were instructed to make slight random movements to test the movement threshold. To prevent these movements from causing unpleasant dizziness, movements below the threshold did not affect the robot's movement.

After exploration by head movement, the users were asked to navigate the robot using the joystick for about 5 to 10 minutes. Once the users got used to controlling the robot with the joystick, they were able to navigate the robot through real space and visualize the environment according to the robot's new position. Navigation tests were fluently and without significant incidents. However, navigation was pretty slow and clumsy. Users had to reduce the speed of the controlling movements to deal with video delays. They explained that when performing fast movements with their head or fast-moving the robot with the joystick, the video had a delay in showing the movement which made them feel dizzy. In addition, they reported having used the move and wait strategy to avoid collisions or getting lost due to the outdated video. The aforementioned video latency incidents turned the user experience less pleasant.

To ameliorate the previous experience, the users were instructed to change to virtual mode and repeat the previous tests in the virtual environment (Figure 11). Users explored and navigated using only virtual mode for about 5 to 10 minutes. In these new tests, the users reported no delay in visualization

Fig. 10. User Experience Real Mode Tests. Visualization, head movement exploration, and robot navigation with the joystick

and smoother navigation and head movement. Users no longer needed to use the move and wait strategy, making their performance visibly faster and more confident.

Finally, the users were allowed to switch modes freely and repeat the explore and navigate tasks for 5 to 10 more minutes. Users preferred to use the virtual model for navigation. Nevertheless, they inclined toward real-mode to explore the details of the environment and verify the robot was performing the same as in the virtual environment. According to users, the teleoperation using this mixed mode felt more comfortable, avoided dizziness, and improved their performance.

User experience could become difficult to measure as it is somewhat subjective. Nevertheless, there are some tools designed to aid in this task. One of the most practical and reliable tools is the System Usability Scale (SUS) [39]. SUS is an inexpensive yet effective tool for assessing the usability of a system. The SUS is composed of ten statements, each having a five-point scale that ranges from Strongly Disagree to Strongly Agree. Its final result is an intuitive 100 points scale. Several studies have shown that SUS results are reliable and accurate. Therefore, SUS has become the most widely used measure of perceived usability [40] and is one of the most used methods in User Experience.

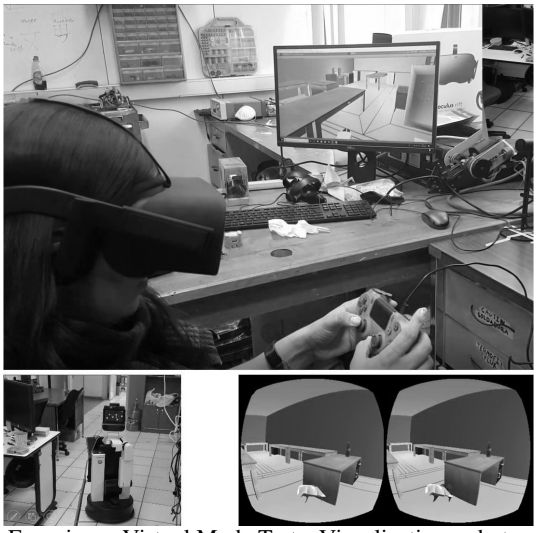

Fig. 11. User Experience Virtual Mode Tests. Visualization, robot navigation, and head movement tests in virtual mode

A SUS questionnaire was applied to users after using the system to evaluate system usability. The answers were processed as indicated by SUS methodology, and an average score was obtained. The usability SUS score of the system was 80.83 ± 3.81. An "Additional comments" section was included in the questionnaire. In there, users could freely express their impressions of the system. The summary of the most popular comments is shown next.

Comments for Real Mode:

- Liked the ability to see the robot's cameras images in 3D.
- Dizziness caused by the delay between movements and visualization.
- Easy to navigate with the joystick controller.
- Frustration when attempting to achieve a task due to delays.
- Liked to visualize the real environment to confirm the robot position.

Comments for Virtual Mode:

- Agreeable transition from real to virtual and vice versa.
- Accurate mapping between real and virtual environments.
- Dizziness by delay completely disappeared.
- Smooth movement of the camera accordingly to head's movement.
- Would like the virtual environment to be a little more realistic.
- Compared the system with a VR video game.

About 80% of the users referred their experience with the system when only using the real mode was quite fascinating but not very comfortable, however when switching to a virtual model, their experience improved, and the uncomfortable sensations diminished or disappeared. 70% explained that using virtual mode only was preferable, but they would have been uncertain about duly completing the desired task. 52% pointed that virtual mode felt like a VR game which could make them forget about the real robot and task. 95% expressed that having the possibility to switch between modes increased their confidence in operating the robot and provided them a feeling of integration between the VR video game-like experience and the visualization of the real 3D cameras.

4.2. Latency and frame rate tests. Video delay issues on real mode clearly affected user performance and led to an unpleasant user experience. As users depend on video for perception, direct interfaces typically demand low-delay communications. The virtual mode was devised to reduce latency and cope with these circumstances. To quantify the improvements of using virtual mode over real mode, the following tests were carried out.

Latency and frame rate are the critical points to measure as the user experience entirely depends on them. Distance is another vital point when transmitting a large amount of data. Therefore, we decided to perform the tests using a local network and virtual private networks (VPN) at different distances. VPNs at Tampa (United States), Singapore, and South Africa were used. The data traveled from our Laboratory through the VPN and back to the computer that controls the robot. Given that network traffic varies during the day, the tests were carried out on different days and daytimes.

A stopwatch was started when sending a stream of images (mjpeg) and stopped when receiving the beginning of video measure the video mode latency. In virtual mode, no video data is sent, therefore, to measure virtual mode latency, the response time of robot base movement and robot head movement was recorded. Another stopwatch was implemented on Unity, and it was started when sending a command to the robot. When receiving the command in ROS, a return signal was sent to indicate the correct data was received. In the case of the robot base, the total response time between sending the command and updating the robot position was recorded. In the case of the head, response time only between sending the commands and receiving them to execute in the robot was considered. The same VPNs and conditions for 20 tests were used. The average response time of 20 tests is shown in Table 1.

To better understand the latency results, the response ratios for Video/Base and Video/Head are shown in Table 2. Those ratios are the results of comparing real and virtual modes latencies. They show how much faster

Table 1. Latency Tests. Average response time of 20 tests

Network type	Video Stream (ms)	Robot Base (ms)	Robot Head (ms)
Local Network	844.00	251.650	86.15
VPN Tampa	1816.00	332.65	132.05
VPN Singapore	1624.00	708.30	544.70
VPN South Africa	5187.00	694.50	596.65

the virtual mode is relative to the real mode. For instance, in the local network, the base response time is 3.35 times faster than video, and head response is 9.79 times faster than video.

These results denote the difference between using real and virtual modes. It can be inferred that the improvement is due to the short amount of data sent in virtual mode compared with the video data needed in real mode. In virtual mode, only commands for the movement of the base and head and the location of the robot are required to be transmitted. In real mode, high frame rate video from two cameras is required, which is a large amount of data. Therefore, the virtual mode is faster and more fluid.

Table 2. Latency Tests. Average response ratio of 20 tests

Network type	Video/Base	Video/Head
Local Network	3.35	9.79
VPN Tampa	5.46	13.75
VPN Singapore	2.29	2.98
VPN South Africa	7.47	8.69

A key feature of VR applications is displaying content at a high enough frame rate to provide a compelling experience. Frame rate refers to the frequency at which consecutive images appear on a screen. Human vision can process individual images up to 10 to 12 images per second [41], but images at higher rates are perceived as motion. That means the higher the frame rate is, the smoother the motion will be perceived.

Video mode frame rate was obtained by counting the received images and dividing this number by time. Every time a complete image was received it was accounted into a counter in Unity and then it was divided by time. For virtual mode frame rate, a similar procedure was used. In this case, when the scene was updated, it was added to the counter. The average frame rate obtained is shown in Table 3.

Table 3. Average frame rate of video and virtual modes (frames per second)

Network type	Video	Virtual
Local Network	35.08	316
VPN Tampa	35.79	316
VPN Singapore	34.56	316
VPN South Africa	22.92	316

The received images were processed using image enhancement as explained in section 3.2 before displaying them on the HMD to avoid possible data quality loss on video transmission. Additionally, as images are constantly arriving to generate video, some data losses do not seriously affect the system performance or the user experience. If a frame is lost or damaged, the next frame arriving replaces the missing information, so users do not perceive it is missing. The video quality was subjectively well evaluated by users; only delays caused by latency affected video user experience. Regarding command and robot data, they were sent using trusted communication and implemented by TCP sockets. As explained in section 3.3, this guarantees the delivery of data and packages, then the quality of this data is not compromised. It is worth mentioning that during all the tests, the robot and command data always arrived and were executed successfully.

4.3. Virtual environment tests. Virtual environment tests were carried out to measure the closeness between the robot in the real world and its representation in the virtual environment. In addition to navigation and head movement tests reported on section 4.1, location, and illumination tests were performed.

On location tests, the users navigated the robot on the real environment to locate itself (Figure 12 TOP LEFT). The robot used its sensors and the real environment map to obtain the real-world coordinates of its position; the robot location could be visualized on RViz (a 3D visualization tool for ROS) on the ROS computer (Figure 12 TOP RIGHT). Then, the robot's real-world coordinates were transmitted to the interface where Unity mapped them into its coordinate system to display the virtual robot (Figure 12BOTTOM).

Real and virtual robots' positions were properly matched in their corresponding environments. This allowed the users to navigate in both environments interchangeably and thus be able to use the most convenient visualization for each situation.

To test illumination variations, two lightning settings were used: regular illumination (with lights ON) and poor illumination (with lights OFF). In the

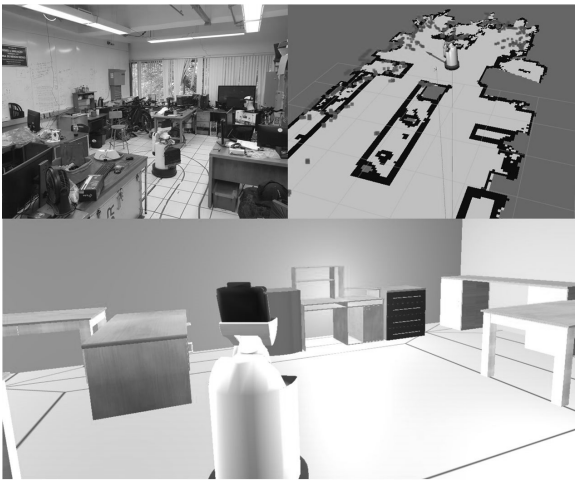

Fig. 12. Localization Test. TOP LEFT, Robot navigating on real space. TOP RIGHT, Robot location found in real-world (RViz). BOTTOM, Robot location represented on virtual space

first case illumination on the virtual environment remained normal (Figure 13 LEFT). In the second case, illumination on the virtual environment was automatically adjusted to darker lighting (Figure 13 RIGHT).

The difference in lighting conditions could be appreciated in the virtual environment but not as drastically as in the real environment which allowed to continue operating the robot even when in the real environment it was not possible.

5. Discussion. The tests suggest that the system was able to successfully display 3D video from the robot cameras on the HMD and provide the operator with a compelling immersive experience. Users intuitively interacted with the interface and the remote environment but reported some difficulties visualizing the real 3D environment. The momentary lags on the video screen caused them to switch to a “move and wait” strategy. Otherwise, they would experience dizziness and other uncomfortable effects. Either way, most of them were satisfied with the “seeing what the robot was seeing” experience. The ability to move the robot’s head with the head was well received, and being able to visualize it in 3D was a great improvement. Even with the "move and wait" strategy, the robot’s visual exploration and navigation were successful.

The real-virtual mode switching mechanism was implemented in a joystick button, which allowed users to switch modes at will. When changing to virtual mode, users reported a significant improvement in the visualization

Fig. 13. Illumination Test. Different lighting conditions in real space and their representation in virtual space. LEFT, Lights ON. RIGHT, Lights OFF

and navigation tasks. On the one hand, users preferred to use virtual mode for navigation and only switched to real mode to confirm they had reached the desired location. On the other hand, visual exploration was preferred in real mode even with the waiting issue. They claimed real mode allowed them to appreciate with detail the real remote location.

Regarding the results from the SUS questionnaire, recent research shows that the magnitude of SUS means closely correspond with means of other questionnaires designed to assess perceived usability. Researchers have found that a SUS score below 68 (average SUS score) indicates issues with the design that need to be improved. The SUS average score for this system (80.83 ± 3.81) was visibly above 68, which denotes that the system is "ACCEPTABLE" considering the SUS acceptability ranges (Figure 14). The average score obtained also suggests an adjective rating between "GOOD" and "EXCELLENT".

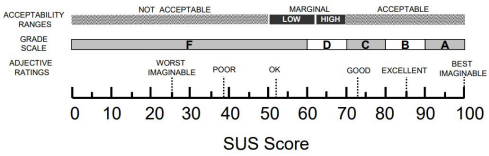

Fig. 14. SUS acceptability [42]

Additional users comments obtained from the questionnaire reveals that the virtual mode seems to have reduced the uncomfortable effects of latency of real-video transmission.

In his work [43], Robert M. Miller claims that a response time of two seconds is a universal requirement for human-computer interaction to be

tolerated and perceived as interactive, and in some cases, up to half a second. Given that one of this system's goals was to provide a convincing immersive user experience, the purpose of having a dual visualization mode was to deal with unpleasant issues caused by latency. On the latency tests (Table 1), it can be observed that video time response in the local network was under 1 second, so it accomplishes the requirement for being perceived as interactive. However, video stream response time considerably increases with distance, up to more than 5 seconds in the farthest location. In contrast, the response time for the robot base and head is significantly lower, and even at the farthest distance, it never exceeds 1 second.

To quantify the improvement in latency of virtual mode over video mode, the Video/Base and Video/Head response ratios are presented in Table 2. On this table, it can be observed that robot base response is up to 7.47 times faster than video stream, and head response up to 13.75 times faster. All these results coincide with the fact that when transmitting real video from the robot's cameras, a large amount of information is sent over the network. On the contrary, only basic information about the robot and the environment is transmitted when visualizing the virtual environment. This denotes that the virtual model relies primarily on the graphics displayed on the local device, not on the streamed video data, which vanishes latency's disgusting effects and improves the user experience.

From table 3, it can be observed that the frame rate was around 35 fps in most of the tests, which is enough to be considered motion video. Nevertheless, some studies have revealed that on VR applications, a frame rate below 90 fps can cause negative effects to the user. "The lower the frame rate, the worse the effects". In this regard, some VR developers guidelines such as Oculus [44], recommend to aim for consistent 90 fps (which is its max supported frame rate) or to maintain 45 fps as minimum specs. This might be another cause of the uncomfortable video mode visualization. However, the frame rate for virtual mode in all the cases was 316 fps, which is above the recommended 90 fps for VR applications. Again these results are consistent with the premise that virtual mode enhances user experience. Analysis of the virtual mode high frame rate suggests that a higher number, textured, or more complex 3D models can be used without diminishing the performance of the virtual visualization. Such improvement on the 3D models would lead to a more realistic virtual environment. This will be considered in future work.

Based on the observations, navigation and other movements of the real robot were mimicked by the virtual robot very closely. Regarding illumination, tests demonstrated that in poor lighting conditions (e.g. with lights OFF), the robot becomes hard to operate in video mode. In such conditions, the robot's

cameras were not able to provide an appropriate view of the environment. This could lead to damaging collisions between the robot and any component of the remote environment. This issue could have been solved by simply changing to virtual mode without considering the light variation, but the feeling of presence would have been diminished by not reflecting the new environmental conditions of the real environment. In addition, the robot's laser sensors can be used to locate it in a completely dark real-world environment as these sensors do not require visible light to work.

Even in poor lighting, the operation was feasible using laser sensors and real environment information in the virtual environment. Users reported being nearly unable to navigate the robot in the dark using the real mode. When switching to the virtual mode, they regained the ability to navigate, maintaining the feeling of presence by visualizing the mimicked lighting conditions on the virtual environment. Finally, it should be noted that a real, uncontrolled Internet connection was used throughout the experiments. This means that in a real environment, this system shows presumably satisfactory results.

In addition to the tests mentioned in section 4, the system was presented in two international robot events. First, the World Robot Summit (WRS) 2018, held in Tokyo, Japan, where our team obtained 4th place at the Partner Robot Challenge (Virtual Space) of the Service Robotics Category (Figure 15). Second, the RoboCup 2019, held in Sydney, Australia, where our team also obtained 4th place at the Domestic Standard Platform of the @Home League. In both events, the teleoperation system was presented as a Demo even when still under development. Some attendants to the events were able to try the system and commented about it briefly.

Fig. 15. World Robot Summit 2018 Tokyo, Japan. Teleoperation System Demo

In summary, they expressed that the system was easy to learn to manipulate thanks to the joystick controller, which provided the feeling of

controlling a video game. The 3D visualization was one of the features that the users enjoyed the most, a significant part of the testers had never experienced or had little experience with a 3D head-mounted display such as Oculus. The real visualization was well received, but the delays caused by latency bothered the users. The main complaints were dizziness, fatigue, and little frustration when trying to perform a task speedily. By switching to virtual mode, all the users felt a substantial improvement, the dizziness and fatigue completely disappeared. The performance also improved, and frustration diminished. After the users felt comfortable with the double-visualization mode, they switched at will from one mode to another to complete the desired tasks. Most of the users were satisfied with the experience and declared that they would use this system in real-life situations. They also expressed happiness when using the system as it sometimes felt like a video game experience. In addition, the users' comments were of great utility when polishing the system features and planning new ones. All these observations match with the comments from the SUS questionnaire applied, which suggests the SUS results are reliable.

6. Conclusion. Teleoperated service robots are an excellent option for combining human intelligence and motion skills with the service robots' features. We developed an immersive mixed reality teleoperation system for service robots. This system consists of an interface for robot operation, a communication interface between the robot and the operation interface, and an interactive virtual environment mapping the real environment for the operator interface. In the operation interface, the operator can visualize the remote environment in two modes: 3D video from the robot's stereo cameras and a virtual reality environment that shows a presumably accurate representation of the real environment. The operator can select the visualization mode that best suits the ongoing task and visualize it on the HMD. The operator can control the robot's head movement by moving its head, and other movements using a joystick.

As key contributions of this work we can mention the operator's immersive feeling of presence, the use of consumer-grade hardware and software to operate, enhanced dual visualization of the environment (real 3D and virtual 3D), the comfortable operation by regular joysticks (video-game-like experience) and head controlled robot movement, the latency's unpleasant effects reduced by the use of virtual mode, and the ability to operate the robot under poor lighting conditions without sacrificing the operator's feeling of presence.

For future work, some improvements are being considered. Improvements on 3D models for a more realistic experience in the virtual environment. Real object recognition and virtual representation using deep learning techniques. Person recognition and its representation with avatars in the virtual

environment to add human-robot interaction. In addition, there is some ongoing research about automatic virtual environment creation which would be beneficial to add to this work.

The use of mixed reality techniques, such as that proposed in this paper, can greatly improve the operator's user experience, which may lead to broader use of teleoperated service robot systems since it becomes easier for the operator to learn and achieve more complex tasks with ease.

References

1. Niemeyer G., Preusche C., Stramigioli S., and Lee D. Telerobotics. Springer International Publishing, Cham, 2016, pp. 1085–1108.
2. Green M.C., and McAllister C.A. Presence. American Cancer Society, 2020, pp. 1–5.
3. Toet A., Kuling I.A., Krom B.N., and van Erp J. B.F. Toward enhanced teleoperation through embodiment. Frontiers in Robotics and AI 7 (2020), 14.
4. Lichiardopol S. A survey on teleoperation. Technische Universitat Eindhoven, DCT report 20 (2007), 40–60.
5. Stotko P., Krumpen S., Schwarz M., Lenz C., Behnke S., Klein R., and Weinmann M. A VR system for immersive teleoperation and live exploration with a mobile robot. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Nov 2019).
6. Almeida L., Menezes P., and Dias J. Improving robot teleoperation experience via immersive interfaces. In 2017 4th Experiment@International Conference (exp.at'17) (2017), pp. 87–92.
7. Chen J., Glover M., Li C., and Yang C. Development of a user experience enhanced teleoperation approach. In 2016 International Conference on Advanced Robotics and Mechatronics (ICARM) (2016), pp. 171–177.
8. Schäfer A., Reis G., and Stricker D. Investigating the sense of presence between handcrafted and panorama based virtual environments. preprint arXiv:2107.03823 (2021).
9. Mütterlein J., and Hess T. Immersion, presence, interactivity: Towards a joint understanding of factors influencing virtual reality acceptance and use. Twenty-third Americas Conference on Information Systems (2017).
10. Gradecki J. The Virtual Reality Programmer's Kit. Wiley, 1994.
11. Hetrick R., Amerson N., Kim B., Rosen E., Visser E.J.d., and Phillips E. Comparing virtual reality interfaces for the teleoperation of robots. In 2020 Systems and Information Engineering Design Symposium (SIEDS) (2020), pp. 1–7.
12. Whitney D., Rosen E., Ullman D., Phillips E., and Tellex S. Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018), IEEE, pp. 1–9.
13. Lee J., Kim M., and Kim J. A study on immersion and vr sickness in walking interaction for immersive virtual reality applications. Symmetry 9, 5 (2017).
14. Rogers H., Khasawneh A., Bertrand J., and Madathil K.C. An investigation of the effect of latency on the operator's trust and performance for manual multi-robot teleoperated tasks. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (2017), vol. 61, SAGE Publications Sage CA: Los Angeles, CA, pp. 390–394.
15. Sheridan T.B., and Ferrell W.R. Remote manipulative control with transmission delay. IEEE Transactions on Human Factors in Electronics (1963), 25–29.
16. Lane J.C., Carignan C.R., Sullivan B.R., Akin D.L., Hunt T., and Cohen R. Effects of time delay on telerobotic control of neutral buoyancy vehicles. In Proceedings 2002 IEEE

International Conference on Robotics and Automation (Cat. No. 02CH37292) (2002), vol. 3, IEEE, pp. 2874–2879.
17. Bouzakaria N., Concolato C., and Le Feuvre J. Overhead and performance of low latency live streaming using MPEG-DASH. In IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications (2014), IEEE, pp. 92–97.
18. Kebria P.M., Abdi H., Dalvand M.M., Khosravi A., and Nahavandi S. Control methods for internet-based teleoperation systems: A review. IEEE Transactions on Human-Machine Systems 49, 1 (2019), 32–46.
19. Kebria P.M., Khosravi A., Nahavandi S., Shi P., and Alizadehsani R. Robust adaptive control scheme for teleoperation systems with delay and uncertainties. IEEE Transactions on Cybernetics 50, 7 (2020), 3243–3253.
20. Young J., Sharlin E., and Igarashi T. What is mixed reality, anyway? considering the boundaries of mixed reality in the context of robots. In Mixed Reality and Human-Robot Interaction. Springer, 2011, pp. 1–11.
21. Milgram P., and Kishino F. A taxonomy of mixed reality visual displays. IEICE TRANSACTIONS on Information and Systems 77, 12 (1994), 1321–1329.
22. Lipton J.I., Fay A.J., and Rus D. Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. IEEE Robotics and Automation Letters 3, 1 (2018), 179–186.
23. Wen M.C., Yang C.H., Tsai M.H., and Kang S.C. Teleyes: A telepresence system based on stereoscopic vision and head motion tracking. Automation in Construction 89 (2018), 199–213.
24. Wilson B., Bounds M., McFadden D., Regenbrecht J., Ohenhen L., Tavakkoli A., and Loffredo D. VETO: An immersive virtual environment for tele-operation. Robotics 7, 2 (2018), 26.
25. Yamamoto T., Nishino T., Kajima H., Ohta M., and Ikeda K. Human support robot (HSR). In ACM SIGGRAPH 2018 Emerging Technologies (New York, NY, USA, 2018), SIGGRAPH '18, ACM, pp. 11:1–11:2.
26. Kitano H., Asada M., Kuniyoshi Y., Noda I., and Osawa E. Robocup: The robot world cup initiative. In Proceedings of the first international conference on Autonomous agents (1997), pp. 340–347.
27. Shibata T. Head mounted display. Displays 23, 1-2 (2002), 57–64.
28. Desai P.R., Desai P.N., Ajmera K.D., and Mehta K. A review paper on Oculus Rift-A Virtual Reality Headset, arXiv preprint arXiv:1408.1173 (2014).
29. Webster R., and Jr. J. F.D. System usability scale (SUS): Oculus Rift® DK2 and Samsung Gear VR®. In 2017 ASEE Annual Conference & Exposition (Columbus, Ohio, June 2017), ASEE Conferences. https://peer.asee.org/28899.
30. Parisi D. A counterrevolution in the hands: The console controller as an ergonomic branding mechanism. Journal of Games Criticism 2, 1 (2015), 1–23.
31. Koubâa A. Robot Operating System (ROS)., vol. 1. Springer, 2019.
32. Savage J., LLarena A., Carrera G., Cuellar S., Esparza D., Minami Y., and Peñuelas U. ViRbot: A system for the operation of mobile robots. In RoboCup 2007: Robot Soccer World Cup XI (Berlin, Heidelberg, 2008), U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds., Springer Berlin Heidelberg, pp. 512–519.
33. Creighton R. H. Unity 3D game development by example: A Seat-of-your-pants manual for building fun, groovy little games quickly. Packt Publishing Ltd, 2010.
34. Fisher R., Perkins S., Walker A., and Wolfart E. Laplacian/laplacian of gaussian. Hypermedia Image Processing Reference (2003).
35. Kessler G.D., and Hodges L.F. A network communication protocol for distributed virtual environment systems. In Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium (1996), pp. 214–221.

36. SIGVerse. http://www.sigverse.org/wiki/en/
37. Inamura T., Shibata T., Sena H. Hashimoto T., Kawai N., Miyashita T., Sakurai Y., Shimizu M., Otake M., Hosoda K. Simulator platform that enables social interaction simulation — SIGVerse: SocioIntelliGenesis simulator. In Proceedings of the IEEE/SICE International Symposium on System Integration (2010), pp. 212–217.
38. ROS web video server package. http://wiki.ros.org/web_video_server.
39. Brooke, J., et al. SUS: A quick and dirty usability scale. Usability evaluation in industry 189, 194 (1996), 4–7.
40. Lewis, J. R. The system usability scale: Past, present, and future. International Journal of Human–Computer Interaction 34, 7 (2018), 577–590.
41. Read P., and Meyer M.P. Restoration of motion picture film. Elsevier, 2000.
42. Bangor, A., Kortum, P., and Miller, J. Determining what individual sus scores mean: Adding an adjective rating scale. Journal of usability studies 4, 3 (2009), 114–123.
43. Miller R.B. Response time in man-computer conversational transactions. In AFIPS Fall Joint Computing Conference (1) (1968), vol. 33 of AFIPS Conference Proceedings, AFIPS / ACM / Thomson Book Company, Washington D.C., pp. 267–277.
44. VR performance optimization. https://developer.oculus.com/documentation/native/pc/dg-performance-guidelines/

Nakayama Angelica – Ph.D., Professor, Institute of research in applied mathematics and systems, department of electrical engineering, engineering faculty, National Autonomous University of Mexico. Research interests: autonomous mobile robots, virtual reality, machine learning. The number of publications – 2. annkym@gmail.com; 3000, Sirkuito Eskolar, Av. Universidad, 04510, Mexico City, Mexico; office phone: +52(55)56223041.

Ruelas Daniel – Professor, Institute of research in applied mathematics and systems, sciences faculty, National Autonomous University of Mexico. Research interests: virtual reality, videogame development. The number of publications – 0. sango265@ciencias.unam.mx; 3000, Sirkuito Eskolar, Av. Universidad, 04510, Mexico City, Mexico; office phone: +52(55)56223041.

Savage Jesus – Ph.D., Professor, Electrical engineering department, National Autonomous University of Mexico. Research interests: autonomous mobile robots, digital signal processing, computer architectures. The number of publications – 0. robotssavage@gmail.com; 3000, Sirkuito Eskolar, Av. Universidad, 04510, Mexico City, Mexico; office phone: +52(55)56223041.

Bribiesca Ernesto – Ph.D., Professor, Institute of research in applied mathematics and systems, National Autonomous University of Mexico. Research interests: pattern recognition, computer vision, chain coding. The number of publications – 100. bribiesca@iimas.unam.mx; 3000, Sirkuito Eskolar, Av. Universidad, 04510, Mexico City, Mexico; office phone: +52(55)56223617.

Acknowledgements. This research was supported by PAPIIT-DGAPA UNAM, Mexico, under Grant AG101721, and by CONACYT Mexico Graduate Scholarship Program.

DOI 10.15622/ia.20.6.1

А. НАКАЯМА , Д. РУЭЛАС , Х. САВАЖ , Э. БРИБЕСКА

ДИСТАНЦИОННО УПРАВЛЯЕМЫЙ СЕРВИСНЫЙ РОБОТ С ИММЕРСИВНЫМ ИНТЕРФЕЙСОМ СМЕШАННОЙ РЕАЛЬНОСТИ

Накаяма А., Руэлас Д., Саваж Х., Брибеска Э. **Дистанционно управляемый сервисный робот с иммерсивным интерфейсом смешанной реальности.**

Аннотация. Сервисные роботы с дистанционным управлением могут выполнять более сложные и точные задачи, поскольку они сочетают в себе навыки робота и человеческий опыт. Связь между оператором и роботом важна для удаленной работы и сильно влияет на эффективность системы. Существует мнение, что улучшение ощущения присутствия оператора также улучшает выполнение задачи. Иммерсивные интерфейсы используются для улучшения опыта удаленной работы, поскольку ощущение присутствия является результатом погружения. Однако задержка или временная задержка могут снизить производительность робота. Временная задержка между входом и визуальной обратной связью сильно влияет на обмен данными между распределенными ведущими и ведомыми системами по сети. Поскольку удаленная визуализация включает в себя передачу большого количества видеоданных, проблема заключается в снижении нестабильности связи. Затем эффективная система дистанционного управления должна иметь подходящий рабочий интерфейс, способный визуализировать удаленную среду, управлять роботом и иметь быстрое время отклика. Эта работа представляет собой разработку системы дистанционного управления сервисным роботом с иммерсивным операционным интерфейсом смешанной реальности, где оператор может визуализировать реальную удаленную среду или виртуальную трехмерную среду, представляющую ее. Виртуальная среда направлена на сокращение задержки при обмене данными за счет уменьшения объема информации, отправляемой по сети, и улучшения взаимодействия с пользователем. Робот может выполнять навигацию и простые задачи автономно или переключаться в дистанционно управляемый режим для более сложных задач. Система была разработана с использованием ROS, UNITY 3D и сокетов для легкого экспорта на различные платформы. Эксперименты показывают, что наличие иммерсивного рабочего интерфейса повышает удобство использования для оператора. Задержка при использовании виртуальной среды увеличивается. Пользовательский опыт улучшается за счет использования техник смешанной реальности; это может привести к более широкому использованию дистанционно управляемых систем сервисных роботов.

Ключевые слова: дистанционно управляемый робот, сервисный робот, иммерсивный рабочий интерфейс, интерфейс смешанной реальности, среда виртуальной реальности

Накаяма Анжелика – профессор, институт исследований прикладной математики и систем, отдел электротехники, инженерный факультет, Национальный автономный университет Мексики (НАУМ). Область научных интересов: автономные мобильные роботы, виртуальная реальность, машинное обучение. Число научных публикаций – 2. annkym@gmail.com; Сиркуито Эсколар, проспект Университад, 3000, 04510, Мехико, Мексика; р.т.: +52(55)56223041.

Руэлас Дэниел – профессор, институт исследований прикладной математики и систем, факультет естественных наук, Национальный автономный университет Мексики (НАУМ). Область научных интересов: виртуальная реальность, разработка видеоигр. Число научных публикаций – 0. sango265@ciencias.unam.mx; Сиркуито Эсколар, проспект Университад, 3000, 04510, Мехико, Мексика; р.т.: +52(55)56223041.

Саваж Хесус – профессор, электротехнический отдел, Национальный автономный университет Мексики (НАУМ). Область научных интересов: автономные мобильные роботы,

цифровая обработка сигналов, компьютерная архитектура. Число научных публикаций – 0. robotssavage@gmail.com; Сиркуито Эсколар, проспект Универсидад, 3000, 04510, Мехико, Мексика; р.т.: +52(55)56223041.

Брибеска Эрнесто – профессор, институт исследований прикладной математики и систем, Национальный автономный университет Мексики (НАУМ). Область научных интересов: распознавание образов, компьютерное зрение, цепное кодирование. Число научных публикаций – 100. bribiesca@iimas.unam.mx; Сиркуито Эсколар, проспект Универсидад, 3000, 04510, Мехико, Мексика; р.т.: +52(55)56223617.

Поддержка исследований. Это исследование было поддержано программой PAPIIT-DGAPA UNAM в Мексике, в рамках Гранта AG101721, и стипендиальной программой для выпускников CONACYT в Мексике.

Литература

1. Niemeyer G., Preusche C., Stramigioli S., and Lee D. Telerobotics. Springer International Publishing, Cham, 2016, pp. 1085–1108.
2. Green M.C., and McAllister C.A. Presence. American Cancer Society, 2020, pp. 1–5.
3. Toet A., Kuling I.A., Krom B.N., and van Erp J. B.F. Toward enhanced teleoperation through embodiment. Frontiers in Robotics and AI 7 (2020), 14.
4. Lichiardopol S. A survey on teleoperation. Technische Universitat Eindhoven, DCT report 20 (2007), 40–60.
5. Stotko P., Krumpen S., Schwarz M., Lenz C., Behnke S., Klein R., and Weinmann M. A VR system for immersive teleoperation and live exploration with a mobile robot. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (Nov 2019).
6. Almeida L., Menezes P., and Dias J. Improving robot teleoperation experience via immersive interfaces. In 2017 4th Experiment@International Conference (exp.at'17) (2017), pp. 87–92.
7. Chen J., Glover M., Li C., and Yang C. Development of a user experience enhanced teleoperation approach. In 2016 International Conference on Advanced Robotics and Mechatronics (ICARM) (2016), pp. 171–177.
8. Schäfer A., Reis G., and Stricker D. Investigating the sense of presence between handcrafted and panorama based virtual environments. preprint arXiv:2107.03823 (2021).
9. Mütterlein J., and Hess T. Immersion, presence, interactivity: Towards a joint understanding of factors influencing virtual reality acceptance and use. Twenty-third Americas Conference on Information Systems (2017).
10. Gradecki J. The Virtual Reality Programmer's Kit. Wiley, 1994.
11. Hetrick R., Amerson N., Kim B., Rosen E., Visser E.J.d., and Phillips E. Comparing virtual reality interfaces for the teleoperation of robots. In 2020 Systems and Information Engineering Design Symposium (SIEDS) (2020), pp. 1–7.
12. Whitney D., Rosen E., Ullman D., Phillips E., and Tellex S. Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2018), IEEE, pp. 1–9.
13. Lee J., Kim M., and Kim J. A study on immersion and vr sickness in walking interaction for immersive virtual reality applications. Symmetry 9, 5 (2017).
14. Rogers H., Khasawneh A., Bertrand J., and Madathil K.C. An investigation of the effect of latency on the operator's trust and performance for manual multi-robot teleoperated tasks. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (2017), vol. 61, SAGE Publications Sage CA: Los Angeles, CA, pp. 390–394.

15. Sheridan T.B., and Ferrell W.R. Remote manipulative control with transmission delay. IEEE Transactions on Human Factors in Electronics (1963), 25–29.
16. Lane J.C., Carignan C.R., Sullivan B.R., Akin D.L., Hunt T., and Cohen R. Effects of time delay on telerobotic control of neutral buoyancy vehicles. In Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292) (2002), vol. 3, IEEE, pp. 2874–2879.
17. Bouzakaria N., Concolato C., and Le Feuvre J. Overhead and performance of low latency live streaming using MPEG-DASH. In IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications (2014), IEEE, pp. 92–97.
18. Kebria P.M., Abdi H., Dalvand M.M., Khosravi A., and Nahavandi S. Control methods for internet-based teleoperation systems: A review. IEEE Transactions on Human-Machine Systems 49, 1 (2019), 32–46.
19. Kebria P.M., Khosravi A., Nahavandi S., Shi P., and Alizadehsani R. Robust adaptive control scheme for teleoperation systems with delay and uncertainties. IEEE Transactions on Cybernetics 50, 7 (2020), 3243–3253.
20. Young J., Sharlin E., and Igarashi T. What is mixed reality, anyway? considering the boundaries of mixed reality in the context of robots. In Mixed Reality and Human-Robot Interaction. Springer, 2011, pp. 1–11.
21. Milgram P., and Kishino F. A taxonomy of mixed reality visual displays. IEICE TRANSACTIONS on Information and Systems 77, 12 (1994), 1321–1329.
22. Lipton J.I., Fay A.J., and Rus D. Baxter's homunculus: Virtual reality spaces for teleoperation in manufacturing. IEEE Robotics and Automation Letters 3, 1 (2018), 179–186.
23. Wen M.C., Yang C.H., Tsai M.H., and Kang S.C. Teleyes: A telepresence system based on stereoscopic vision and head motion tracking. Automation in Construction 89 (2018), 199–213.
24. Wilson B., Bounds M., McFadden D., Regenbrecht J., Ohenhen L., Tavakkoli A., and Loffredo D. VETO: An immersive virtual environment for tele-operation. Robotics 7, 2 (2018), 26.
25. Yamamoto T., Nishino T., Kajima H., Ohta M., and Ikeda K. Human support robot (HSR). In ACM SIGGRAPH 2018 Emerging Technologies (New York, NY, USA, 2018), SIGGRAPH '18, ACM, pp. 11:1–11:2.
26. Kitano H., Asada M., Kuniyoshi Y., Noda I., and Osawa E. Robocup: The robot world cup initiative. In Proceedings of the first international conference on Autonomous agents (1997), pp. 340–347.
27. Shibata T. Head mounted display. Displays 23, 1-2 (2002), 57–64.
28. Desai P.R., Desai P.N., Ajmera K.D., and Mehta K. A review paper on Oculus Rift-A Virtual Reality Headset, arXiv preprint arXiv:1408.1173 (2014).
29. Webster R., and Jr. J. F.D. System usability scale (SUS): Oculus Rift® DK2 and Samsung Gear VR®. In 2017 ASEE Annual Conference & Exposition (Columbus, Ohio, June 2017), ASEE Conferences. https://peer.asee.org/28899.
30. Parisi D. A counterrevolution in the hands: The console controller as an ergonomic branding mechanism. Journal of Games Criticism 2, 1 (2015), 1–23.
31. Koubâa A. Robot Operating System (ROS)., vol. 1. Springer, 2019.
32. Savage J., LLarena A., Carrera G., Cuellar S., Esparza D., Minami Y., and Peñuelas U. ViRbot: A system for the operation of mobile robots. In RoboCup 2007: Robot Soccer World Cup XI (Berlin, Heidelberg, 2008), U. Visser, F. Ribeiro, T. Ohashi, and F. Dellaert, Eds., Springer Berlin Heidelberg, pp. 512–519.
33. Creighton R. H. Unity 3D game development by example: A Seat-of-your-pants manual for building fun, groovy little games quickly. Packt Publishing Ltd, 2010.

34. Fisher R., Perkins S., Walker A., and Wolfart E. Laplacian/laplacian of gaussian. Hypermedia Image Processing Reference (2003).
35. Kessler G.D., and Hodges L.F. A network communication protocol for distributed virtual environment systems. In Proceedings of the IEEE 1996 Virtual Reality Annual International Symposium (1996), pp. 214–221.
36. SIGVerse. http://www.sigverse.org/wiki/en/
37. Inamura T., Shibata T., Sena H. Hashimoto T., Kawai N., Miyashita T., Sakurai Y., Shimizu M., Otake M., Hosoda K. Simulator platform that enables social interaction simulation — SIGVerse: SocioIntelliGenesis simulator. In Proceedings of the IEEE/SICE International Symposium on System Integration (2010), pp. 212–217.
38. ROS web video server package. http://wiki.ros.org/web_video_server.
39. Brooke, J., et al. SUS: A quick and dirty usability scale. Usability evaluation in industry 189, 194 (1996), 4–7.
40. Lewis, J. R. The system usability scale: Past, present, and future. International Journal of Human–Computer Interaction 34, 7 (2018), 577–590.
41. Read P., and Meyer M.P. Restoration of motion picture film. Elsevier, 2000.
42. Bangor, A., Kortum, P., and Miller, J. Determining what individual sus scores mean: Adding an adjective rating scale. Journal of usability studies 4, 3 (2009), 114–123.
43. Miller R.B. Response time in man-computer conversational transactions. In AFIPS Fall Joint Computing Conference (1) (1968), vol. 33 of AFIPS Conference Proceedings, AFIPS / ACM / Thomson Book Company, Washington D.C., pp. 267–277.
44. VR performance optimization. https://developer.oculus.com/documentation/native/pc/dg-performance-guidelines/