A Proposed Immersive Digital Twin Architecture for Automated Guided Vehicles Integrating Virtual Reality and Gesture Control

Mokhtar Ba Wahal[1], Maram Selsabila Mahmoudi[1], Ahmed Bahssain[1], Ikrame Rekabi[1], Abdelhalim Saeed[1], Mohamad Alzarif[1], Mohamed Ellethy[2], Neven Elsayed[3], Mohamed Abdelsalam[2] and Tamer ElBatt[1,4]

[1]*Department of Computer Science and Engineering, The American University in Cairo, Cairo, Egypt*

[2]*Siemens EDA, Cairo, Egypt*

[3]*Know-Center, Graz, Austria*

[4]*Department of Electronics and Communications Engineering, Faculty of Engineering, Cairo University, Giza, Egypt*

{*mokhtarsalem, marammahmoudi, ahmed.bahusain, ikrame.rekabi, abdelhalim_saeed, mohamadalzarif, tamer.elbatt*}*@aucegypt.edu,* {*mohamed.abd_el_salam_ahmed, mohamed.ellethy*}*@siemens.com, nelsayed@know-center.at*

Keywords: Human-Robot Interaction, Virtual Reality, Modeling, Digital Twin, Immersive Systems, Cyber-Physical Systems.

Abstract: Digital Twins (DTs) are virtual replicas of physical assets, facilitating a better understanding of complex Cyber-Physical Systems (CPSs) through bidirectional communication. As CPS grows in complexity, the need for enhanced visualization and interaction becomes essential. This paper presents a framework for integrating virtual reality (VR) with a Dockerized private cloud to minimize communication latency between digital and physical assets, improving real-time communication. The integration, based on the Robot Operating System (ROS), leverages its modularity and extensive libraries to streamline robotic control and system scalability. Key innovations include a proximity heat map surrounding the digital asset for enhanced situational awareness and VR-based hand gesture control for intuitive interaction. The framework was tested using TurtleBot3 and a 5-degree-of-freedom robotic arm, with user studies comparing these techniques to traditional web-based control methods. Our results demonstrate the efficacy of the proposed VR and private cloud integration, providing a promising approach to advance Human-Robot Interaction (HRI).

1 INTRODUCTION AND RELATED WORK

Designing a digital representation of a physical asset has always been considered a method to study, analyze, and control physical assets effectively. Dr. Michael Grieves first introduced the concept of Digital Twins (DTs) in 2002 as real-time duplicates of physical assets with seamless data exchange for several purposes, including monitoring, comprehension, and optimization (Alexandru et al., 2022). However, the concept of Digital Twins gained more attention in 2010 when NASA saw its potential uses. The interest of NASA in the concept of Digital Twins helped shift the focus to advancing the technology, even if it did not successfully implement the concept of Digital Twins at the time (Saddik, 2018). DTs are considered important strategic technologies because of their effectiveness in time, cost reduction, and precise simulation of operational settings by using the digital asset as a virtual representation for designing, testing, and operations (Attaran and Celik, 2023) (Crespi et al., 2023).

The implemented applications of Digital Twins are mostly focused on manufacturing compared to the number of Digital Twins in critical mission applications such as robotics operating in rescue missions (Pirker et al., 2022). The success of critical missions operated by robotics is dependent on the situational awareness of the robot and the environment (Riley and Endsley, 2004). Most of the rescue Digital Twins found in the literature do not include an Autonomous Guided Vehicle (AGV), a mobile robot designed for autonomous navigation, limiting their effectiveness in dynamic environments. The traditional methods for actuating robotic arms using web-based dashboards or keyboard buttons have proven to be limited in terms of flexibility and ease of use, particularly for users un-

Wahal, M. B., Mahmoudi, M. S., Bahssain, A., Rekabi, I., Saeed, A., Alzarif, M., Ellethy, M., Elsayed, N., Abdelsalam, M. and ElBatt, T.
A Proposed Immersive Digital Twin Architecture for Automated Guided Vehicles Integrating Virtual Reality and Gesture Control.
DOI: 10.5220/0013176900003912
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025) - Volume 1: GRAPP, HUCAPP and IVAPP*, pages 267-275
ISBN: 978-989-758-728-3; ISSN: 2184-4321
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

familiar with robotic systems (Abdulla et al., 2020).

In recent years, there has been increasing interest in the potential of Human-Robot Interaction (HRI) techniques for enhancing Digital Twins applications, driven by the need for more intuitive and effective methods to improve the situational awareness of operators (Gallala et al., 2022). One effective technology for enhancing HRI is Virtual Reality (VR), which enables users to engage and interact within a synthesized three-dimensional environment (Gigante, 1993). VR directly enhances HRI by offering a more immersive user experience, interactive simulations, and intuitive actuation methods. All these features together result in a more effective and efficient environment for monitoring and controlling DT-aided telerobotics, specifically AGV with an integrated robotic arm. In their survey, (Mazumder et al., 2023) highlighted the effectiveness of VR data processing for enhancing robot control and assisting human operators. They discussed the development of a telerobotic workspace with Digital Twin integration for gesture-based robot control. Additionally, they noted that VR/MR (Mixed Reality) interfaces significantly improved DT-aided telerobotics by providing more immersive controls, particularly useful for operations in hazardous environments. Both assets must be synchronized to ensure the correct bi-directional data flow. Ellethy et al. (Ellethy et al., 2023) proposed a novel architecture that utilizes a Dockerized private cloud to minimize the latency between the two edges of the system. A Dockerized private cloud is a set of Docker containers hosting services such as storage, AI, and real-time monitoring (Ellethy et al., 2023). A comparative analysis of public vs. private cloud-based systems using latency and computation as metrics showed the comparative advantage of using a Dockerized private cloud to host the services provided by the Digital Twin.

In this work, we propose a novel architecture that increases the HRI and immersive experience of Digital Twins practitioners through the integration of VR with a Dockerized private cloud ensuring minimized latency that allows the user to handle critical missions. We used an open-source Software Development Kit (SDK) to construct the Digital Twin, in addition to extending it to integrate the virtual reality part of the system.

This paper consists of the following sections. Section 2 provides an overview of the system by highlighting the tools and components used to implement the project. Afterward, a detailed description of the system architecture is presented in Section 3, with technical details about integrating a Dockerized Private Cloud and Virtual Reality. In Section 4, methods to improve HRI using VR technology are introduced, including a Heat Map to enhance situational awareness of the user and gesture-based actuation methods for more immersive control of the AGV and the robotic arm. In addition, this section discusses a user study that evaluates the proposed actuation methods. Finally, Section 5 presents conclusions about the system and suggests potential directions for future research.

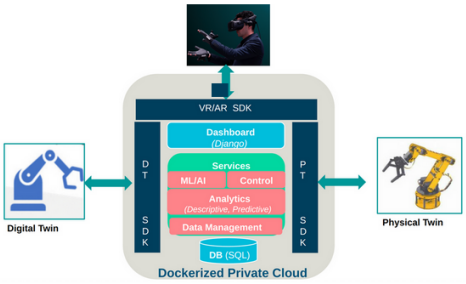


Figure 1: A diagram showcasing the proposed Digital Twin architecture, making use of a Dockerized private cloud.

2 SYSTEM OUTLINE

This section gives a background about the tools used in the project, as familiarity with the following concepts and tools is essential to the full understanding of the details of the project.

The TurtleBot3 is a small, programmable mobile robot that has two models: “Waffle” and “Burger.” We opted for the waffle model in our setup for its compatibility with the arm manipulator and its suitability for research and educational purposes. In general, the TurtleBot3 was selected for its open-source architecture, modularity, and extensive community support, making it ideal for developing complex robotic applications (Amsters and Slaets, 2020).

The TurtleBot3 is equipped with a Light Detection and Ranging (LIDAR) sensor, mounted on top of the robot, that can accurately measure the distance to the surrounding objects by emitting laser pulses and analyzing their reflections. It can navigate autonomously using Simultaneous Localization and Mapping (SLAM), a mapping method that allows robots and other autonomous vehicles to build a map and localize themselves within it simultaneously. A G-mapping package that provides laser-based SLAM as a ROS node called slam_gmapping was used to map our environment.

In addition to the LIDAR, other key components of the TurtleBot3 include the Raspberry Pi (RPi) and the OpenCR board. The Raspberry Pi, placed one level below the LIDAR, serves as the main computer,

providing processing power to run the robot's software, manage communication, and handle data from sensors. While the OpenCR board, placed below the RPi, is the robot's microcontroller, it interfaces with the hardware components like motors and sensors and executes low-level control tasks for precise and real-time operation. Finally, on the lowest level, we find the battery and two Dynamixel XL430 W250 servo motors to help stabilize the robot's speed and position (Amsters and Slaets, 2020).

Beyond the TurtleBot3, we integrated a five-degrees-of-freedom robotic arm, which is digitally represented by the OpenMANIPULATOR-X simulation, providing a comprehensive Digital Twin for testing and control.

The TurtleBot3 operates on a Linux-based operating system, specifically Ubuntu 20.04, installed on the RPi SD card. To program and control the TurtleBot3 and its robotic arm, we utilized ROS, a flexible and powerful framework for robotics software development. ROS provides features that support a microservices architecture, emphasizing decentralization, fault tolerance, and decoupling of system components through the use of ROS nodes and topics. These nodes can run across multiple devices and connect via various mechanisms, such as requestable services or publisher/subscriber communication. We selected ROS as the primary framework due to its widespread adoption in the robotics community, extensive library support, and facilitation of modular development, which accelerates implementation and fosters collaboration (Estefo et al., 2019).

The Dockerized private cloud architecture is a layered architecture originally proposed by Ellethy et al.(Ellethy et al., 2023). The architecture offers significant advantages regarding latency reduction, scalability, and maintenance. It is composed of three main layers:

The Hardware-Specific Layer. It includes both digital and physical assets. Each asset is managed by a ROS Master and an identical ROS subsystems. The separation of ROS Masters is to avoid single points of failure and prevent bottlenecks. At the same time, the identical subsystems ensure modularity and seamless replication. Individual ROS nodes handle different functionalities, such as collecting sensory data, processing actuation commands, and video streaming.

The Middleware Layer. It facilitates the communication between the first and third layers. It marshals data between the two layers, ensuring that each ROS node from Layer 1 has a corresponding middleware node for data translation. Communication between the hardware-specific layer and the Dockerized private cloud layer is achieved through web sockets, making this layer hardware-agnostic and scalable.

Dockerized Private Cloud Layer. As illustrated in the grey box of Figure 1. This layer provides a set of Docker containers that offer services such as AI-based processing, real-time monitoring, and data storage. By leveraging Docker Compose, multiple containers are orchestrated to run services such as database management and server operations. This setup ensures that the digital twin is always available with minimal latency and improved security as compared to architectures relying on public or remote clouds.

As shown in Figure 1, each of the subsystems –Digital Twin, Physical Twin, and Virtual Reality– comes with its designated SDK. These SDKs streamline the process of reproducing the system or creating other Digital Twins for customized use. Instead of building everything from scratch, users can rely on the SDKs to generate the essential scripts for all subsystems. They can then edit and customize these scripts as needed to suit their specific requirements.

The hand gesture control interface was implemented using an Oculus Quest 2 and its enabling SDKs. Oculus Quest 2 provides an affordable entry point into Virtual Reality and offers users an immersive VR experience. To enable the user to control both the AGV and the robotic arm in a fully immersive environment, we utilized the XR Controller package in Unity. This package integrates the Input System and translates tracked input from controllers into XR interaction states. Specifically, we used Unity's OpenXR XRSDK framework, which allows for robust interaction with VR hardware.

3 VIRTUAL REALITY INTEGRATION WITH A DOCKERIZED PRIVATE CLOUD

The system presents a comprehensive framework for integrating Virtual Reality with a Dockerized private cloud infrastructure to benefit from the minimized communication latency between both the physical and digital assets and the VR headset. Our work builds upon the findings of a previous work in which Ellethy et al. (Ellethy et al., 2023) highlighted the benefits of using a private cloud over a public one. A comparative study was conducted to evaluate the latency and accuracy of a Digital Twin using different cloud systems. The results showed that a local cloud was 5.5 times faster than a public cloud when transmitting an average data size of 128 bytes. In a local network

environment, such as a Wireless Local Area Network (WLAN), data does not need to travel long distances or pass through multiple servers, as it does in public cloud infrastructures like AWS. This significantly reduces network latency between physical and digital assets, resulting in faster response time and more accurate real-time control, which are critical for AGV operations and enhancing the immersion of VR environments through a synchronized DT.

The integration of VR with a Dockerized private cloud plays a significant role in enhancing user interaction with the digital asset. VR provides immersive interaction for the user by simulating its presence in this digital environment, which is an ideal tool for interfacing with DTs. The choice of Unity as the holder of the digital asset instead of Gazebo was driven by several factors. Primarily, Gazebo plugins for deploying the system to VR headsets have not been updated to keep pace with the rapid development the industry has seen in recent years. Additionally, Unity provides greater flexibility for creating custom environments and integrating advanced graphic features. Most notably, Unity excels in supporting innovative interactive input interfaces, such as hand gestures. In general, the ROS-Unity3D architecture for the simulation of mobile ground robots proved to be a viable alternative for ROS-Gazebo and the best option to integrate VR into the system (Platt and Ricks, 2022). Due to the mentioned reasons, the common practice among robotics developers and researchers is to divide the system into two subsystems: Ubuntu for interaction with the physical asset and Windows for the VR development in Unity.

The system's latest update involves creating a Unity scene to host the digital simulation.

Map: The environment was first constructed on an Ubuntu machine using the slam_gmapping node. The mesh files were then transferred to the Unity project on a Windows machine, where surfaces and colliders were added to the walls to enhance realism.

URDF Importer: The Unified Robotics Description Format (URDF) Importer is a Unity robotics package that enables the import of robot assets defined in the URDF format into Unity scenes. The imported asset retains its geometry, visual meshes, kinematic, and dynamic attributes. The Importer leverages PhysX 4.0 articulation bodies to parse the URDF file. Using the URDF Importer, the TurtleBot3 OpenMANIPULATOR-X was imported, and additional objects were added to replace some of the default actuation scripts, as advised by the package development team. Customized scripts were implemented to better represent the AGV and its robotic arm within Unity.

3.1 Robotic Arm Integration

The TurtleBot3 was augmented with a five-degree-of-freedom robotic arm. The arm is powered by the TurtleBot3 OpenCR board while its servo motors are connected to a servo motor controller board for easier setup. The latter board is connected to the RPi and communicates with it using the I2C protocol, a simple serial protocol used for communication between two devices or chips in an embedded system, to receive commands and control the arm.

To digitally represent the robotic arm, we utilized the OpenMANIPULATOR-X model, which is mounted on the TurtleBot3 and imported into Unity. The arm follows the same three-layer architecture as the TurtleBot3, with two separate ROS Masters: one for the Physical Twin and the other for the Digital Twin. The hardware layer of the Digital Twin is responsible for collecting the sensory data from the arm, specifically the joint_state data. This data is organized and sent to the middleware layer, which propagates it to the cloud layer via web sockets. Once the data reaches the cloud, it is processed and sent back through the same layers in reverse, eventually reaching the hardware layer of the physical asset to actuate the robotic arm.

The robotic arm utilized underwent meticulous assembly via piece-by-piece integration, following the manufacturer's instructions. During the assembly sequence, linkages were established between the components. Each connection necessitated proper execution to ensure the arm's proper functioning. Given the study's specific demands, minor modifications were implemented on the arm. Despite its six servo motors, only five were ultimately employed in the experiment to match the Digital Twin in the degrees of freedom. Of the accessories initially included, only a select few were deemed suitable for the task; the arm grabber required replacement. This modification resulted from uncertainties regarding its precision in the execution of object manipulation tasks. The robotic arm was mounted on an additional surface placed on top of the TurtleBot3 platform to integrate the robotic arm into the experimental setup. This position was chosen to prevent interference with the LIDAR sensor and, at the same time, to keep the balance of the TurtleBot3.

The robotic arm consists of five servos, each offering a range of 180 degrees. Each servo motor was strategically placed to enable precise control over the arm's movements. Operating particularly, a merger of such functions:

- Servo Motor 1 facilitates the rotation of the arm from side to side, serving as the base of its movement.

- Servo Motor 2 controls the forward and backward motion of the arm, allowing it to extend and retract as needed.
- Servo Motor 3 is responsible for the vertical movement of the upper half of the arm, enabling it to adjust its height.
- Servo Motor 4 controls the opening and closing of the gripper mechanism, enabling the arm to grasp and release objects.
- Servo Motor 5 provides control over the gripper itself, allowing precise manipulation during object handling tasks.

Overall, the robotic arm was integrated into the experimental setup with careful consideration of its design and functionality. By selecting appropriate components and making necessary modifications, the arm was tailored to meet specific needs while ensuring reliable performance and accurate data collection.

3.2 Unity-ROS Messages and Communications

Integrating VR with ROS necessitated incorporating Unity into the project due to its robust support for 3D simulation and VR technologies (Allspaw et al., 2023). Unity-ROS communication was established using a ROS-TCP connector, which involved initiating a TCP endpoint on the ROS Master of the Digital Twin. Within Unity, publishers and subscribers for various topics were implemented to facilitate communication with the ROS network. Through the server endpoint, serialized ROS messages—such as distance measurements from the distance-sensor topic, camera feed frames, Twist messages, and joint-state messages—are exchanged between Unity and the ROS Master of the Digital Twin.

As illustrated in Figure 2, the data flow between Unity and ROS comprises multiple critical stages. Unity serves as a simulation environment, generating messages that are published to topics created by the Digital ROS Master. These messages are subsequently transmitted to the physical ROS Master via the Dockerized private cloud using web sockets. This setup ensures reliable and low-latency communication between the digital simulation and the physical robot. The figure illustrates how serialized ROS messages are propagated from Unity to the physical asset, with each message type—such as sensor readings or actuation commands—managed through designated topics. This architecture is essential for maintaining synchronization between virtual and physical systems, ensuring accurate and real-time operation of robotic assets.

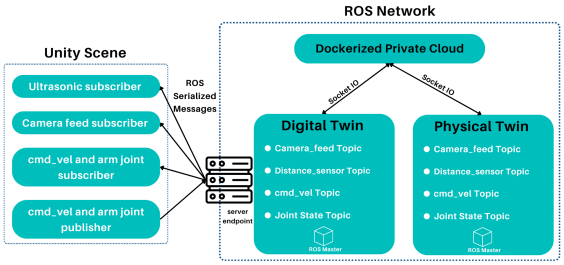


Figure 2: Data flow between ROS and Unity.

4 METHODS TO IMPROVE HUMAN-ROBOT INTERACTION USING VR

4.1 Heat Map Implementation

To enhance the user's situational awareness, we augmented the AGV with several ultrasonic sensors to implement a Heat Map. This map is integrated on a quad with a shader and is applied only to the surrounding environment of the TurtleBot3, improving its ability to detect and react to nearby objects. Ultrasonic sensors are strategically positioned around the TurtleBot3 to ensure full coverage, allowing for obstacle detection from all directions. The Heat Map is dynamically adjusted by the shader to the red color, with the intensity of the color depending solely on the distance from the TurtleBot3 to the surrounding object. The distance is determined by the data received from the ultrasonic sensor. The closer the object, the more intense the red color is. A threshold is set to determine the change of the color intensity. For example, a distance between 1 and 20 cm indicates that the object is very close, causing the red color to be very intense. A distance between 20 and 30 cm makes it less intense, and so on. If the distance is above 50 cm, no Heat Map is shown, indicating that the TurtleBot3 is navigating in a safe environment.

In addition to providing a visual representation, the ultrasonic sensors trigger real-time responses from the AGV based on proximity to obstacles. If an object is detected within the 1-20 cm range, the AGV automatically adjusts its movement to avoid collisions by slowing down or stopping. This reactive behavior not only provides a visual representation of the presence and proximity of obstacles but also ensures that the AGV can operate safely in a dynamic environment by actively avoiding obstacles. Figure 3 shows an image of how the Heat Map looks when an obstacle is within the 1-20 cm range from an obstacle.

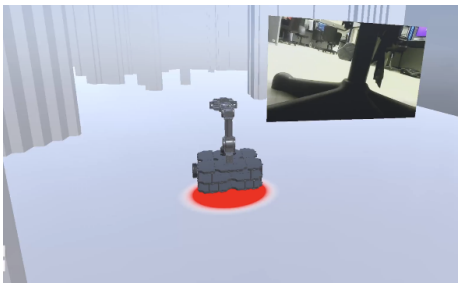


Figure 3: The heat map representation.

4.2 Arm Actuation Methods

To address the limitations of traditional robotic arm actuation methods, we introduce three actuation methods based on the user's hand gestures and movements. These methods aim to provide the user controlling the AGV and the robotic arm with a more immersive experience.

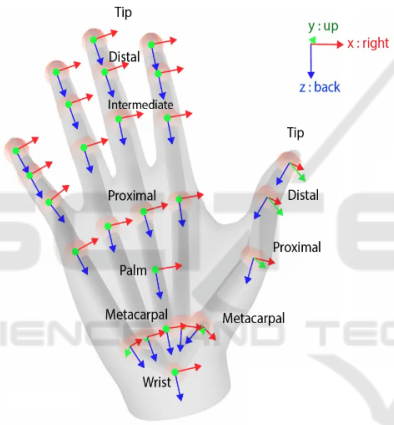


Figure 4: XR-Hand Joints.

Hand tracking and gesture recognition of the Oculus Quest 2 are implemented using the XR Hands SDK in Unity, which tracks the user's hand movements. The headset's cameras stream live video to Unity, where the XR Hands package provides real-time tracking of the (x,y,z) coordinates of several joints of the hands including the wrist, the palm, and individual finger joints as demonstrated in Figure 4. Moreover, the XR hands package provides gesture recognition based on the hand's shape and orientation. Finally, the aforementioned collected data about the user's hand is used to design and implement three different VR-based methods to actuate the robotic arm. After recognizing the user's hand gestures and movements, they are mapped to actuation commands. These commands are propagated from Unity to the digital ROS Master through a TCP endpoint as demonstrated in Figure 2. Once the messages are received, the Dockerized private cloud is then responsible for managing the communication between the digital and physical ROS Masters. It forwards the ROS messages to the physical ROS Master, which is responsible for actuating the physical assets based on the received ROS messages.

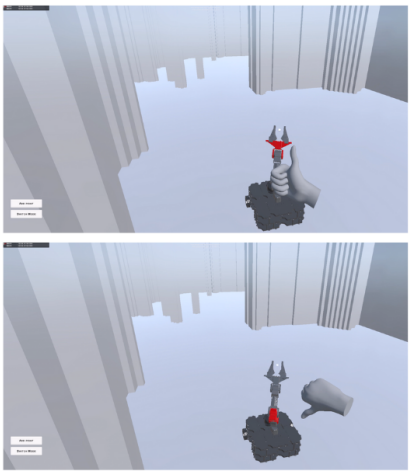


Figure 5: Method 1 : Sequential Selection.

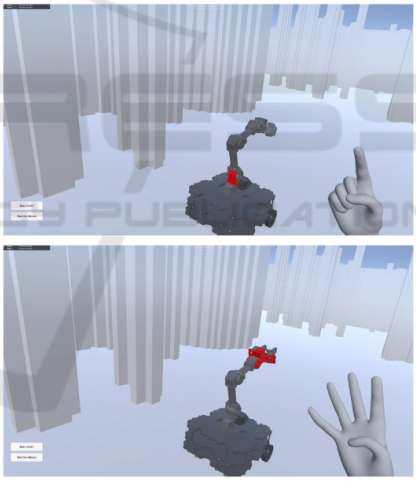


Figure 6: Method 2 : Numerical Selection.

The three actuation methods are:

Method 1, Sequential Selection: It utilizes hand gesture recognition of the user's hand to perform the selection of the target joint out of the five joints in the robotic arm, in addition to the actuation of the selected joint shown in Figure 5, given that the joints are numbered from 1 to 5. The user can switch between the five joints sequentially, using thumbs-up and thumbs-down gestures to switch to the next upper or lower joint, respectively. To actuate the arm based on the selected joint, highlighted in red in Figure 5, the opening or closing hand gesture can be detected and used to increase or decrease the angle of the se-

lected joint, respectively. This method utilizes two separate and independent procedures for the selection and actuation of the arm joints.

Method 2, Numerical Selection: Similar to method 1 in terms of the actuation of the selected joint. However, the selection is numerical. The user shows the index of the target joint using their hand, and the respective joint is selected and highlighted in red, as demonstrated in Figure 6. This method also separates the selection and actuation of the arm joints.

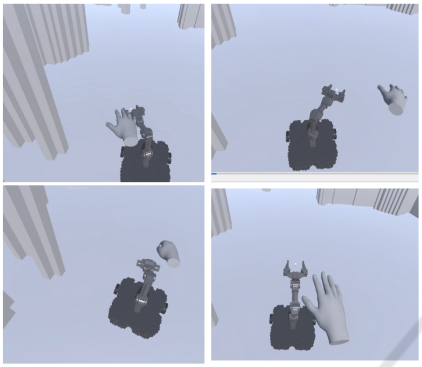
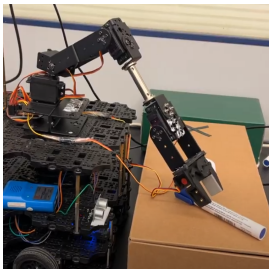


Figure 7: Method 3: Mixed Selection and Actuation.

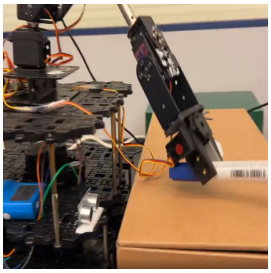
Method 3, Intuitive Control: This method differs from the first two by integrating the selection and actuation of the joints together. In this approach, the user's hand movements are tracked, enabling the robotic arm to replicate them as demonstrated in Figure 7 (e.g., if the user moves their hand upward, the robotic arm moves upward). The hand's position along the x-axis controls the first joint (Servo Motor 1), allowing the arm to rotate, while movements along the z-axis control the second joint (Servo Motor 2) for forward and backward motion. Similarly, movements along the y-axis adjust the third joint (Servo Motor 3) for upward and downward motion. The orientation of the user's hand dictates the position of the fourth joint (Servo Motor 4); for example, when the hand points downward, the gripper points downward, and vice versa. Additionally, hand gestures, such as opening or closing the palm, control the gripper (Servo Motor 5). The gripper closes when the user's hand is closed and opens when the hand is open.

4.3 User Study of the Arm Actuation Methods

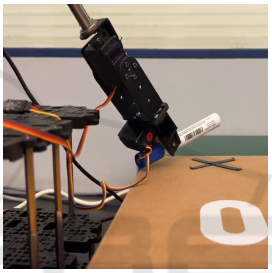
To ensure an immersive user experience for the human controller, a user study was conducted with two main objectives. The first objective was to test the integration of VR features into the system and evaluate the performance of the VR-based features and actuation methods compared to traditional methods (e.g., keyboard controls and dashboard buttons). The second objective was to compare the three VR-based actuation methods for the robotic arm, identify the advantages and shortcomings of each, and determine the most effective and immersive method based on the collected data.



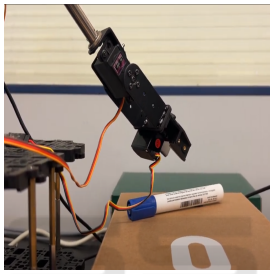
(a) Task 1: move the arm close to the object.



(b) Task 2: pick up the object (e.g., pen).



(c) Task 3: move the pen to the target position.



(d) Task 4: Drop the object (e.g., the pen).

Figure 8: The sequence of tasks in the user study.

The user study was conducted as follows: 15 volunteers were randomly selected from the university, and most of the volunteers were asked to test two of the actuation methods, as we assumed that asking every volunteer to test each of the actuation methods could be overwhelming. In an effort to ensure fairness and reduce bias, the methods each user evaluates, as well as their evaluation order, are determined randomly. Before each volunteer begins evaluating an actuation method, they are provided with a detailed explanation of control instructions for the respective method. Once they complete the evaluation of the first selected method, they are asked to take a 15-minute break to reset and minimize confusion with the upcoming selected method for evaluation. After the break, the control instructions for the next method are explained before the volunteer begins the subsequent evaluation. For each evaluation procedure, volunteers were then asked to wear the Quest 2 headset and perform a sequence of four tasks:

1. Move the arm close to the target object(e.g., a pen), shown in Figure 8(a).

2. Use the gripper to pick up the target object, as shown in Figure 8(b).
3. Move the arm to the target position, as shown in Figure 8(c).
4. Open the gripper and drop the object to the target position, as shown in Figure 8(d).

To achieve the objectives of this study, both quantitative and qualitative metrics were specified. For the quantitative metrics shown in Table 1, two key measures were chosen: (1) Completion time, defined as the average time required for a volunteer to complete the four aforementioned tasks; and (2) Offset error, which represents the distance between the target position of the object (e.g., a pen) and its final position after the volunteer's attempt to complete the sequence of tasks. These quantitative metrics provide objective and measurable data that helps assess the developed actuation methods. The completion time reflects how efficient each method is, while the offset error measures how accurate volunteers were in completing tasks. In contrast, qualitative metrics shown in Table 2 capture subjective user experiences, focusing on factors such as the intuitiveness of each actuation method and the level of user satisfaction. These metrics are crucial for evaluating the actuation methods, as they offer insights into how users perceive the system and help assess the user-friendliness of the developed features, insights that cannot be easily inferred from quantitative metrics alone.

Table 1: Quantitative Results of the user study.

Method	Completion Time (in minutes)	Offset error (in cm)
Method 0 (Dashboard)	04:13	7.3
Method 1 (Sequential)	03:36	6.67
Method 2 (Numerical)	03:34	7.72
Method 3 (Intuitive)	01:28	2.2

As demonstrated in Table 1, all three novel VR-based actuation methods resulted in faster completion times compared to the traditional dashboard button control method. The average completion time for the traditional method was over 4 minutes (4:13), while the average completion time for the VR-based methods was notably lower, particularly for method 3 (1:28). These results clearly indicate that VR integration significantly improved efficiency, with volunteers completing the tasks more quickly using the VR-based actuation methods. Among the VR actuation methods, method 3 proved to be the most efficient (1:28), showing a substantial time difference compared to methods 1 and 2 (around 3:30).

Table 2: Qualitative Results of the user study.

Method	User-Satisfaction (out of 5)	Intuitiveness (out of 5)
Method 0 (Dashboard)	3.1	2.4
Method 1 (Sequential)	3.83	3.33
Method 2 (Numerical)	3.58	3.56
Method 3 (Intuitive)	4.55	4.90

For the average offset error measurements shown in Table 1, the offset error of the traditional dashboard actuation method was not much larger than that of methods 1 and 2, with all methods exhibiting an average offset error of around 7 centimeters, except for the third VR-based method. Method 3 demonstrated significantly better accuracy, with an average offset error of only 2.2 centimeters, making it the most efficient and accurate of the proposed methods.

The qualitative results presented in Table 2 show that the traditional actuation method (method 0) received the lowest user ratings in both intuitiveness (2.6/5) and user satisfaction (2.1/5) compared to the VR-based actuation methods. On the other hand, the third VR-based method received the highest user ratings, with a score of 4.9/5 for intuitiveness and 4.55/5 for user satisfaction, significantly outperforming methods 1 and 2. The high user ratings for intuitiveness, particularly for method 3, likely explain why volunteers were able to complete the sequence of tasks much faster (as shown in Table 1) compared to the traditional dashboard method.

5 CONCLUSION AND FUTURE WORK

Digital Twins are powerful tools that can be utilized throughout the whole manufacturing life-cycle, from designing and planning to maintaining existing facilities. Additionally, they are beneficial for telepresence, remote control, and situations where human presence is risky, such as rescue missions. This paper presents a novel and comprehensive framework for integrating Virtual Reality with a Digital Twin architecture, using a Dockerized private cloud for minimal communication latency. Our project yields significant insights into the practical applications of Digital Twins in robotics.

Through this research, the feasibility and effec-

tiveness of integrating Digital Twins with robotic systems have been demonstrated, particularly in areas such as robotic arm control, and virtual reality interfaces. The user study, in particular, shows the advantages of integrating the VR interfaces in robot and arm control. As the results of the user study clearly show that the VR-based methods have generally performed better than the traditional dashboard control in almost all of the aforementioned quantitative and qualitative metrics. Moving forward, future work will focus on refining and expanding the capabilities of Digital Twins in robotics for critical missions. This includes exploring advanced AI algorithms for obstacle avoidance and autonomous path suggestions, as well as developing robust communication systems for remote operation. Additionally, efforts will be made to secure data within the Dockerized private cloud to ensure the safety and integrity of sensitive information. Future work will also focus on optimizing the system for portable, low-power hardware to make it more adaptable in real-world scenarios. Furthermore, we aim to extend the range of hand gestures for actuation and allow user personalization to improve the system's versatility and user experience. Finally, real-world testing and validation of our approach in simulated and controlled rescue scenarios will be conducted, with the ultimate goal of deploying these technologies in emergency response situations to save lives and mitigate disaster impacts.

SCIENCE AND TECHNOLOGY PUBLICATIONS

REFERENCES

Abdulla, R., Gadalla, A., and Balakrishnan, A. (2020). Design and implementation of a wireless gesture controlled robotic arm with vision.

Alexandru, M., Dragos, C., and Zamfirescu, B.-C. (2022). Digital twin for automated guided vehicles fleet management. *Procedia Computer Science*.

Allspaw, J., LeMasurier, G., and Yanco, H. (2023). Comparing performance between different implementations of ros for... OpenReview.

Amsters, R. and Slaets, P. (2020). Turtlebot 3 as a robotics education platform. In *Chapter in Robotics Education*.

Attaran, M. and Celik, B. G. (2023). Digital twin: Benefits, use cases, challenges, & opportunities. *Decision Analytics Journal*.

Crespi, N., Drobot, A., and Minerva, R. (2023). The digital twin: What & why? In *Springer eBooks*.

Ellethy, M. et al. (2023). A digital twin architecture for automated guided vehicles using a dockerized private cloud.

Estefo, P., Simmonds, J., Robbes, R., and Fabry, J. (2019). The robot operating system: Package reuse and community dynamics. *Journal of Systems and Software*, 151:226–242.

Gallala, A., Kumar, A. A., Hichri, B., and Plapper, P. (2022). Digital twin for human–robot interactions by means of industry 4.0 enabling technologies. *Sensors*, 22(13):4950.

Gigante, M. (1993). Virtual reality: Definitions, history and applications.

Mazumder, A. et al. (2023). Towards next-generation digital twin in robotics: Trends, scopes, challenges, & future. *Heliyon*, 9(2).

Pirker, J., Loria, E., Safikhani, S., Künz, A., and Rosmann, S. (2022). Immersive virtual reality for virtual and digital twins: A literature review to identify state of the art and perspectives. In *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 114–115, Christchurch, New Zealand.

Platt, J. and Ricks, K. (2022). Comparative analysis of ros-unity3d and ros-gazebo for mobile ground robot simulation. *Journal of Intelligent & Robotic Systems*.

Riley, J. M. and Endsley, M. R. (2004). The hunt for situation awareness: Human-robot interaction in search and rescue. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 48(3):693–697.

Saddik, A. E. (2018). Digital twins: The convergence of multimedia technologies. *IEEE Journals & Magazine – IEEE Xplore*.