The International Journal of Advanced Manufacturing Technology (2025) 139:6233–6257
https://doi.org/10.1007/s00170-025-16236-w

ORIGINAL ARTICLE

Enhancing real-time robot teleoperation with immersive virtual reality in industrial IoT networks

Toan Luu[1] · Quang Nguyen[1] · Thien Tran[1,3] · Minh Tran[1] · Songlin Ding[2] · Jonathan Kua[3] · Thuong Hoang[3]

Received: 27 December 2023 / Accepted: 31 July 2025 / Published online: 13 August 2025
© The Author(s) 2025

Abstract
During the early stages of the Industry 4.0 era, teleoperation with human involvement in manufacturing processes and robot operation remained important. The existing teleoperation technologies have limitations in providing operators with spatial awareness and initiative control mechanisms, making their applications difficult in tasks with high complexity and precision. This study implements the advancements from the internet of things (IoT) and virtual reality (VR) technology to improve the real-time teleoperation for industrial robots. The proposed design utilizes the benefits of the Message Queuing Telemetry Transport (MQTT) protocol in conjunction with a cloud broker, which has not been addressed significantly in previous studies for teleoperation. This setup enables the system to efficiently handle substantial amounts of data in short periods of time, allowing for an uninterrupted user experience while maintaining a consistent connection regardless of distance. Real-time robot control is accomplished by tracking the user's hand movements with a VR controller while providing the user with visual feedback from the VR environment. The incorporation of VR technology intends to improve the intuitiveness of controlling robot operations, provide more responsive feedback, and improve the quality of work performed by operators. Practical experiments are conducted with an industrial robot to observe the performance against the system under different internet conditions and different QoS setups. The evaluation criteria primarily concern about latency and accuracy. The experiment aims to evaluate the effectiveness and scalability of the suggested system and to offer significant insight into its advantages and limitations relative to existing control system through systematic investigation.

Keywords Industrial robot · Teleoperation · Virtual reality · Internet of things · MQTT · QoS · Telerobotics

1 Introduction

As part of the fourth industrial revolution (Industry 4.0), articulated robots have seen significant applications in a variety of industries, including manufacturing, material handling, warehousing, agriculture, and others [1–4]. With the improvements in automation technology, robots are enabled to execute their task autonomously. However, the human factor should not be overlook as human influence is necessary for the first steps as well as supervision during work execution. Many of these robotic systems necessitate an initial teaching phase, during which preset spatial coordinates are required to guide the robot's operations [5]. Several computer-aided manufacturing (CAM) systems that rely heavily on industrial robots are moving away from robot-specific programming languages and toward teaching pendants as more user-friendly options for inputting essential spatial data and generating component trajectories. This method needs the operator to manually manipulate the robot to facilitate it to reach a specific place or follow a predefined trajectory. This series of commands is then stored in the robot's controller, allowing it to reproduce and execute the action autonomously [6]. As a result, the initial stages of robot operation necessitate human instruction followed by subsequent monitoring processes to solve any possible problem that occurs. In order to guarantee timely human intervention when the system encounters issues, teleoperation integration is recommended. With the ability to provide human monitoring and instruction despite the geographical distance and the potentially hazardous conditions within the operational

✉ Minh Tran
minh.tranquang@rmit.edu.vn

1 School of Science, Engineering and Technology, RMIT Vietnam, Ho Chi Minh City, Vietnam

2 School of Engineering, RMIT Australia, Melbourne, Australia

3 School of Information Technology, Deakin University, Burwood, Australia

environment, an optimal teleoperation solution can increase safety and stability and reduce the operation cost for any robotic system.

Despite the numerous benefits of teleoperation systems, experts have raised concerns regarding their applicability, effectiveness, and the optimization challenges [7, 8]. Primarily, the control mechanism should be designed to be user-friendly, responsive, and intuitive [9]. It must replicate the dexterity skills of human hands to allow users to provide precise commands comfortably. Furthermore, the safety of teleoperation systems receives many precautions and evaluations [10]. For instance, collision risks increase due to the operator's reduced awareness of the robot's surroundings, and insufficient information about the robot's condition and status may lead to operational errors. To effectively resolve misinformation challenges, teleoperation systems must incorporate comprehensive visual feedback and provide detailed awareness of the robot's status and the surroundings.

The widespread of immersive technologies has created opportunities for the integration of virtual reality (VR) into remote control systems. This integration has the potential to overcome existing limitations and improve the user experience. Specifically, VR can significantly enhance both control and feedback mechanisms [11, 12]. Compared to traditional control devices such as joysticks, control pads, mice, and keyboards, VR controllers offer notable advantages [13–15]. While traditional methods remain practical for industrial tasks, their limited responsiveness in three dimensions can make them less intuitive for new users, particularly in remote-control scenarios where users lack a direct line of sight to the robot [16, 17]. In contrast, VR controllers enable users to direct the robot's end effector through hand gestures, providing an intuitive and natural control approach that facilitates seamless human-robot interaction [18]. This method gives the impression that the robot directly responds to human hand movements. Since the human arm is a seven-degree-of-freedom structure, gesture-based control can replicate and guide most articulated industrial robots effectively [19, 20].

Additionally, a 3D model of the robot within a virtual environment can enhance the operators understanding of the status of the robot, particularly by improving depth perception [21]. The immersive nature of VR allows users to move within the virtual space, observing the robot from different angles to adapt to the task at hand, thereby eliminating the challenges posed by conventional 2D video streams. Traditional 2D visual feedback often lacks sufficient depth perception and obscures certain angles, especially when observing robot movements along the camera's axis. Although using multiple camera perspectives could mitigate this limitation, it introduces an overwhelming influx of information for the operator. Processing and filtering relevant details from multiple video feeds imposes a significant cognitive load on the user, leading to inefficiencies and a reliance on the operator's experience and skill [22, 23].

The full potential of a VR system cannot be realized without a consistent and fast connection between the robot and the operator [24]. Network issues, such as high latency, packet loss, and bandwidth throttling, can significantly degrade task performance by disrupting the consistent exchange of data between the operator and the robot, leading to delays in execution, inaccuracies in control, and a reduced ability to provide timely feedback. Latency introduces a time delay between the user's input and the feedback, increasing the risk of the operator overshooting the target and potentially causing collision accidents. Other issues such as packet loss and throttling prevent the robot from receiving complete instructions from the user, while feedback information may fail to reach the operator in a timely manner.

To enhance the communication between the robot and the user, the MQTT protocol could be considered. MQTT is not a new technique; it has been used wisely in IoT and in multiple different applications such as industrial, environment monitoring, agriculture, and smart housing [25–28]. A significant advantage of MQTT compared to other methods, such as HTTP, is its lightweight, efficiency, and versatility. Being a message-oriented protocol and using the publish/subscribe model, MQTT can provide the system with flexibility and the ability to function in high-frequency applications [29]. Furthermore, MQTT include robust error handling and the ability to perform efficiently in low bandwidth and unstable networks. The manifold utility of MQTT makes it highly suitable for a precise real-time control system in industrial applications.

The combination of VR and MQTT technology offers a promising approach to addressing the limitations of traditional teleoperation systems in industrial settings. VR, through its immersive features, provides users with an accurate and detailed telepresence of both the robotic system and its surrounding environment. Additionally, the virtual environment overcomes the constraints of 2D interfaces, significantly improving task performance and reducing error rates by enhancing spatial awareness, minimizing cognitive load, and offering a more intuitive and engaging control interface. The gesture control mechanism of the VR controller preserves the dexterity of the human hand, ensuring efficiency and precision when performing complex and intricate tasks.

However, the use of VR technology necessitates the transfer of a larger amount of data points to ensure that the robot can accurately replicate the operator's movements. As a result, any issues with data transmission could have a greater impact on the overall system performance. With its capability for multi-device connectivity and near real-time data streaming, MQTT facilitates the rapid transmission of essential parameters from the robotic system to the user. This

uninterrupted data flow, seamlessly integrated into the VR environment, enhances the operator's situational awareness and understanding of the remote worksite. Furthermore, the error-handling features of MQTT provide an additional layer of reliability, minimizing the potential for errors and thereby improving the system's overall stability and robustness.

The goal of this work is to develop a system that combines the benefits of VR technology and MQTT protocol to improve the teleoperation system and apply it to articulated robotic platforms. This research distinguishes itself by its concentrated investigation into the system communication architecture, specifically exploring the feasibility and advantages of leveraging the MQTT protocol within an industrial internet of things (IIoT) framework for the teleoperation of articulated robotic platforms.

The proposed system allows the remote operation of the ABB IRB1200 robot on a global scale via the Internet. The operator uses the HTC Vive VR kit to manipulate and monitor the robot's status. Furthermore, this research looks into the implementation of the MQTT protocol and the development of a dedicated broker to investigate the performance of this technology in the industrial environments. The study executes experiments that examine, evaluate, and compare the movement of the robot's end effector with the user input under varying Internet conditions. The experiment seeks to examine the adverse effects of network impairments on robot movements, including latency, throttling, and packet loss. It explores various MQTT configurations to conduct a comprehensive evaluation and identify the most suitable setup for different internet conditions.

This article is structured as follows: Section 1 outlines the research aims and motivation, establishing the context and significance of this work. In Sect. 2, a comprehensive literature review is presented, covering key areas such as VR and communication. Section 3 delves into the design and architecture of the whole proposed system. Subsequently, Sect. 4 presents the experiments setup and procedure to test the system feasibility. A discussion of the testing results, findings, and limitations is presented in Sect. 5, exploring the challenges and issues of the proposed platform. Finally, Sect. 6 concludes the article by summarizing the key findings, drawing conclusions, and outlining potential avenues for future research and development.

2 Literature review

2.1 Virtual reality

VR technology has been extensively investigated for its applications across various sectors, particularly those involving human interaction. The effectiveness of immersive technologies stems from their capacity to replicate objects and environments within a three-dimensional immersive space, making them suitable for simulating robotic movements, states, and interactions in detail. This capability is especially advantageous for enhancing spatial awareness and user perception, as demonstrated in multiple fields. For instance, our previous studies [30, 31] introduced the immersive cobot teleoperation platform with mixed reality (MR) and investigated the user experience of two novice groups: one trained with conventional classroom methods and the other using an MR collaborative training approach in immersive 360 augmented training environment. The outcome demonstrated that MR-based instruction enables greater engagement, increases learner confidence, and promotes a deeper understanding of system operations than traditional teaching methods. Similarly, an AR-aided training platform for industrial assembly demonstrated enhanced training outcomes, with trainees exhibiting faster task completion times while simultaneously minimizing errors [32]. Additionally, studies of Eswaran M. present that immersive technologies are effective for visualizing, testing, and simulating automation and assembly line designs, allowing users to test and refine systems before deploying them in real-world production environments [33–36]. Several other studies have implemented VR technology to enhance visualization in robotic control tasks, and the outcomes showed positivity. Yunpeng Su [37] studies have compared the worker performance in the pick and place task. The findings indicate that combining 3D model visualization with a traditional 2D view results in better performance compared to using only a 2D view. Participants completed tasks more quickly and experienced greater comfort when provided with 3D visual feedback. Moreover, research from Xiyao Wang [38] suggests that user's subjective feedback about the experience with 3D visualization could be negative, due to hardware limitations or a lack of experience. However, their task accuracy improved significantly, while execution times remained comparable to using 2D visualizations.

In addition to providing more immersive feedback, VR hardware can enhance user control experiences. The structural similarity between human hands and articulated robots allows gesture-based control to be a fast and natural control method for users [19, 39, 40]. Several studies have investigated how gesture control can improve robot control mechanisms, aiming to offer users a more intuitive and comfortable experience [41, 42]. In research for teleoperation and control for the human assistant systems, González et al. [43] find that integrating haptic feedback enables users to better adapt to and handle complex manufacturing tasks, such as finishing, sanding, and grinding. Similarly, Chen's research [44] demonstrates that most participants reported reduced physical and mental demands when using VR controllers compared to traditional graphical user interfaces for robotic control. Therefore, VR handheld controllers are typi-

cally equipped with sensors, accelerometers, and gyroscopes to capture hand gestures, making them highly suitable as input devices for robotic control.

The remote-control system can benefit from both strengths of VR technology, as users can easily recognize the surrounding environment and the state of the robot, while the control mechanism allows them to easily manipulate, improvise, and adjust easily. Many systems and frameworks have been researched and proposed, demonstrating the potential of VR teleoperation in many different fields. The proposed real-time teleoperation Vicarios system from Abdeldjallil Naceri [45] uses an HTC Vive Pro headset, integrated with an RGB-D camera to provide 3D pose and dimension information of the remote workstation and object. The system has demonstrated the advantage of VR technology when it improves user experience and the success rate of users in robot maneuver tasks. Md Tahmid Bin Touhid and his team [46] present a low-cost framework with the utilization of the Google Cloud platform to create a cloud-based digital twin system with remote control and monitor ability for smart manufacture and assembly lines. With the advantage of virtual reality, teleoperation robot platforms have become particularly useful for hazardous and obstructed environments, allowing digital transformation and improved scalability. Yunpeng Su et al. [47] proposed a VR platform with a haptic input device and hidden Markov models algorithm to improve performance and intuitiveness in remote welding operations. Yuan Sun et al. implement VR [48] to support construction-related tasks with mobile robots. Leveraging inferential technique, VR technology enables operators to visualize conceptual visualization of both proximate and distant human-robot interactions within construction settings.

2.2 Telecommunication

Maintaining a reliable connection between the operator and the robot is an essential factor in teleoperation systems. Stable connectivity, minimal data transfer delays, and the prevention of data loss are vital to ensure smooth operation and prevent errors. For local network configurations, LAN Ethernet is commonly chosen for its low latency in data transfer between input devices and the robot. Common protocols for communication in local network setup are traditional protocols such as TCP/IP or UDP/IP. One example is Conrad Fifelski-von Böhlen et al. [49] telemanipulation platform in caregiving. The platform utilized TCP for transmitting all the data from multiple depth cameras to the main processing computer for virtual environment generation. The platform maintained 27 FPS and 250ms latency at 2.5 meters. Another popular implementation of communication is the use of ROS bridge and ROS Sharp framework [50]. These frameworks from ROS provide developers with JSON API as the robot data structure and WebSocket package for remote communication and web browser interaction. In Francesco De Pace et al. [51] enhanced virtual reality framework, while point cloud data from the RGB-D camera are still streamed over a custom solution based on TCP/UDP protocols, the robot data itself is exchanged using the ROS# Library on LAN network.

Another communication aspect that needs consideration is communication architectures, which form the foundational framework for information exchange. Each architecture has its own distinct characteristics and suitability for specific applications. Centralized architectures are some of the most common choices not only in teleportation but also in cloud computing and web services. An example of centralized architecture is server-client architecture, which were implemented in Wen Fan's mixed reality framework for immersive teleoperation [52]. The framework utilizes Websocket protocol, where the physical robot acts as the server and the digital twin will generate the control command and sent it to the server as a client through a Unity script. Another popular selection is peer-to-peer architecture, characterized by a network topology with a relatively small number of nodes. David Black's mixed reality teleoperation system [53] demonstrated the advantages of WebRTC's peer-to-peer architecture, namely its reliable and low-latency communication. Publish-subscribe systems are becoming more popular, especially with machine-to-machine communication due to efficient dissemination of information to interested subscribers, facilitating scalable and flexible information distribution. Elvis Borges's framework [54] utilizes MQTT protocol and pub-sub architecture to create connection between robot operation system and extended reality glass HoloLens 2. The selection of an appropriate communication architecture is dependent on a range of factors, including scalability requirements, performance demands, security considerations, and fault tolerance.

MQTT is a lightweight messaging protocol built on TCP/IP. It uses a publish/subscribe architecture for machine-to-machine communication and is commonly used in IoT applications [55]. This protocol has a number of advantages that make it ideal for this remote-control application. Firstly, MQTT's compact message format with minimal overhead minimizes latency and facilitates high-frequency data exchange. While performance comparisons with transport-layer protocols like TCP, UDP, and DTLS may not always yield a clear MQTT advantage, its position at the application layer means the protocol came with a feature set well-suited for industrial applications. The IoT architecture of MQTT makes the protocol ideal in applications where a large number of client connections is required such as automation lines [56, 57]. Secondly, the publish/subscribe paradigm, where messages are categorized and delivered based on predefined topics, reduces traffic on the client's network. This, coupled with the small message size, enables the broker to efficiently handle a high volume of messages without demanding exces-

sive hardware resources [58, 59]. Thirdly, MQTT's support for persistent connections, retained messages, and quality of service (QoS) levels enhances its resilience in unreliable network environments with limited bandwidth. These features ensure message delivery and connection persistence between client and server even during network disruptions [60, 61]. Finally, since MQTT is a popular protocol with open-source software, it has multiple libraries available to support different devices [62].

With that strength, MQTT protocol is an rising approach for communication in the industrial environment. Gustavo Caiza [63] proposed a robotic teleoperation architecture for oil and gas following the IEC-61499 standard with MQTT as the primary communication protocol between the gesture sensor and the robot, which results in a scalable platform for real-time control of the robot. Carlos A. Garcia [64] proposed a human-robot collaboration system that utilizes the MQTT protocol for transmitting command signals and feedback information between the user and a KUKA robot. The study yielded positive results, demonstrating the system's ability to accurately follow welding instructions while maintaining a stable connection with no packet loss using MQTT.

2.3 Research gap

While VR technology has been researched and proven to offer numerous benefits for remote control systems, its application in industrial environments could have several further enhancements and optimization. Although 3D controllers provide adaptability and user convenience, they often frequently show instability and increased sensitivity. Unlike simpler 2D systems, such as joystick, where the user is only controlling the robot in the two axis at most, the VR controller constantly tracks the moment in 3D space, allowing the operator to control the robot in all axes at the same time. As the result, control demands higher data rates, more bandwidth-intensive, and lower latency to maintain smooth operation [65, 66]. In addition, any fluctuation or degradation in network quality such as packet loss and bandwidth throttling can significantly impact the responsiveness and accuracy of VR-based systems.

To explore the trend and approaches that are commonly employed in academics, a rapid review is conducted. Firstly, the PRISMA framework was deployed to guide the quantitative review, as it provides a structured approach to assess the usage frequency of different communication protocols. Due to large amount of research in the recent years regarding robotic field, an in-depth analysis of individual studies and meta-analysis were deemed beyond the scope of this rapid review. The review included published, peer-reviewed articles retrieved from the Web of Science database, focusing on research investigating immersive technologies for remote robot control. The search query employed was *AB=((“robot*”) AND (“teleoperat*” OR “remote”) AND (“virtual reality” OR “augmented reality” OR “mixed reality” OR “extended reality” OR “immersive technology” OR VR OR XR OR AR OR ImT OR ImTech))*. The search was limited to the period between 2015 and 2025 to capture the significant rise of commercially available virtual reality headsets during this timeframe. The initial database search yielded 395 articles. After screening abstracts and full-text articles, 72 papers were included in the final analysis. Data extracted from these papers included the communication protocol, network architecture, and network type.

TCP/IP emerged as the most frequently utilized communication protocol, as observed in over a third of the studies. ROSBridge and UDP/IP followed as the second and third most common protocols, respectively. MQTT, conversely, was employed in only 3 out of the 72 papers. Server-client architecture demonstrated the highest prevalence, adopted by over 90% of the studies. Finally, local area networks were the preferred testing environment, with 57 out of 72 papers utilizing this network type (Fig. 1).

Prior research on teleoperation systems has predominantly centered around server-client architectures. While effective in specific scenarios, this approach can exhibit limitations in efficiency, particularly in automation applications involving a large number of robots, sensors, or clients that require connection, monitoring, and control. The publish-

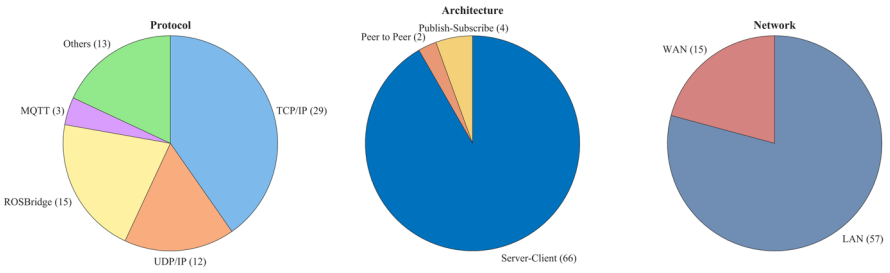


Fig. 1 Quantitative review of past research methodology

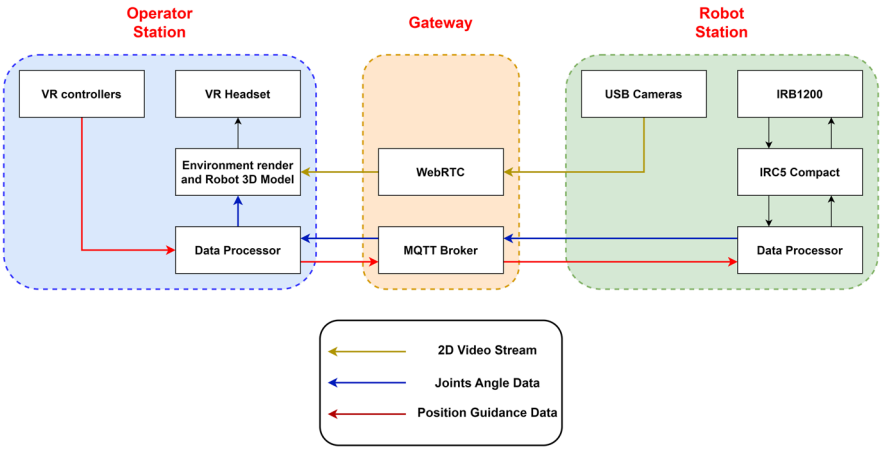


Fig. 2 System architecture

subscribe paradigm, as exemplified by the Message Queuing Telemetry Transport (MQTT) protocol, offers a compelling alternative. MQTT facilitates efficient communication with numerous devices and machines while minimizing network overhead, a critical consideration for distributed teleoperation systems. The integration of a cloud-based broker further enhances scalability, control, and security compared to traditional client–server models. The review of existing literature revealed a limited exploration of MQTT within the context of robotic and manufacturing teleoperation applications. By addressing the inherent limitations of scalability and latency associated with this approach, this work aims to contribute to the advancement of teleoperation technologies. Furthermore, the study included testing in both local and wide area networks to validate the findings.

3 System design

3.1 System architecture

The system can be divided into two stations, the robot station and the operator station. The system architecture and station communication flow can be viewed in Fig. 2.

The robotic station is equipped with several components, including the IRC5 Compact robot controller, which controls the operation of the 6-degree robotic arm ABB IRB1200, as seen in Fig. 3. In addition, the setup includes a personal computer (PC) and two USB cameras. The PC runs a C# application developed on a.NET framework that serves three functions. It primarily receives and processes control commands from the user, which could include information like starting motion, changing speed, and changing direction of movement. Moreover, given the employment of the External Guided Motion (EGM) feature by IRC5 for robot control, the C# program must function as the designated communication endpoint for this capability. The tasks consist of receiving and processing user-specified end-effector positions, including Cartesian position and Quaternion rotation, and then sending them to the controller. Finally, the C# application performs image processing on data acquired from the two USB cameras before transmitting the video feed to the operator station.

On the operator station, a complete setup requires a PC and one HTC Vive VR set, including 1 VR headset, 2 VR controllers, and 2 VR base stations, as shown in Fig. 4. All of the VR devices are able to connect and recognize a PC thanks to an application from Valve called Steam VR. The VR application is powered by the Unity engine and uses openVR API to develop. This application creates a virtual environment for the user to access through the VR headset. In the immersive environment, users are able to find an identical 3D model of the IRB1200. This virtual robot uses real-time feedback on joint positions collected from the actual robot to synchronize its motions with its physical counterpart. The VR controller, in particular, acts as the primary tool for user interaction and navigation within virtual reality. Furthermore, the right-side controller serves as an input device for robot control.

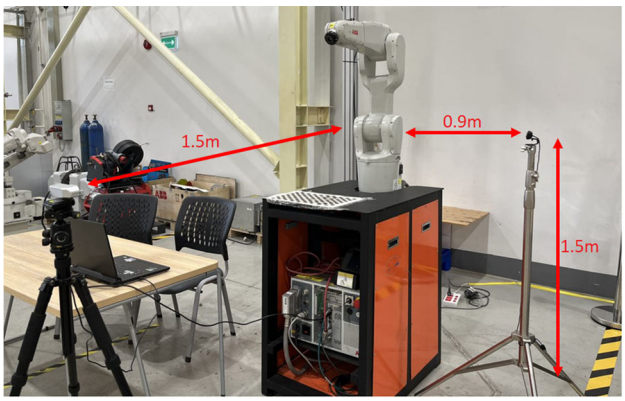


Fig. 3 Robot station

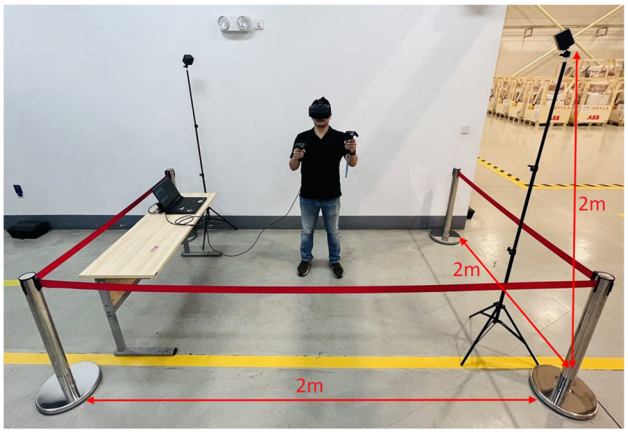


Fig. 4 VR station

This function is activated by pressing the trigger on the VR controller, which allows the VR application to capture the user's motions by extracting Cartesian position and Quaternion rotation data via the openVR API.

Hardware requirements vary significantly across the three primary system components. The MQTT broker, responsible for data reception and management, only requires minimal processing power. The device suitable for this task should have energy efficiency and network connectivity available. The robot station computer, acting as the interface between the VR station and the robot system, demands hardware compatibility with the specific robot manufacturer's specifications. While not requiring high-performance processing, it must fulfill the robot's communication and receive the control command from the user. Finally, the operator station, responsible for rendering the virtual environment, must meet the demanding hardware specifications of the chosen VR platform such as SteamVR. It is important to note that hardware requirements for all three components will scale dynamically with the system's complexity and the number of concurrent operations.

Hardware recommendations are presented as follows: The MQTT broker and the robot station computer should be equipped with a 2.0 GHz or faster processor with at least 2 cores, 8 GB of RAM, and 10 GB of available storage. The robot station computer must meet the operating system requirements specified by the robot manufacturer and another requirement if any. The operator station has the highest demanding hardware to support immersive environment rendering. Recommended specifications include an Intel Core i5-4590 or AMD Ryzen 1500 equivalent or greater CPU, an NVIDIA GeForce GTX 1060 or AMD Radeon RX 480 equivalent or greater GPU, and 8 GB or more of RAM.

3.2 Robot station

Among three utilities provided by the EGM feature, the system takes the most advantage of EGM position stream and EGM position guidance. It allows extracting the current mechanical property of the robot including the rotation angle of each joint and pose of the robot and, at the same time, writing position data as movement order for the robot. To use EGM, the robot controller communicates with the PC by a protocol named EMG sensor protocol, which is the combination of Google Protocol Buffer (Google Protobuf) as encoder and UDP as the data transmission protocol to prioritize data transmission time. In normal circumstances, the data package will be exchanged between the robot and PC every 4 ms with about 10–20 ms delay, guaranteeing smooth and stable motion for the robot. For the usage in this project, the feedback from the robot will contain joint data and the robot will be controlled using the pose data, including Cartesian coordinate and orientation in the form of Quaternion.

The data captured by the VR controller naturally matches the requirements of the EGM feature. However, it is essential to be aware that the coordinate systems used in both the VR station and the robot system are different. Therefore, an adjustment is required to avoid incorrect outcomes or, in extreme cases, potential hardware damage caused by commands that exceed the robot's operational limits. The suggested method involves calculating the displacement vector in the Unity coordinate system and then converting it into the robot's coordinate system. The destination will be calculated based on the displacement and the current position of the robot. From three points within the virtual environment, the user can define a robot coordinate with respect to the Unity coordinate. The first two points $P_1 = \begin{bmatrix} x_1 & y_1 & z_1 \end{bmatrix}$ and $P_2 = \begin{bmatrix} x_2 & y_2 & z_2 \end{bmatrix}$ are on the desire X-axis, while the third point $P_3 = \begin{bmatrix} x_3 & y_3 & z_3 \end{bmatrix}$ is located on the Y-axis. Each unit vector and the origin point is given by the equation:

$$\vec{i} = \frac{\vec{P_1P_2}}{\left|\vec{P_1P_2}\right|} \tag{1}$$

$$O = P_1 - \vec{i} \times \vec{P_1P_3}.\vec{i} \tag{2}$$

$$\vec{j} = \frac{\vec{OP_3}}{\left|\vec{OP_3}\right|} \tag{3}$$

$$\vec{k} = \vec{i} \times \vec{j} \tag{4}$$

Normally, these points are predefined so that the robot can do a mirror motion of user, but the user can have full access

to modify this aspect. When the user presses the trigger of the VR controller to indicate the first movement, the current pose of the controller will be saved as the relative origin $O_{re} = \begin{bmatrix} x_{re} \\ y_{re} \\ z_{re} \end{bmatrix}$ and latest robot pose saved as $O_s = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix}$. As the input point $P = \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$ is constantly capture, the calculated destination for the robot is given by

$$P_r = W^{-1} \times (P - O_{re}) + O_s \tag{5}$$

In which, rotation matrix W is define as

$$W = \begin{bmatrix} \vec{i} \\ \vec{j} \\ \vec{k} \end{bmatrix} \tag{6}$$

3.3 Operator station

In the virtual environment, a 3D virtual robot rendered by the Unity application precisely reconstructs the ABB IRB1200 model with an exact 1:1 ratio. To create the virtual robot, the model is first exported in OBJ file format from ABB Robot Studio to start the creation process. Then, it is imported into Blender, a powerful 3D creative tool, where modifications to the rig and minute features are carried out to resemble the actual robot closely. It is further integrated into Unity in the FBX file format after this Blender polishing stage. The structure of the 3D model object tree is altered within Unity to create a parent–child relationship, starting with joint 1 of the 3D robot model as the foundation parent and extending to joint 6. The pivot point is also modified to overlap with the contact point between its joints. The 3D model robot can accurately and precisely imitate the movements made by its real-world counterpart thanks to its hierarchical design.

The 3D model requires the rotational degrees of each joint from the actual robot to reproduce the movements of the real robot.

To execute the rotation movement for the virtual robot, the program uses Unity's *Quaternion.Euler* function to convert the joint angles from Euler representation to a quaternion, matching Unity's orientation format. It will rotate each joint a set angle value around the X, Y, and Z axes, defined with a pivot point. It is noted that the Unity coordinate system follows the left-hand rule with the Y-axis pointing upward, as demonstrated in the left side of Fig. 5. While the robot coordinates use a right-hand rule system with the Z-axis pointing upward, as shown on the right side of Fig. 5. As a result, the rotation angle around the X-axis needs to be flipped side before used.

3.4 Telecommunication

A significant part of the system is allowing users to operate remotely in realtime. This requires the communication channels to deliver data with low latency while maintaining data integrity. The system transfers two data types between the stations: robot control/feedback data and 2D video visual feedback, delivered using MQTT and WebRTC, respectively.

3.4.1 Data communication protocol

To transfer messages between the two stations, an MQTT broker must be established. The MQTT broker's primary function is to verify client connections, route, publish, and subscribe requests to the correct destinations, as well as store

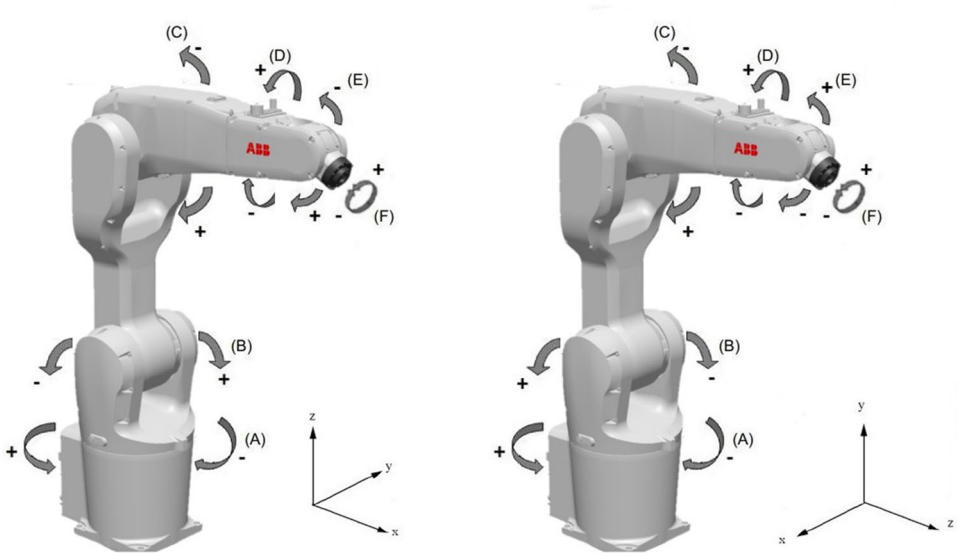


Fig. 5 Coordinate systems of robot and VR

retained messages. As mentioned earlier, MQTT brokers do not require significant computational resources [67, 68]. In this paper, we use the same broker for all testing configurations, regardless of their differences:

- CPU: Intel® Core™ i3-10110U
- RAM: 16GB
- Ethernet: Intel® Ethernet I219-V
- OS: Windows 10
- Software: Eclipse Mosquitto 2.0.15

To enable computers in the two stations to find the broker's IP address when connected to different networks, we use a Dynamic Domain Name System (DDNS) service in conjunction with Network Address Translation (NAT), which is applied to the MQTT application's IP address and port.

Security is a major concern for MQTT systems, given their ability to transfer high-frequency messages using a publish-subscribe architecture. Our system implements two layers of encryption: transport layer encryption and application layer encryption. To publish or subscribe to messages, clients must first send a connection request to the broker with their username and password. Clients are only allowed to send and receive data if the broker verifies these credentials. Before sending a message, both stations' computers encrypt the packet using Transport Layer Security version 1.2 (TLSv1.2). All data are sent in a JSON structure and published to different topics. Clients with the correct credentials can either publish or subscribe to these topics.

When sending or receiving data, users can choose from a few options. First, the retain option allows the last sent message to be stored on the broker since messages are generally not stored. Second, QoS is a set of technologies that ensure network data delivery [69]. MQTT has three QoS levels:

- QoS 0: Messages are sent from publishers to subscribers once (at most once).
- QoS 1: Messages are sent from publishers to subscribers. However, the process is only complete if the subscriber acknowledges to the publisher that the message has been received (at least once).
- QoS 2: There are two rounds of transmission. In the first round, the message is sent similarly to QoS 1. After the message has been delivered, the publisher sends a release message to the subscriber. If the subscriber has received the release message and sent back a complete message, the process is complete.

In our system, we use the retain option for fixed information such as scale and robot information. We use different QoS levels for different types of data. To optimize performance, we prioritize essential data by sending it with a higher QoS level, while the rest of the data is delivered with a lower QoS level.

3.4.2 Video stream protocol

Web Real-Time Communication (WebRTC) is an open-source application programming interface (API) that enables developers to design web-based applications for transmitting video, audio, messages, and files directly peer-to-peer with minimal latency [70]. WebRTC stands out from traditional video streaming protocols due to its simplicity, scalability, and security.

- Simplicity: WebRTC is simple to use because it is a web application that does not require any additional dependencies or plugins.
- Scalability: WebRTC is scalable because it uses peer-to-peer data transfer, which eliminates the need for a central server. This allows users and enterprises to scale their products more easily [71].
- Security: WebRTC uses Secure Real-Time Transport Protocol (SRTP) and Secure Real-Time Transport Control Protocol (SRTCP) to provide end-to-end encryption [72].

In our system, we use the Interactive Connectivity Establishment (ICE) framework to connect peers. ICE requires Session Traversal Utilities for NAT (STUN) and Network Address Translation (NAT) servers [73]. We use Agora's Software-Defined Real-Time Network (SD-RTN) service for this purpose. We also use Agora's RTC API to develop video streaming applications for Web and Unity platforms. We will host the web video streaming applications on Render.

For security, we have set up a REST website to generate authentication tokens. When a user logs in, they must enter basic parameters such as username, password, and channel ID. This data is used to generate a token on the REST website. The token is then verified with SD-RTN for authentication.

4 System evaluation

4.1 Study design

The experiment is designed to put the system under varying Internet conditions in order to examine its responses within each context. The robotic system will track the position of its end effector and compare it to user-input data. The precision of the robotic trajectory indicates the system's overall performance quality. A trajectory that closely resembles the input behavior is considered to indicate preferred system performance. A complete analysis of the system's performance is included in the experimental settings.

- **Ideal**: The system is implemented within the same local network.
- **Normal setup:** The robot station and operator station use two different networks.
- **Lost package:** Data packet transfer between two stations cannot be located or tracked within the expected or specified delivery or transmission route.
- **Throttle:** System experiences the deliberate regulation or restriction of data transfer rates or network usage by a network provider or administrator.

This study also examines the system's performance under various QoS configurations for the MQTT protocol. The primary goal is to identify the optimal QoS level that effectively mitigates the negative effects of variable Internet conditions.

4.2 Experiment setup

The experimental setup comprises two stations. The robotic station consists of an ABB IRB 1200 robot equipped with an IRC5 compact controller, a dedicated PC for communication, and two Logitech C920 Pro USB cameras. The operator station includes a PC and an HTC Vive VR headset. Both PCs within the experimental setup utilize the same Intel I219-V Ethernet controller as the MQTT broker and fulfilling the system's recommended specifications.

There are two steps to set up the robot station. Firstly, the rapid code needs to be loaded onto the robot controller. The process is simplified by using Robot Studio from ABB. At this stage, the PC is required to adjust its IP address in order to connect to the IRC5. An Ethernet cable is used to link the PC and the robot controller. Once this connection is made, the write function in Robot Studio may be used to load the rapid code into the robot's control unit.

After validating the robot's connection with the PC, the robot app is executed on the PC, keeping the devices connected as the PC SDK and EGM demand constant contact with the robot. Two cameras are attached to the system through USB cables to provide visual monitoring. When the system is operating, relevant information about the robot is displayed on the user interface (UI).

The experiment has two unique network setups. For testing the system under ideal situation and emulating Internet conditions, two stations and the broker are placed in the same area, sharing the same network, with the topology shown in Fig. 6. This setup helps improve consistency in the measurement. In conjunction with using a local network, the MQTT broker computer will also be installed with a network emulation software named SoftPerfect Connection Emulator. By using the emulation tool in the almost ideal setup, the experiment can eliminate other uncontrollable external impacts when transferring the data through the Internet.

When testing the system under normal usage, another network configuration is used. As the operator station and robot station are located at a different place, they are not sharing a network with either the other or the broker. Before establishing the connection, a DDNS service will be utilized for computers at VR and robot stations to find the broker through a domain name, as depicted in Fig. 7.

4.3 Procedure

The execution of the experiment involves an operator using a VR device to control the robot, following a predefined trajectory. The chosen maneuver to test the system performance is to follow a rectangle. The starting point is coincident with the robot's home position. The tracing track is located on an inclined surface to ensure that movement appears on all three axes.

The position of the robot's end effector is recorded in realtime. This requires modification in the software, as EGM does provide the feedback in the form of Cartesian position. Similarly, the input position or the VR controller position, in other words, is recorded using openVR API. The recorded data is saved in the CVS file. Once the operator presses the "connect" button to connect two stations, the data set will

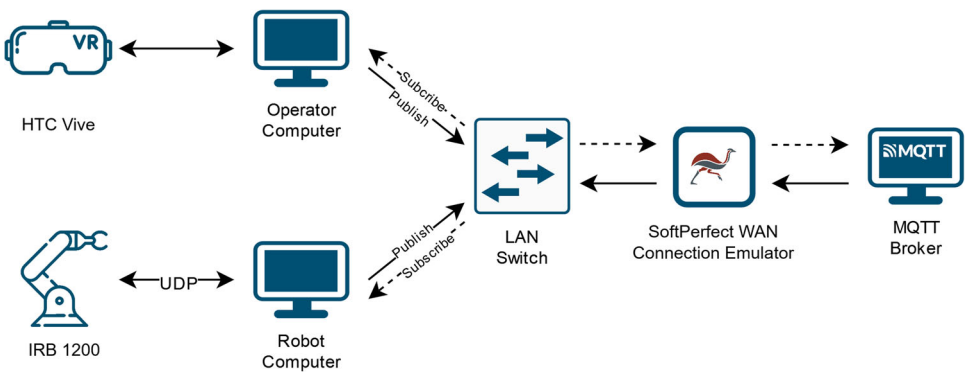


Fig. 6 Local network topology

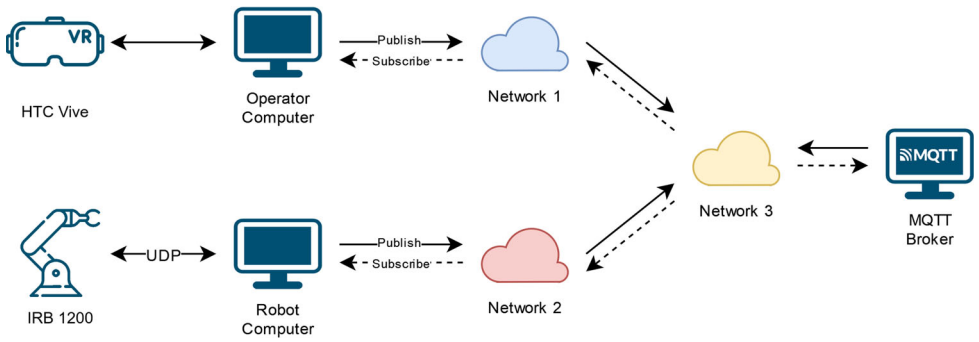


Fig. 7 Normal circumstanced network topology

indicate this moment for syncing the motion of the robot and user. The data set is sketched into a trajectory using MATLAB, and the performance is evaluated based on the root mean square error value between the desire and the measure position of the robot, calculated in all three different axes:

$$RMSE = \frac{1}{T}\sum_{T}^{k=1}(r_k - d_k)^2 \tag{7}$$

where r_k is the coordinate of the robot's recorded position and d_k is the desire position at the instant.

There is a problem that appears from the fact that the refresh rate of the VR application is 60 Hz, and robot feedback is provided with a frequency of 250 Hz, resulting in the difference between the data points of the two data sets. Therefore, a possible comparison will need a harmonization solution. The proposed method operates on the assumption that there is a linear trajectory between each recorded position of the VR controller. A coherence in the population of data points within the two datasets is obtained by extending numerous intermediate points along this linear path.

To observe the transmitted data, an application named Wireshark is used. By running Wireshark on the PC in both stations, all packages that come and go can be recorded and analyzed. By sorting them by the destiny IP and the destiny port, it is possible to identify and isolate the guidance position and the feedback information. Wireshark allows us to observe the number of packages sent per second as well as the throughput of each application.

5 Result and discussion

5.1 Local setup

The examination starts by measuring the system's baseline delay. This is accomplished by recording data from the VR interface and following feedback data from the robotic system. In this context, latency measurement begins at the precise moment when the user initiates an action, as indicated by the activation of the controller's trigger, and ends when the robot has traveled a minimum distance of 1 cm from its starting point, which is illustrated in Fig. 8. This 1-cm threshold is designed to mitigate any potential influence caused by motor vibrations or sensor errors inherent in the robotic system. The estimated latency is a combination of data transmission duration between stations, the temporal extent of robot movement, and the time allotted to communication exchanges between the robot and the central processing unit.

According to the examination as detailed in Fig. 9, QoS level 0 has the most significant latency, with an average delay of 139.3 ms. QoS levels 1 and 2 perform similarly, with QoS 1 slightly exceeding QoS 2, with corresponding latencies of 94.7 and 101.3 ms. The significant variance in latency found in QoS 0 is likely due to its inherent lack of message delivery certainty, resulting in a more significant delay.

Furthermore, a noticeable pattern appears from the recorded dataset, highlighting an inverse link between the degree of transfer assurance level and latency variance. QoS level 2, which denotes the highest guarantee level, has the least delay variability, with measured values ranging from 95 to 109 ms. In comparison, QoS level 1 exhibits latency variation ranging from 84 to 102 ms, while QoS level 0 exhibits the greatest significant variability, comprising data points ranging from 108 to 167 ms.

From this dataset, several discussions can be made. First of all, despite the use of a local network setup, the chance of data transfer failures still exists, increasing the overall system's latency. The imperatives of low latency and reliable data flow are critical in the context of robotic control operations. As a result, implementing QoS level 0 is not recommended due to its lack of message delivery certainty, a particularly significant disadvantage in scenarios requiring high-frequency data interchange.

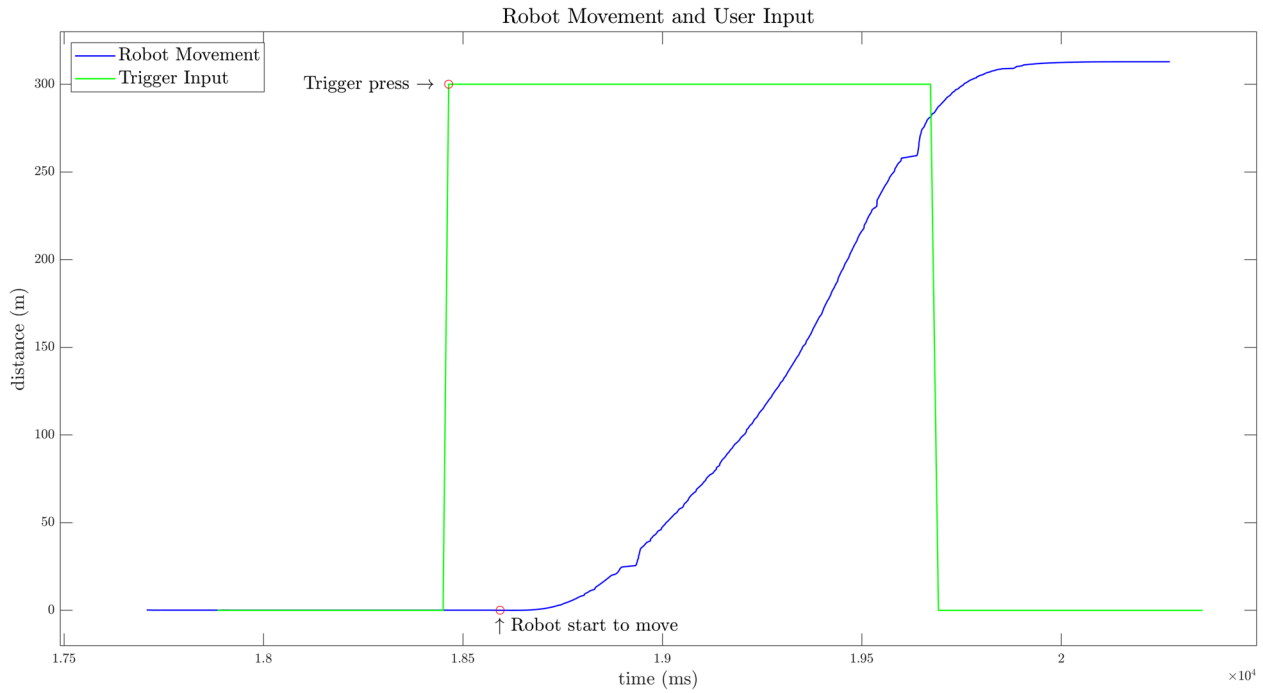
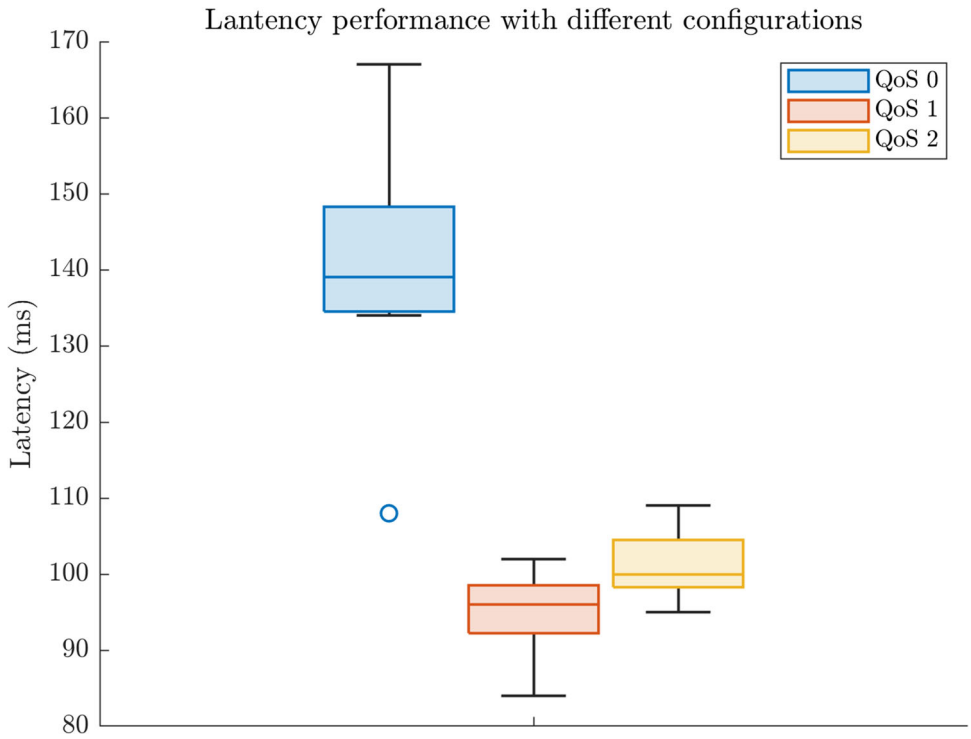


Fig. 8 User input (green) and robot movement (blue)

Fig. 9 Local Latency performance



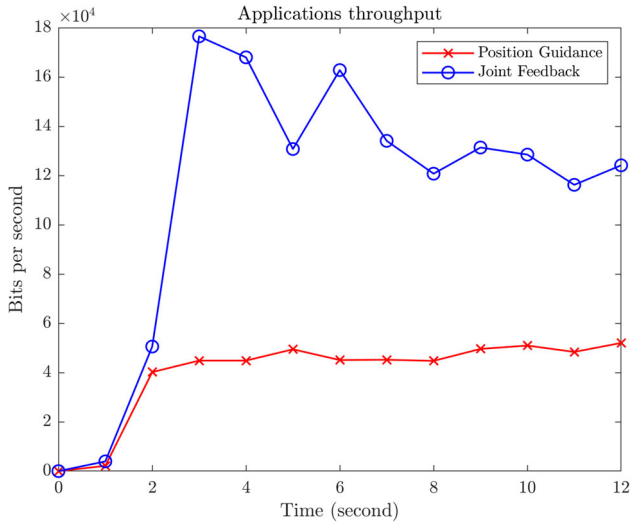


Fig. 10 System throughput

In comparison, QoS levels 1 and 2 have similar latency profiles and degrees of delay variation. Within the constraints of this limited network test environment, QoS level 1 is the practical alternative for ensuring the reliable transmission of messages between the two clients. While QoS level 2 provides more dependability via additional acknowledgment messages, the related improvement in latency uniformity does not justify the rise in data transfer volume.

The network activity of the computer is also monitored using Wireshark. The main focus is investigating the throughput of the two programs for VR and robots. Figure 10 delineates two distinctive data streams: the red line represents the positional guidance data sent by the VR application. In contrast, the blue line represents the received data by the robot program. The position guidance data sent from the operator station averages 46.8 thousand bits per second, while the average throughput of joint angle feedback from the robot station is 131.2 thousand bits per second. These results align with our messages sending frequency with position guidance data at 60 Hz, joints feedback at 250 Hz, and the data length of positional guidance being slightly longer.

It is also worth noting that as a TCP-based protocol, MQTT shares the same quality and tends to aggregate multiple messages into a single package. The statistics, as shown in Fig. 11, show that almost two-thirds of the joint feedback packets contain three unique datasets. In contrast, the vast majority of position guidance packets only carry a single dataset. This occurrence is likely due to the update frequency of each data type and network condition and will be explored further in the latter part of the test.

The robot's performance within the teleoperation system is highly dependent on the quality of the Internet connection. In order to establish a foundational benchmark for the experimental study, the system is configured under ideal conditions, employing a local network setup and utilizing QOS 1 configuration. Figure 12 shows the 3D trajectory of user input, the robot's end effector position, and the coordinates of each axis shown in Fig. 13.

Under these optimal circumstances, the robot consistently maintains an accuracy level within a range of 22 mm across all three axes. The specific result is shown in Table 1.

Even though the transmission of input data has no limitation, the presence of errors can be attributed to several factors, including hardware limitations within the robot, particularly concerning directional changes and motor speed

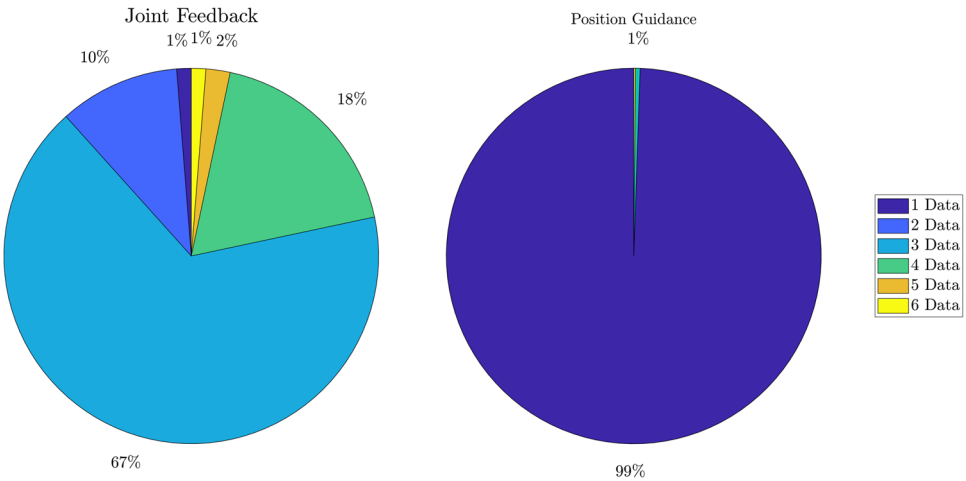


Fig. 11 Data included in a packet by percentage

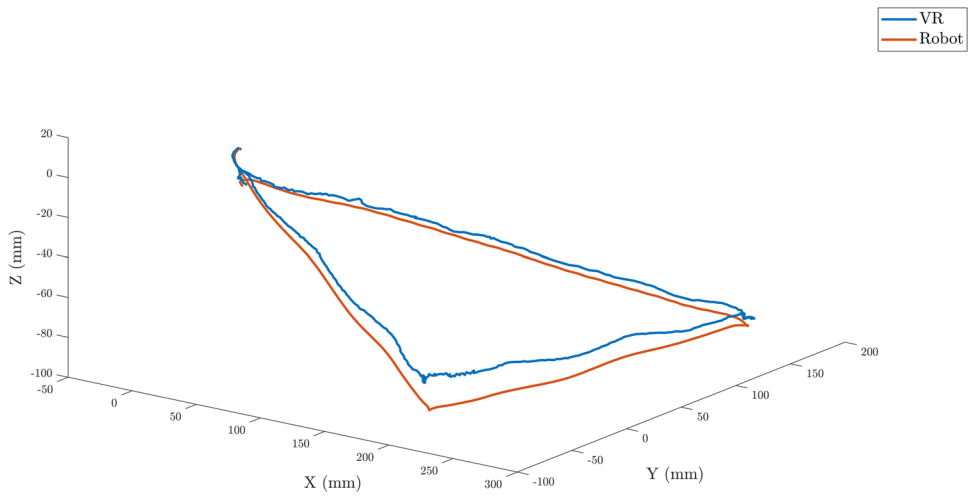


Fig. 12 Local network setup trajectory

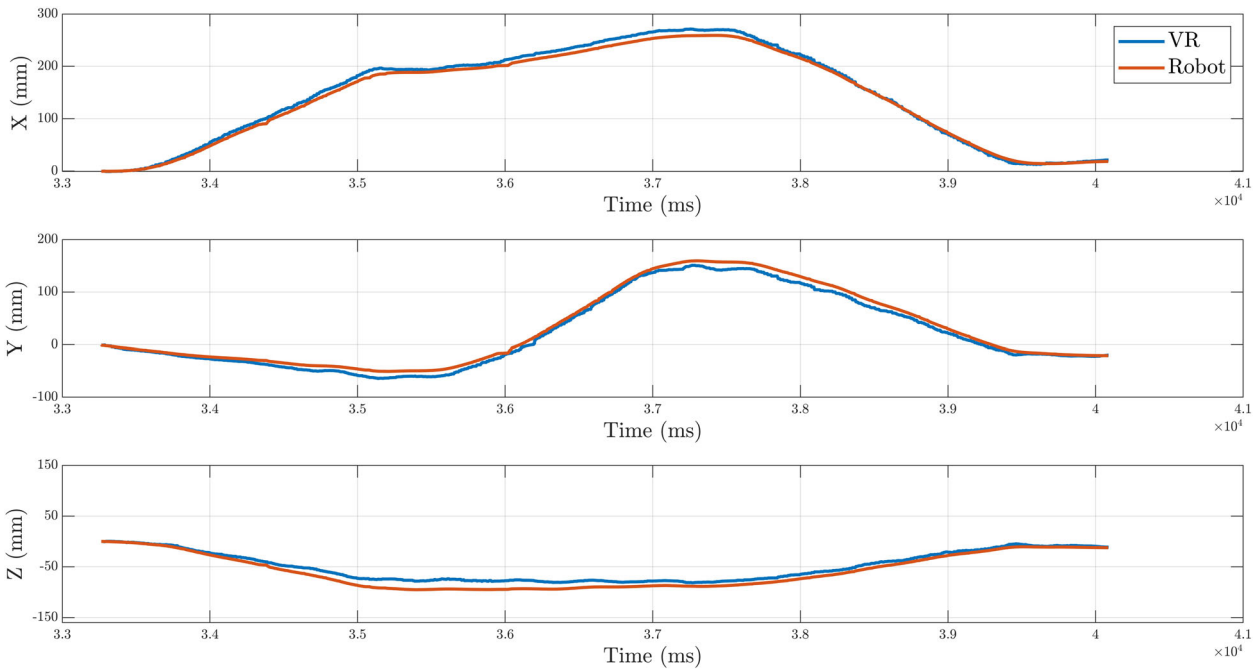


Fig. 13 Local network position graph

Table 1 Local network error value

Axis	Max	Mean
X	17.6269 mm	8.1095 mm
Y	21.5093 mm	9.5335 mm
Z	20.3307 mm	10.5197 mm

adjustments. These constraints become most apparent when the robot's movement direction changes. Additionally, the impact of user input noise is substantial, mainly when input data is acquired at high frequencies. A closer examination of the Z-axis reveals significant shakiness and instability. This is due to the experimental setup, in which the vertical axis provides limited support for the operator. As a result, the Z-axis has the highest average inaccuracy. Finally, differences in sample rates between the VR system and the robot's EGM function lead to the observed error rate inconsistencies.

5.2 Extreme condition performance

5.2.1 Package lost

The setup on Clumsy is 25% loss, and it shows the significant different on the system's performance.

The trajectory in Fig. 14 and the coordinate graph in Fig. 15 showed that the input movement is ignored by the robot in the first half of the experiment. This result can be verified when compared to the record of packages sent and received demonstrated in Fig. 16.

Upon establishing the initial connection between the two stations, the system experiences a substantial decline in the number of transmitted packages at both ends. Since the packages are not transferred, the input signal is lost during the transmission process, while the user continues to enter a new command position. Measurements show that there is a significant error in the robot's trajectory even after the initial package drop event, as it departs significantly from the planned location. Furthermore, when the system recovers, the robot continues to suffer from the impacts of data loss, as the number of received packages remains lower than the number of sent packages. In summary, the calculated result is shown in Table 2 below.

The experiment proceeds with an analysis aimed at establishing the best QoS configuration to reduce the likelihood of data loss. A separate system, consisting of two independent personal computers (PCs), is used to simplify the results and avoid unwanted damage to the robot hardware. One PC serves as the data publisher, while the other serves as the data receiver. These devices send about 10,000 positional data packets at a frequency of 10 Hz. SoftPerfect is still used to emulate varied degrees of data loss.

When a packet is lost, it must be retransmitted, which causes latency to increase. This pattern can be seen in all three test scenarios, referring to Fig. 17, where higher packet loss rates correspond to longer latency durations. Surprisingly, QoS 0 consistently has the lowest latency numbers across all three packet drop levels. Nonetheless, with a packet drop rate of 30%, QoS 0 saw a 2% loss in sent data. As the sample rate increases, the lost percent will increase significantly, resulting in the lousy robot performance shown above. QoS 1 and QoS 2, on the other hand, managed to maintain data integrity throughout the test, yet at the sacrifice of latency performance. QoS 1 performs similarly to QoS 0, while QoS 2 delays significantly more.

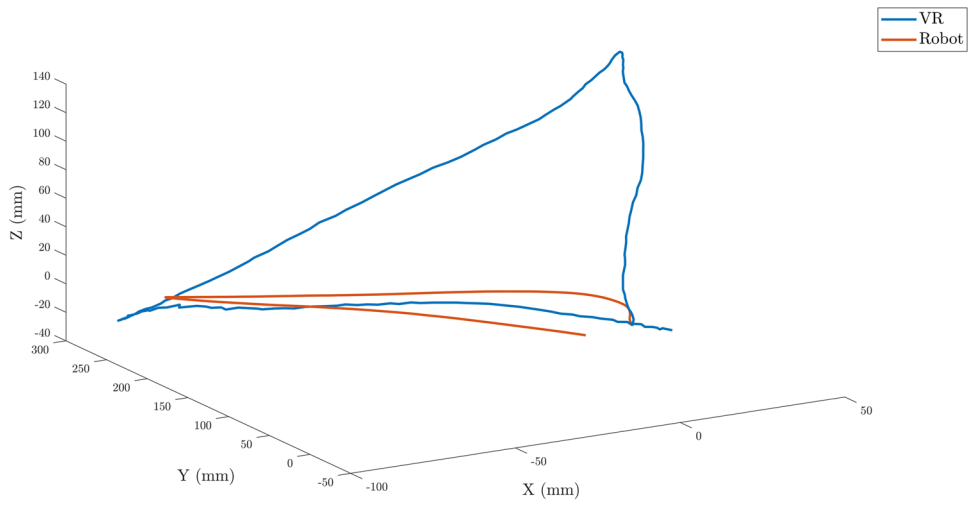


Fig. 14 Lost emulation trajectory

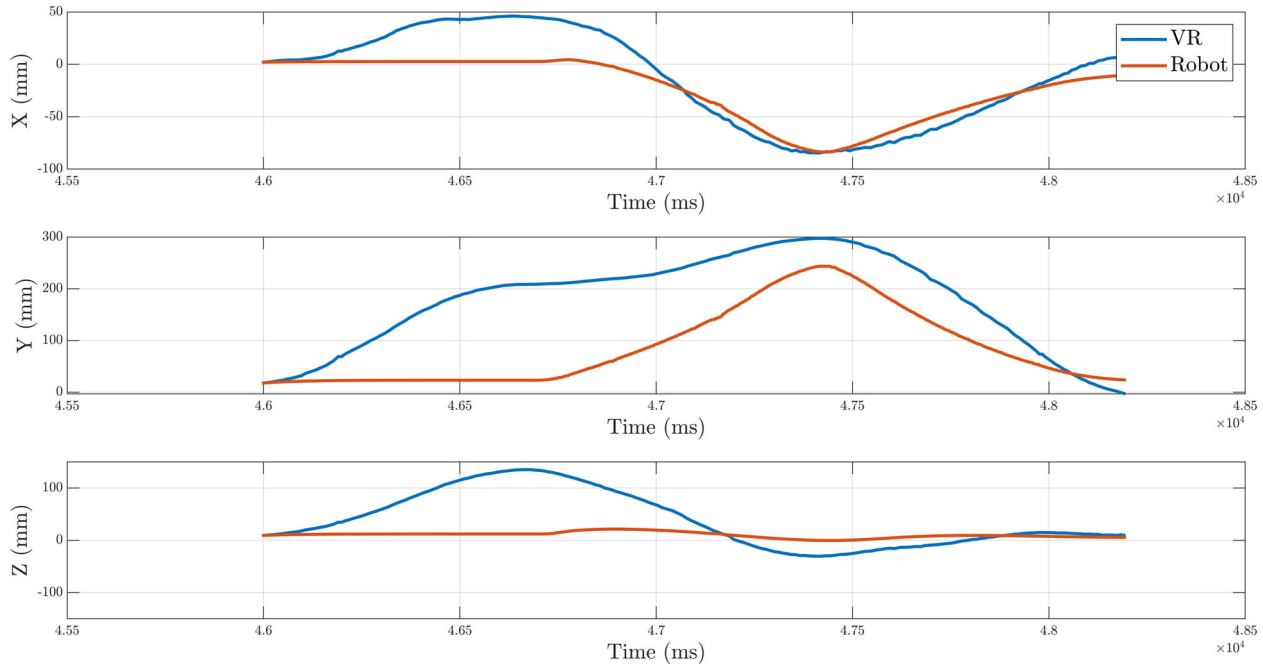


Fig. 15 Lost emulation position graph

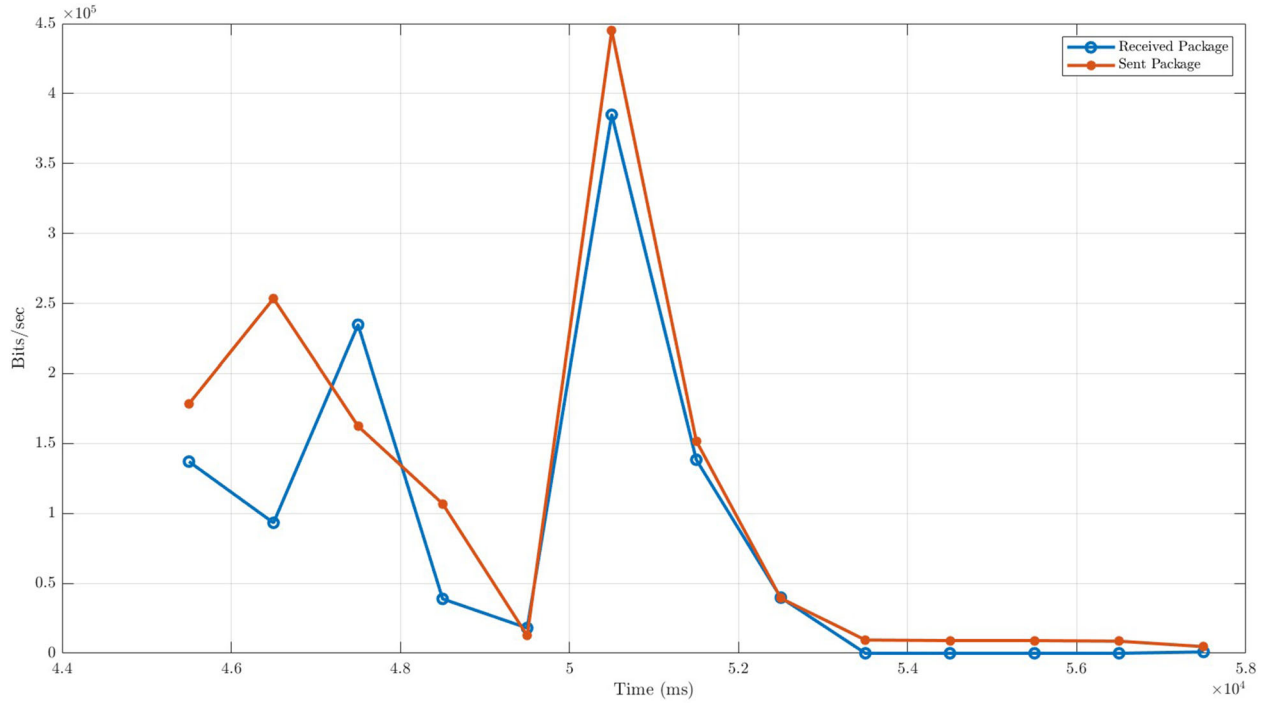


Fig. 16 Lost emulation transfer package record

Table 2 Package lost emulation error

	Max	Mean
X	2674.3 mm	2641.4 mm
Y	577.6322 mm	466.1216 mm
Z	1345.2 mm	1253.0 mm

5.2.2 Throttle

Besides loss, the throttle can also significantly affect the system. This situation happened as the Internet condition had a sudden quality drop, resulting in the data package arriving late and accumulating at the exact moment. The testing setup for the throttle happened within 500 ms maximum, and the occur change of 60%, result trajectory and the position graph shown in Figs. 18 and 19.

Despite a package loss rate of 3.56% shown by Wireshark analysis, the system behaves similarly to the package loss situation. This behavior can be related to the characteristics of the system and the TCP transmission protocol. When throttling happens, packages are compressed and transmitted at the same time, resulting in the observed behavior.

According to Fig. 20, the robot station does not receive most of the sent packages during the initial period. These undeliverable packages are only received once Internet conditions improve, resulting in a peak following the initial drought period. This behavior can ensure that the quantity of packages sent and received eventually equals. However, from the perspective of the robot station, several data streams cannot be processed simultaneously. Only the final data from a single transmission cycle is typically processed into the robot's movement orders. This explains why the robot skips the first half of the run, which is comparable to what happens when the package is lost. In summary, the maximum and mean error values are displayed in Table 3.

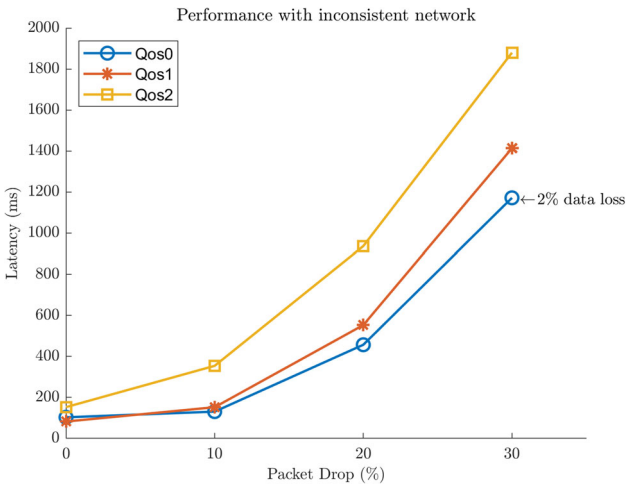


Fig. 17 Latency performance with different packet drop levels

5.3 Bandwidth

The bandwidth test results show promising prospects for project expansion. The system demonstrates its competence in managing the transfer of data from multiple robotic systems simultaneously. The results can be viewed from Fig. 21.

At 10 robots, all three configurations perform similarly to the latency test above, with the only exception being QoS 2, with the highest average latency at 158.8 ms. QoS 0 averages 153.7 ms with the highest variation in latency, ranging from 120 to 235 ms. QoS 1 had the best performance overall, with the lowest average latency of 94.5 ms and the lowest variation between 79 and 118 ms.

At 50 robots, all configurations showed performance degradation. QoS 2, with the highest data volume cost, had the most severe degradation, with its average latency jumping to 210 ms and its highest latency reaching nearly half a second. QoS 0 and 1 show slight degradation with an average latency of 171.5 and 101.4 ms, respectively.

At 100 robots, the system shows a significant performance drop, with all three configurations averaging more than half a second. QoS 1 still performs best, with an average latency of 481.9 ms. QoS 0 and 2 have an average latency of 1135 and 852.7 ms, respectively. Although QoS 1 and 2 have an average latency of less than 1 s, results are well above the 1-s threshold.

These results indicate that the limit for the number of robots that can be controlled simultaneously is 50 in the current testing setup. In contrast to the previous test, the extra acknowledgment messages of QoS 2 significantly impacted performance compared to QoS 1. QoS 0 performed comparably to QoS 2 due to its low data load. However, at 100 robots, QoS 0 showed the worst result, with its fourth quartile falling between the 2- and 5-s range. A pattern observed in the data at 100 robots is that the first two quartiles of all three configurations are comparable to the data at 50 robots. In contrast, quartiles 3 and 4 have much more significant variation. This can be attributed to the message queuing characteristics of the MQTT protocol. Depending on when the request is sent, the data may have to wait for the previous message buffering to be completed.

5.4 Remote setup

Lastly is the experiment result for the remote setup, which resembles closest to the normal usage of the system (Fig. 22).

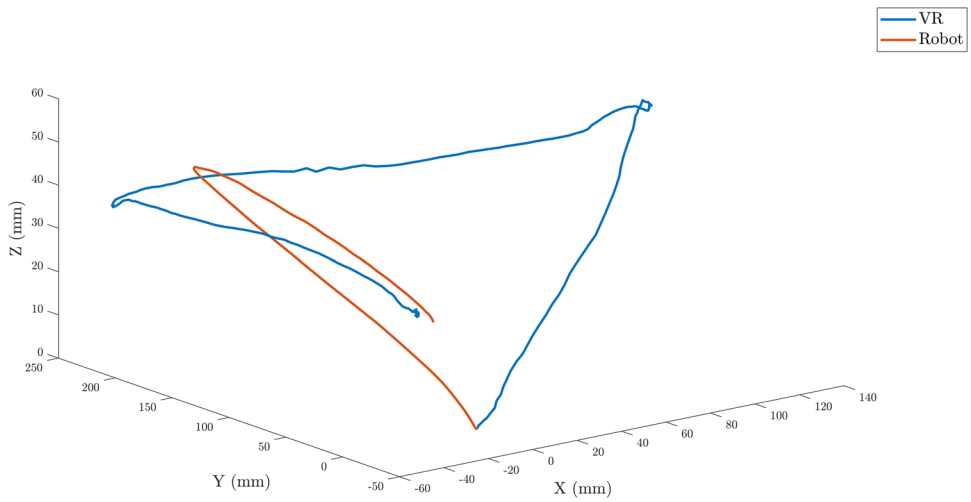


Fig. 18 Throttle emulation trajectory

All three configurations exhibited increased latency variation, but the trend of higher message delivery guarantee reducing latency variation remained true. QoS 2 had the lowest latency variation, with data ranging from 128 to 169 ms, despite a significant increase in average delay to 146 ms. QoS 1 experienced a slight increase in average latency to 99 ms with a data range of 65 to 138 ms with an outliner at 179 ms. QoS 0 had the highest latency variation, with data ranging from 103 to 280 ms and an average latency slightly above QoS 2 at 158 ms.

In the context of a more realistic scenario where the broker and both stations are situated in different locations and do not share the same network, the system experiences only marginal changes in its performance. In this configuration, besides the fact that the distance for data transferring is longer, the station is mandated to obtain the IP address of the broker through a DDNS service. Consequently, there is an increase in the delay from the moment when the user initiates a movement until the robot's initial motion. Despite this latency factor, the accuracy of the robot remains relatively

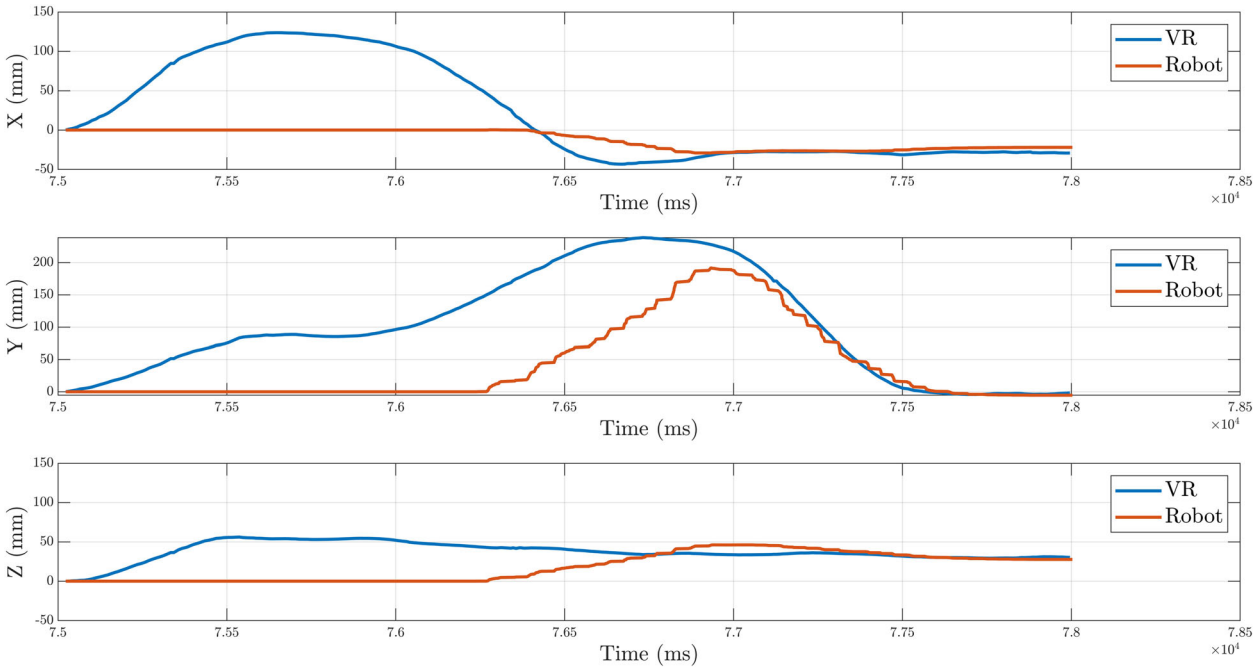


Fig. 19 Throttle emulation position graph

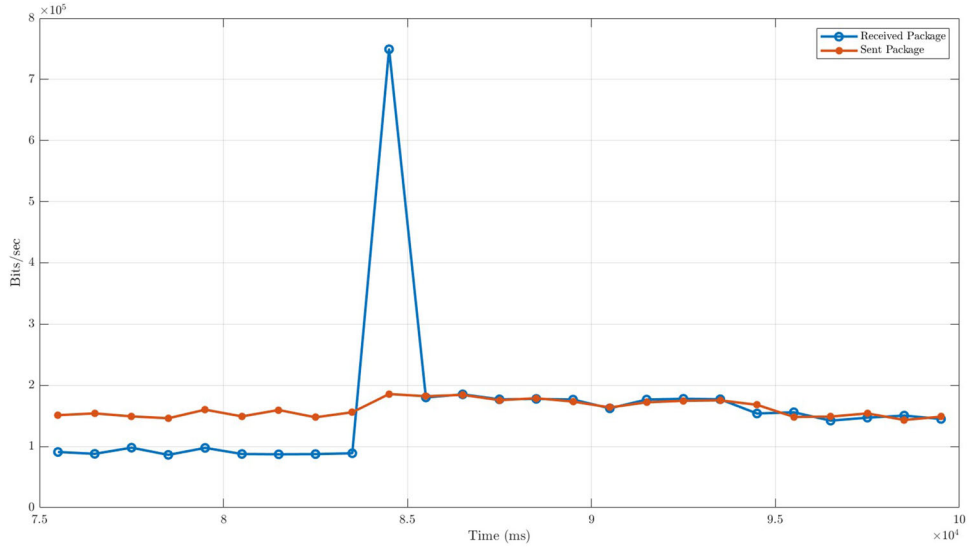


Fig. 20 Throttle emulation transfer package record

consistent with its performance under ideal conditions, as shown in Figs. 23 and 24.

In term of error from the robot, there is little to no difference. The value is shown in Table 4.

When comparing the mean values of the two setups, the inaccuracy increases by approximately 15.17%, 18.11%, and 8.47% for the X, Y, and Z axes, respectively. This rise is primarily due to the lag in robot movements. Even though the calculations sync two trajectories to match the first motions of both the user and the robot, this latency statistic remains unstable and subject to changes in bandwidth or an increase in the volume of data transfers. This impact is visible in Fig. 24, particularly on the X and Y axes, where changes in robot movement direction lag behind the VR representation by a few milliseconds. In addition, uncontrollable variables such as loss packets and throttle situations might contribute to the spread of errors. The system is more vulnerable to these two potential scenarios.

5.5 Discussion

The evaluation result showed a significant correlation between connection quality and robot accuracy. In a local setup, an observed discrepancy between the robot's movement and user input is less than 11 mm across all 3 axes. This deviation can originate from various sources, including the network jitter, control system, inaccuracies in robot position calculations, and inherent hardware limitations of the robot itself. Furthermore, even in a local setup, it still takes time to transmit the required input signals, which can also contribute to inaccuracies. The system in local setup has the lowest latency with the QoS level 1, QoS 2 has very similar performance, while QoS 0 has the highest latency, about 50 ms higher than others. QoS levels 1 and 2 also exhibited consistent performance with low variability, while the results obtained with QoS level 0 demonstrated a wide range of latency, highlighting a strong dependence on network performance.

Nevertheless, consistent network performance cannot be guaranteed due to various factors, ranging from external influences such as environmental interference and network

Table 3 Throttle emulation error value

	Max	Mean
X	2887.6 mm	2830.3 mm
Y	616.3547 mm	556.1694 mm
Z	1234.6 mm	1199.0 mm

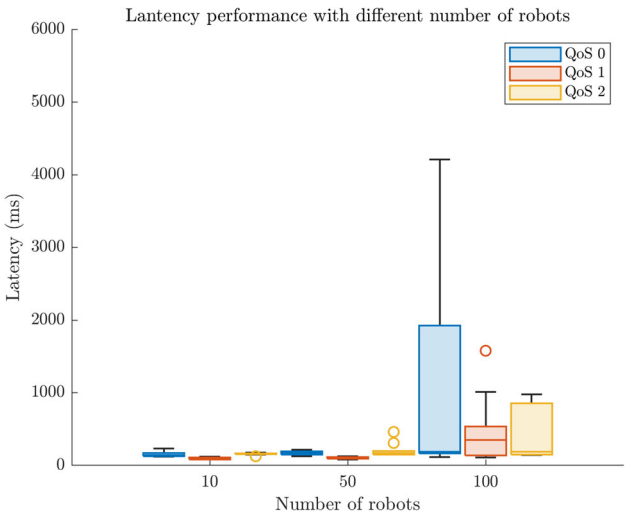


Fig. 21 Latency performance with different numbers of robot

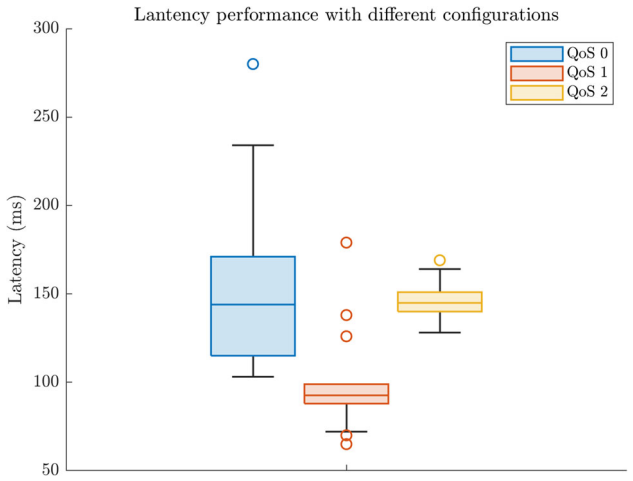


Fig. 22 Remote latency performance

congestion to internal factors like hardware limitations and insufficient bandwidth. In the event that package loss happens, the latency increases as the system uses higher QoS mechanisms. The disparity between QoS levels becomes more pronounced as the packet drop rate increases. However, while QoS levels 1 and 2 ensure the delivery of all user data to the robot station, there is a risk that user control signals fail to reach the robot station when operating at QoS level 0. In the occasion where the system experiences loss package, the robot's accuracy drops significantly. The average error ranges from 466 to 2641 mm, depending on the axis. This level of inaccuracy renders the robot incapable of executing any task effectively.

When evaluated in a distributed environment, with the robot station, user station, and broker located at distinct geographical locations, a noticeable increase in latency was observed. However, the relative performance of the different QoS levels remained consistent with the local setup. Specifically, QoS 0 exhibited the highest latency of 158 ms, while QoS 1 demonstrated marginally better performance compared to QoS 2, averaging about 99 ms and 146 ms, respectively. This increase in latency is primarily due to the limitations imposed by geographical distance. The fundamental constraint comes from the finite speed of light within fiber optic cables, establishing a lower bound for data transmission times. In industrial teleoperation, the observed latency can be considered within an acceptable range. While a definitive threshold for acceptable latency remains unestablished and dependent on the application requirements, past research suggests that for optimal performance in high precision applications, the required latency values should be below 300 ms [74–76].

Experimental results demonstrate that the current system architecture can facilitate the simultaneous control of up to 50 robots while maintaining latency within the recommended thresholds. To assess the scalability limits of the system, we refer to the optimal layout planning for human-robot collaborative assembly systems proposed by Eswaran et al. [34]. Their optimal layout comprises 15 robots, suggesting that the current system has the potential to accommodate up to three

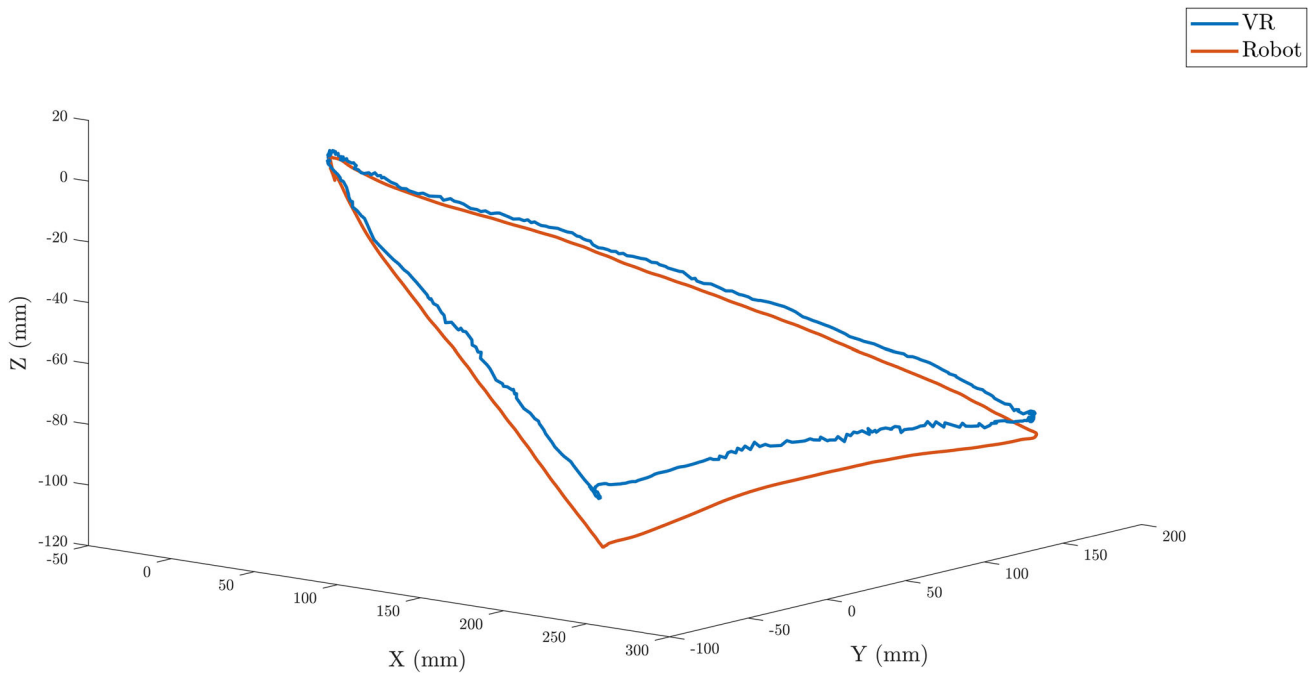


Fig. 23 Different network trajectory

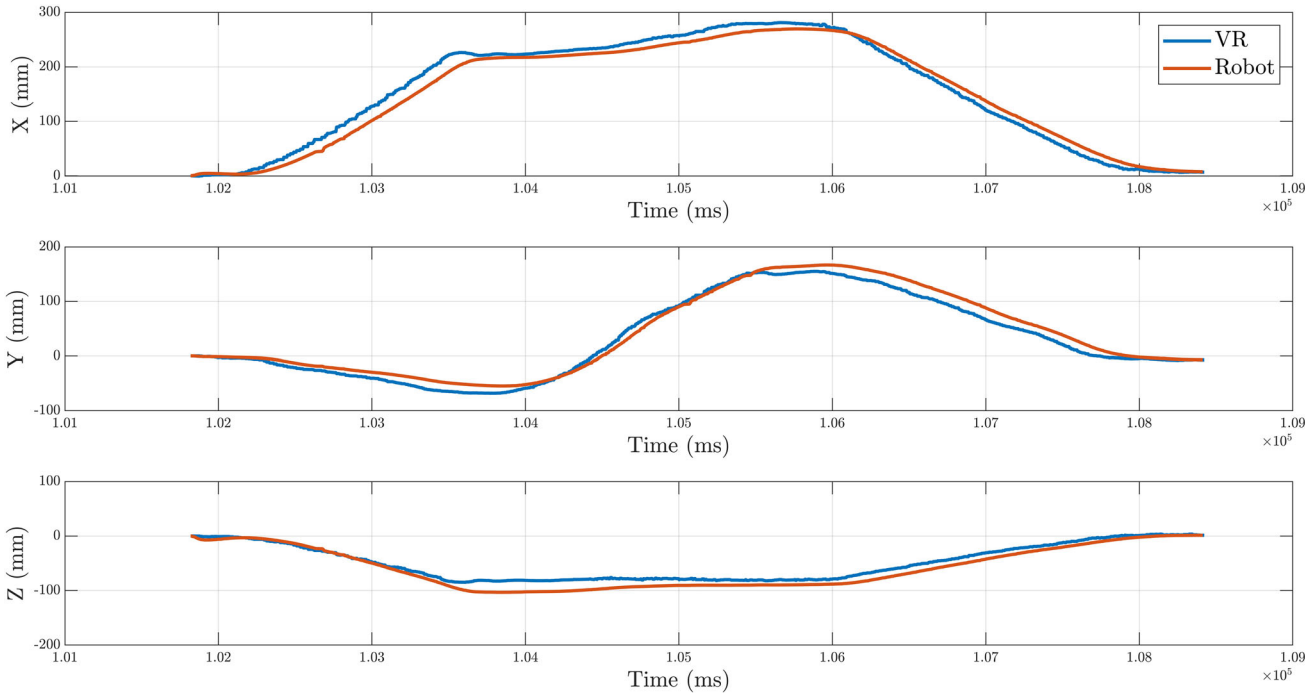


Fig. 24 Different network position graph

concurrent assembly lines. However, in real-world industrial settings, robots often operate autonomously based on pre-programmed routines, significantly reducing the demand for real-time control. These findings suggest that the system has the potential to be scaled effectively to meet the demands of different automation lines, demonstrating its versatility and adaptability for various industrial applications.

Despite the positive result, the integration of VR into robotic applications presents challenges related to the accuracy of user input. While VR controllers offer intuitive input mechanisms, their tracking, often reliant on lighthouse systems, inherently possesses limitations in precision. Moreover, human physiological tremor further diminishes input accuracy. To address this problem, data analysis can be initiated immediately upon packet reception and data ingestion. The primary function of this data analysis block is to scrutinize, categorize, and potentially adjust the incoming data, particularly if it has been affected by network transmission. A variety of methodologies can be employed in this stage, including the extended Kalman filter for noise reduction or predictive modeling incorporating robotic dynamics and real-time network conditions to refine input data and robot parameters. The integration of machine intelligence presents a promising avenue for more adaptable and robust data analysis solutions. To enhance robustness and safety, the robot's control system should incorporate modifications such as limiting motor speed and acceleration. These constraints serve to mitigate sudden jerks that may arise from network throttle or packet loss, thereby protecting the robot's hardware. To enhance the user experience and mitigate human error, target points for the robot can be defined and rendered within the virtual environment, providing users with a visual guide for the desired trajectory.

Table 4 Remote setup error value

Axis	Max	Mean
X	23.1827 mm	9.3405 mm
Y	27.3154 mm	11.2597 mm
Z	24.6945 mm	11.4107 mm

To optimize network performance for teleoperation, multiple techniques can be employed. First, it is recommended to set the QoS level to level 1 to ensure consistent data delivery between clients and servers. Prioritizing teleoperation traffic can further enhance performance by minimizing latency and packet loss. Secondly, information aggregation methods can be used to reduce the number of tiny packets sent [77]. By combining multiple small packets into larger messages, communication overhead is minimized, and network fault detection, correction, and overall efficiency are improved. Finally, network redundancy can be achieved by creating numerous communication channels between the client and server. This strategy improves fault tolerance while reducing the impact of single-point failures on the teleoperation system.

6 Conclusion

This paper introduces a teleoperation system designed for an industrial articulated robot. The system uses virtual reality technologies, including visual feedback from a virtual robot within an immersive environment and gesture-based control via a VR controller. This approach provides a more intuitive and natural control interface, with visual feedback improving responsiveness, precision, and reducing cognitive burden while improving depth perception for the user.

This research focuses on combining IoT techniques like the MQTT protocol and WebRTC to establish a consistent and smooth connection between the robot station and the user station. By using the Internet, this integration extends the system's reach, enabling users to control the robot remotely on a global scale without necessitating network sharing. The development enables the monitoring and troubleshooting of automation systems across geographical boundaries. Experiments that were conducted as part of this study confirm the MQTT protocol's applicability in real-time control applications. Evaluation of the system's operation under ideal and real-world conditions shows insignificant differences. Although the system may fail when encountering conditions such as packet loss and Internet throttling, using QoS level 1 is proven to be ideal, as it can prevent data loss while maintaining low latency. Finally, the bandwidth test showed the system's scalability potential. The suggested system, with the ability to communicate data for a maximum of 100 robots, enables users to monitor the entire automation line.

The presented framework can be applied for teleoperation application of industrial robotics manufacturing. The remote-control feature extends the capabilities of the robotic system into scenarios limited by human access, such as hazardous environments. This expands the operational range and potential applications of the system. The ability to operate multiple systems at the same time makes the system potential for monitoring and diagnosing robots in manufacturing line, further optimizing the production and asset management. Compared to traditional frameworks, the proposed solution has the potential to improve cost-efficiency, reduce energy consumption, and reduce maintenance requirements by reducing physical infrastructure and on-site activities of technicians and operators [78]. However, this statement will require further measurement and evaluation. Future studies should focus on optimizing the use of energy across the system. This involves reducing energy usage during immersive VR rendering and improving power management for the broker device.

In the near future, improvements in 5G and edge computing technologies will provide significant potential for improving system performance. The low latency and high bandwidth of 5G will allow for real-time, high-fidelity VR experiences while also improving the teleoperation system's responsiveness. Edge computing will enable more localized data processing and analysis, lowering latency and increasing overall system efficiency. These technological improvements will open the way for scalable, dependable, and high-performance VR-IoT teleoperation solutions, extending their utility to a wider variety of industrial applications such as remote maintenance, disaster response, and hazardous environment exploration.

However, ethical considerations must also be thoroughly investigated in future research. Increased dependence on teleoperation may have substantial societal effects, including job displacement and skill degradation in the workforce as the demand for on-site technician intervention declines. To reduce these risks, hybrid models that combine virtual and physical training programs have the potential to preserve important operational skills and adaptability. Furthermore, research should examine the social and psychological implications of long VR immersion for teleoperators to ensure their well-being and job satisfaction.

Acknowledgements This work acknowledges ABB Automation & Electrification Ltd. (Vietnam) – ABB Vietnam, Robotics & Discrete Automation Business – ABB Robotics Vietnam (ABB Robotics & Automation Solution Center – RASC, Ho Chi Minh City and ABB Robotics Technical and Service Center – RTSC, Bac Ninh) for non—financial, technical, and resource support and the valuable research strategy provided by RMIT University and Deakin University colleagues.
Individuals, the research team appreciates the invaluable support of Mr. Phu Huynh – Former Business Director, Mr. Tin Tran – Former Operations Manager & Customer Service, Mr. Hong Dang – Robotics Consultant & Sale Manager, Mr. Ninh Nguyen – Chief Engineer, Mr. Thai Truong – Technical Specialist, Mr. Minh Phung – Senior Technical Engineer, Mr. Man Dien – Robotics Engineer from ABB Robotics Vietnam and Mr. Anh Truong – RMIT Alumni for Technical Support.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Data Availability Not applicable.

Materials Availability Not applicable.

Code Availability Not applicable.

Declarations

Ethics Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication The authors give the consent for the publication in the International Journal of Advanced Manufacturing Technology.

Conflict of Interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the

source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Barbosa W, Gioia M, Natividade V, Wanderley R, Chaves M, Gouvea F, Gonçalves F (2020) Industry 4.0: examples of the use of the robotic arm for digital manufacturing processes. Int J Int Design Manuf (IJIDeM) 14:1569–1575
2. Jain R, Zafar M, Mohanta J (2019) Modeling and analysis of articulated robotic arm for material handling applications. IOP Conf Series: Mater Sci Eng 691:012010
3. Zhang B, Xie Y, Zhou J, Wang K, Zhang Z (2020) State-of-the-art robotic grippers, grasping and control strategies, as well as their applications in agricultural robots: a review. Comput Electron Agricul. 177:105694. https://www.sciencedirect.com/science/article/pii/S0168169920311030
4. Custodio L, Machado R (2020) Flexible automated warehouse: a literature review and an innovative framework. Int J Adv Manuf Technol 106:533–558
5. Billard A, Calinon S, Dillmann R, Schaal S (2008) Survey: robot programming by demonstration. (Springer)
6. Nagata F, Yoshitake S, Otsuka A, Watanabe K, Habib M (2013) Development of CAM system based on industrial robotic servo controller without using robot language. Robot Comput-Integrated Manuf. https://doi.org/10.1016/j.rcim.2012.09.015
7. Luo J, He W, Yang C (2020) Combined perception, control, and learning for teleoperation: key technologies, applications, and challenges. Cogn Comput Syst 2:33–43
8. Basañez L, Suárez R (2009) Teleoperation. Springer Handbook Of Automation. pp 449-468
9. Atre P, Bhagat S, Pooniwala N, Shah P (2018) Efficient and feasible gesture controlled robotic arm. In: 2018 Second international conference on intelligent computing and control systems (ICICCS), pp 1–6
10. Kim S, Hernandez I, Nussbaum M, Lim S (2024) Teleoperator-robot-human interaction in manufacturing: perspectives from industry, robot manufacturers, and researchers. IISE Transactions On Occupational Ergonomics And Human Factors 12:28–40
11. González C, Solanes J, Munoz A, Gracia L, Girbés-Juan V, Tornero J (2021) Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback. J Manuf Syst 59:283–298
12. Chandra Sekaran S, Yap H, Musa S, Liew K, Tan C, Aman A (2021) The implementation of virtual reality in digital factory–a comprehensive review. Int J Adv Manuf Technol 115:1349–1366
13. Franzluebbers A, Johnson K (2019) Remote robotic arm teleoperation through virtual reality. Symposium On Spatial User Interaction, pp 1–2
14. Whitney D, Rosen E, Phillips E, Konidaris G, Tellex S (2019)Comparing robot grasping teleoperation across desktop and virtual reality with ROS reality. Robotics Research: The 18th International Symposium ISRR, pp 335–350
15. Fang Y, Qiao Y, Zeng F, Zhang K, Zhao T (2022) A human-in-the-loop haptic interaction with subjective evaluation. Front Virtual Real 3:949324
16. Jiang H, Wachs J, Pendergast M, Duerstock B (2013) 3D joystick for robotic arm control by individuals with high level spinal cord injuries. 2013 IEEE 13th international conference on rehabilitation robotics (ICORR), pp 1–5
17. Jimeno A, Puerta A (2007) State of the art of the virtual reality applied to design and manufacturing processes. Int J Adv Manuf Technol 33:866–874
18. Su Y, Ahmadi M, Bartneck C, Steinicke F, Chen X (2019) Development of an optical tracking based teleoperation system with virtual reality. In: 2019 14th IEEE conference on industrial electronics and applications (ICIEA), pp 1606–1611
19. Wang Z, Cai Z, Cui L, Pang C (2018) Structure design and analysis of kinematics of an upper-limbed rehabilitation robot. MATEC Web Of Conf 232:02033
20. Fang Y, Liu Q, Xu Y, Guo Y, Zhao T (2023) Virtual reality interaction based on visual attention and kinesthetic information. Virtual Reality, pp 1–11
21. De Pace F, Gorjup G, Bai H, Sanna A, Liarokapis M, Billinghurst M (2021) Leveraging enhanced virtual reality methods and environments for efficient, intuitive, and immersive teleoperation of robots. In: 2021 IEEE international conference on robotics and automation (ICRA), pp 12967–12973
22. Rakita D, Mutlu B, Gleicher M (2018) An autonomous dynamic camera method for effective remote teleoperation. In: Proceedings Of The 2018 ACM/IEEE international conference on human-robot interaction, pp 325–333
23. Ha C, Park S, Her J, Jang I, Lee Y, Cho G, Son H, Lee D (2015) Whole-body multi-modal semi-autonomous teleoperation of mobile manipulator systems. In: 2015 IEEE international conference on robotics and automation (ICRA), pp 164–170
24. Zhou T, Zhu Q, Du J (2020) Intuitive robot teleoperation for civil engineering operations with virtual reality and deep learning scene reconstruction. Adv Eng Inf 46:101170. https://www.sciencedirect.com/science/article/pii/S1474034620301415
25. Rosli A, Mohamad R, Yusof Y, Shahbudin S, Rahman F (2020) Implementation of MQTT and LoRaWAN system for real-time environmental monitoring application. In: 2020 IEEE 10th symposium on computer applications & industrial electronics (ISCAIE), pp 287–291
26. Sacoto Cabrera E, Palaguachi S, León-Paredes G, Gallegos-Segovia P, Bravo-Quezada O (2020) Industrial communication based on MQTT and Modbus communication applied in a meteorological network. Int Conf Adv Emerg Trends Technol pp 29-41
27. Mukherji S, Sinha R, Basak S, Kar S (2019) Smart agriculture using internet of things and MQTT protocol. 2019 International conference on machine learning, big data, cloud and parallel computing (COMITCon) pp 14–16
28. Yalcinkaya F, AYDİLEK H, Erten M, İNANÇ N (2020) IoT based smart home testbed using MQTT communication protocol. Int J Eng Res Dev 12:317-324
29. Gemirter C, Şenturca Ç, & Baydere Ş (2021) A comparative evaluation of AMQP, MQTT and HTTP protocols using real-time public smart city data. 2021 6th International conference on computer science and engineering (UBMK). pp 542–547
30. Tran T, Nguyen Q, Luu T, Tran M, Kua J, Hoang T, Dien M (2025) Empowering robotic training with kinesthetic learning and digital twins in human–centric industrial systems. J Ind Inform Int 43:100743
31. Tran T, Dien M, Trinh D, Tran M, Hoang T (2023) Application of mixed reality in industrial and collaborative robot control. 2023 International conference on control, robotics and informatics (ICCRI). pp 24–29
32. Eswaran M, Raju Bahubalendruni M (2003) Augmented reality aided object mapping for worker assistance/training in an industrial assembly context: exploration of affordance with existing guidance

techniques. Comput Ind Eng 185:109663 (2023,11). https://doi.org/10.1016/j.cie.2023.109663
33. Eswaran M, Bahubalendruni M (2024) AR/VR assisted integrated framework of autonomous disassembly system for industrial products. Comput Ind Eng 196:110522
34. Eswaran M, Inkulu A, Tamilarasan K, Bahubalendruni M, Jaideep R, Faris M, Jacob N (2024) Optimal layout planning for human robot collaborative assembly systems and visualization through immersive technologies. Exp Syst Appl 241:122465
35. ME, Prasad V, Hymavathi M, Bahubalendruni M (2024) Augmented reality guided autonomous assembly system: a novel framework for assembly sequence input validations and creation of virtual content for AR instructions development. J Manuf Syst 72:104-121. https://www.sciencedirect.com/science/article/pii/S0278612523002297
36. Eswaran M, Gulivindala A, Inkulu A, Raju Bahubalendruni M (2023) Augmented reality-based guidance in product assembly and maintenance/repair perspective: a state of the art review on challenges and opportunities. Exp Syst Appl. 213:118983. https://www.sciencedirect.com/science/article/pii/S0957417422020012
37. Su Y, Chen X, Zhou T, Pretty C, Chase G (2022) Mixed reality-integrated 3D/2D vision mapping for intuitive teleoperation of mobile manipulator. Rob. Comput-Integr Manuf 77:102332. https://www.sciencedirect.com/science/article/pii/S0736584522000217
38. Wang X, Besançon L, Ammi M, Isenberg T (2022) Understanding differences between combinations of 2D and 3D input and output devices for 3D data visualization. Int J Human-Comput Stud 163:102820
39. Kaczmarek W, Panasiuk J, Borys S, Banach P (2020) Industrial robot control by means of gestures and voice commands in off-line and on-line mode. Sensors 20:6358
40. Bordoni S, Tang G (2023) Development and assessment of a contactless 3D joystick approach to industrial manipulator gesture control. Int J Ind Ergon 93:103376
41. Conn M, Sharma S (2016) Immersive telerobotics using the oculus rift and the 5DT ultra data glove. 2016 International conference on collaboration technologies and systems (CTS), pp 387–391
42. Heidari O, Perez-Gracia A (2019) Virtual reality synthesis of robotic systems for human upper-limb and hand tasks. 2019 IEEE Conference On virtual reality and 3d user interfaces (VR), pp 966–967
43. González C, Solanes J, Munoz A, Gracia L, Girbés-Juan V, Tornero J (2021) Advanced teleoperation and control system for industrial robots based on augmented virtuality and haptic feedback. J Manuf Syst 59:283–298
44. Chen J, Moemeni A, Caleb-Solly P (2023) Comparing a graphical user interface, hand gestures and controller in virtual reality for robot teleoperation. Companion Of The 2023 ACM/IEEE international conference on human-robot interaction, pp 644–648
45. Naceri A, Mazzanti D, Bimbo J, Tefera Y, Prattichizzo D, Caldwell D, Mattos L, Deshpande N (2021) The Vicarios virtual reality interface for remote robotic teleoperation. J Int Rob Syst 101 (2021,4). https://doi.org/10.1007/s10846-021-01311-7
46. Touhid M, Marne M, Oskroba T, Mirahmadi S, Zhu E, Mehrabian A, Defersha F, Yang S (2023) Building a cloud-based digital twin for remote monitoring and control of a robotic assembly system. Int J Adv Manuf Technol (2023,11). http://dx.doi.org/10.1007/s00170-023-12611-7
47. Su Y, Lloyd L, Chen X, Chase J (2023) Latency mitigation using applied HMMs for mixed reality-enhanced intuitive teleoperation in intelligent robotic welding. Int J Adv Manuf Technol 126:2233–2248
48. Sun Y, Jeelani I, Gheisari M (2023) Safe human-robot collaboration in construction: a conceptual perspective. J Safety Res 86 pp 39–51. https://www.sciencedirect.com/science/article/pii/S0022437523000804
49. Böhlen C, Brinkmann A, Mävers S, Hellmers S, Hein A (2020) Virtual reality integrated multi-depth-camera-system for real-time telepresence and telemanipulation in caregiving. 2020 IEEE International conference on artificial intelligence and virtual reality (AIVR) pp 294–297
50. Codd-Downey R, Forooshani P, Speers A, Wang H, Jenkin M (2014) From ROS to unity: leveraging robot and virtual environment middleware for immersive teleoperation. 2014 IEEE International conference on information and automation (icia). pp. 932-936
51. Pace F, Gorjup G, Bai H, Sanna A, Liarokapis M, Billinghurst M (2021) Leveraging enhanced virtual reality methods and environments for efficient, intuitive, and immersive teleoperation of robots. 2021 IEEE International conference on robotics and automation (ICRA), pp 12967–12973
52. Fan W, Guo X, Feng E, Lin J, Wang Y, Liang J, Garrad M, Rossiter J, Zhang Z, Lepora N (2023) Others Digital twin-driven mixed reality framework for immersive teleoperation with haptic rendering. IEEE Rob Autom Lett
53. Black D, Nogami M, Salcudean S (2024) Mixed reality human teleoperation with device-agnostic remote ultrasound: communication and user interaction. Comput Graph 118:184–193
54. Borges E, Rieder J, Aschenbrenner D, Scharff R (2022) Framework for armature-based 3d shape reconstruction of sensorized soft robots in extended reality. Frontiers In Robotics And AI. 9:810328
55. Mishra B, Kertesz A (2020) The use of MQTT in M2M and IoT systems: a survey. IEEE Access. 8:201071–201086
56. Bellavista P, Zanni A (2016) Towards better scalability for IoT-cloud interactions via combined exploitation of MQTT and CoAP. 2016 IEEE 2nd International forum on research and technologies for society and industry leveraging a better tomorrow (RTSI), pp 1–6
57. Shahri E, Pedreiras P, Almeida L (2024) A scalable real-time SDN-based MQTT framework for industrial applications. IEEE Open J Ind Electron Soc
58. Mishra B, Mishra B, Kertesz A (2021) Stress-testing MQTT brokers: a comparative analysis of performance measurements. Energies 14:5817
59. Naik N (2017) Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. 2017 IEEE International systems engineering symposium (ISSE), pp 1–7
60. Gruener S, Koziolek H, Rückert J (2021) Towards resilient IoT messaging: an experience report analyzing MQTT brokers. 2021 IEEE 18th International conference on software architecture (ICSA), pp 69–79
61. Luzuriaga J, Perez M, Boronat P, Cano J, Calafate C, Manzoni P (2015) A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. 2015 12th Annual IEEE consumer communications and networking conference (CCNC), pp 931–936
62. Hillar G (2017) MQTT essentials: a lightweight IOT protocol: the preferred IOT publish-subscribe lightweight messaging protocol. (Packt Publishing)
63. Caiza G, Garcia C, Naranjo J, Garcia M (2020) Flexible robotic teleoperation architecture for intelligent oil fields. Heliyon 6:e03833 (2020,4). https://doi.org/10.1016/j.heliyon.2020.e03833
64. Garcia C, Montalvo-Lopez W, Garcia M (2020) Human-robot collaboration based on cyber-physical production system and MQTT. Procedia Manuf 42:315–321
65. Brunnström K, Dima E, Qureshi T, Johanson M, Andersson M, Sjöström M (2020) Latency impact on quality of experience in a virtual reality simulator for remote control of machines. Signal Process Image Commun 89:116005

66. Van Damme S, Sameri J, Schwarzmann S, Wei Q, Trivisonno R, De Turck F, Torres Vega M (2024) Impact of latency on QoE, performance, and collaboration in interactive multi-user virtual reality. Appl Sci 14:2290
67. Upadhyay Y, Borole A, Dileepan D (2016) MQTT based secured home automation system. *2016 symposium on colossal data analysis and networking (CDAN)*, pp 1–4
68. Eridani D, Widianto E (2018) Performance of sensors monitoring system using raspberry Pi through MQTT protocol. 2018 International seminar on research of information technology and intelligent systems (ISRITI), pp 587–590
69. Toldinas J, Lozinskis B, Baranauskas E, Dobrovolskis A (2019) mqtt quality of service versus energy consumption. 2019 23rd International conference electronics, pp 1–4
70. Zinnah Apu K, Mahmud N, Hasan F, Sagar S (2017) P2P video conferencing system based on WebRTC. 2017 International conference on electrical, computer and communication engineering (ECCE), pp 557–561
71. Papalini M, Carofiglio G, Compagno A, Mantellini A, Muscariello L, Samain J, Sardara M (2020) On the scalability of WebRTC with information-centric networking. 2020 IEEE International symposium on local and metropolitan area networks (LANMAN, pp 1–6
72. Barnes R, Thomson M (2014) Browser-to-browser security assurances for WebRTC. IEEE Int Comput 18:11–17
73. Sredojev B, Samardzija D, Posarac D (2015) WebRTC technology overview and signaling solution design and implementation. 2015 38th International convention on information and communication technology, electronics and microelectronics (MIPRO), pp 1006–1009
74. Wang Y, Ai Q, Shi T, Gao Y, Jiang B, Zhao W, Jiang C, Liu G, Zhang L, Li H, Gao F, Ma X, Li H, Zhang X (2024) Influence of network latency and bandwidth on robot-assisted laparoscopic telesurgery: a pre-clinical experiment. Chin Med J (2024,8). http://dx.doi.org/10.1097/CM9.0000000000003257
75. Black D, Andjelic D, Salcudean S (2024) Evaluation of communication and human response latency for (human) teleoperation. IEEE Trans Med Robot Bionics
76. Noguera Cundar A, Fotouhi R, Ochitwa Z, Obaid H (2023) Quantifying the effects of network latency for a teleoperated robot. Sensors 23:8438
77. Xiang L, Luo J, Vasilakos A (2011) Compressed data aggregation for energy efficient wireless sensor networks. 2011 8th Annual IEEE communications society conference on sensor, mesh and Ad Hoc communications and networks, pp 46–54
78. Eswaran M, Bahubalendruni M (2022) Challenges and opportunities on AR/VR technologies for manufacturing systems in the context of Industry 4.0: a state of the art review. J Manuf Syst 65 pp 260–278

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.