

Virtual Reality Control of a Planetary Rover Manipulator Using Unity and Unreal Engine

Paweł Kwiaton^{1*}, Krzysztof Sokół¹, Grzegorz Ptak¹, Piotr Ptak¹ and Dawid Cekus¹

¹Department of Mechanics and Machine Design Fundamentals, Faculty of Mechanical Engineering, Czestochowa University of Technology, Dąbrowskiego 73, 42-201 Czestochowa, Poland

*Corresponding author: pawel.kwiaton@pcz.pl

Submitted 28 August 2026; Revised 10 October 2026; Accepted 15 October 2026; Available online 28 October 2026.

Copyright © 2026 The Authors.

Abstract: This paper presents a method for controlling a manipulator mounted on a planetary rover using virtual reality (VR) technology. The proposed approach combines VR controllers with stereovision cameras to provide the operator with an improved view of the manipulator workspace. Unity and Unreal Engine were used to visualize the rover and the manipulator in a virtual environment. The proposed control system was implemented and tested on a real rover, and its operation was compared with a conventional joystick-based control system. In addition, the performance of both graphics engines was evaluated during object rendering operations using selected CPU, GPU, RAM, and frame-rate metrics. The results indicate that the proposed VR-based approach can reduce task completion time while providing immersive visual feedback during manipulator operation. The developed system can also be adapted to other mobile or industrial platforms equipped with manipulators.

Keywords: Control system; Game engines; Manipulator; Planetary rover; Virtual reality.

1. INTRODUCTION

Manual control of a manipulator requires considerable concentration and operator experience. In industrial robots, this type of control is mainly used during initial programming. However, in mobile platforms equipped with manipulators, effective operation requires good spatial orientation and simultaneous hand-eye coordination using a joystick and camera feedback. Since the 1970s, the development of manned space missions and planetary exploration has led to an increasing use of robotic manipulators in tasks that are unsafe or impossible for humans to perform directly [1]. In planetary teleoperation, communication delays between Earth and Mars can reach several minutes and represent an additional challenge. Moreover, mobile robots and their manipulators are commonly operated using images acquired from onboard cameras. Due to limited data transmission capacity and the restricted spatial information provided by conventional 2D images, perception of the manipulator workspace may be difficult. Virtual reality (VR) can provide the operator with a more complete representation of the environment surrounding the rover [2]. Therefore, VR can also support Human-Robot Cooperation (HRC) approaches [3], which have gained increasing attention with the development of Industry 4.0 technologies [4].

Control systems for manipulators mounted on planetary rovers have been investigated from different perspectives. However, many studies focus primarily on mechanical design [5, 6], structural analysis [7], or optimization [8], while fewer works address alternative methods of manipulator control. In [9], an algorithm for controlling a Martian rover manipulator in an autonomous mode was proposed. The control algorithm was developed in Matlab by solving the inverse kinematics problem and implementing linear motion interpolation. The proposed system was evaluated in simulation to determine the maximum arm velocity and an appropriate operating scenario. The problem of remote control of a robotic arm mounted on a planetary rover was also investigated in [10]. In this approach, the operator controlled the manipulator using a specially designed glove. Experimental tests showed that the proposed device allowed tasks to be completed faster than with the dedicated manufacturer controller. In [11], the forward kinematics problem of a planetary rover manipulator was considered. The study focused on an analytical model and compared a formulation based on classical mechanics with the Denavit-Hartenberg notation, without implementing the algorithm in a complete control system.

A lightweight manipulator and its control system were presented in [12]. A distributed control architecture based on the Controller Area Network bus was developed to enable coordinated multi-axis control. In this solution, the computer was responsible for trajectory planning, while the actuators performed interpolation using the PVT method. The kinematic parameters of the manipulator were also determined, and the inverse kinematics problem was solved geometrically. Virtual reality has also been used for safe mission planning of planetary rovers [13], where a three-dimensional environment based on actual Martian

terrain was used to determine an optimized and safe rover path.

Virtual reality is currently used in many engineering applications and is closely related to technologies such as cyber-physical systems, the Internet of Things, and Industry 4.0 [14]. In [15], VR was applied to support interaction between a human operator and a robot. Assembly operations demonstrated in a virtual environment were reproduced by an industrial robot as sequences of movements, achieving a reported activity recognition accuracy of 95%. Virtual reality has also been applied to industrial robot control [16]. A Digital Twin consisting of an industrial robot and a Unity-based virtual environment was developed to reproduce the configuration of the physical workstation. The comparison of virtual and physical robot trajectories indicated a small delay in the motion of the virtual model. Digital Twin and VR technologies in the context of Industry 4.0 were also considered in [17], where a virtual workstation was developed for testing and operator training before implementation on a physical production system. Human-robot interaction using VR was investigated in [18], with particular attention to robot perception and stereoscopic signals.

VR and augmented or mixed reality have also been applied to the design and operation of robotic systems. In [19], an integrated platform combining VR and AR was proposed for the design of human-robot collaborative workspaces. The approach enabled the estimation of basic production parameters, including cycle time and process planning, and supported system optimization. A mixed-reality robot control system using HoloLens and Unity3D was presented in [20], including the implementation of inverse kinematics in the virtual environment. Mixed reality and the Unity engine were also used for teleoperation of mobile manipulators in [21]. The proposed architecture consisted of an operator control station and a remote robotic system and was experimentally evaluated using a KUKA industrial robot. The results indicated that mixed reality can improve the visualization of the area around the manipulator gripper. Similar robotic and autonomous systems have also been investigated in other applications, including cultural heritage [22] and medical environments [23].

More recent studies confirm the continued development of immersive interfaces for robotic teleoperation. Recent studies published in the field of modelling and simulation also demonstrate the increasing use of virtual models and simulation environments for the development and validation of robotic systems, including digital-twin-based mobile robots [24] and simulation-based control of robotic manipulators [25]. Galarza et al. [26] developed a VR teleoperation system for a mobile manipulator using Unity and a virtual representation of the robot. Gallipoli et al. [27] proposed a VR-based dual-mode teleoperation architecture combining immersive interaction with robotic manipulation and evaluated the system experimentally. More recently, George et al. [28] presented an Oculus-based VR teleoperation framework for controlling a robotic manipulator using commercially available hardware. These studies demonstrate the continuing interest in combining virtual environments, immersive interfaces, and physical robotic manipulators.

Compared with the existing studies, the present work combines several elements within a single experimentally implemented system. The main contribution is the integration of VR-based manipulator control, stereoscopic visual feedback, and a physical planetary rover platform. In addition, equivalent virtual environments were developed in both Unity and Unreal Engine, which enabled a direct comparison of their computational performance using CPU, GPU, RAM, and FPS metrics under the considered application conditions. Finally, the developed VR-based control approach was verified on the physical manipulator and compared with the conventional joystick-based control method.

In this work, a virtual-reality-based control and simulation system for a planetary rover manipulator is presented (Figure 1). The system was designed and implemented at Czestochowa University of Technology as part of the University Rover Challenge project. The proposed approach uses virtual reality to support the operator during manipulator control and combines a stereovision system with a virtual representation of the rover. Unity and Unreal Engine were used to implement the virtual environment, and their computational performance was evaluated on two computers with different hardware specifications. CPU, GPU, RAM usage, and the number of rendered frames per second were measured during the tests. Finally, the developed control system was experimentally verified on the physical rover (Figure 2) and used during the 2021 Virtual Rover Challenge. Although the physical implementation originates from the 2021 project, the focus of the present study is the developed control architecture, its integration with the physical rover, and its computational and experimental evaluation rather than the novelty of the specific VR hardware used at that time.

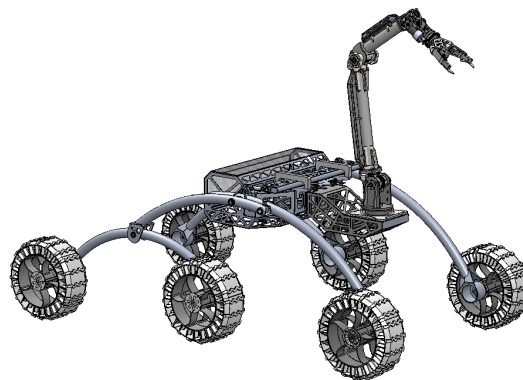


Figure 1. Modernity III rover model made in SolidWorks.



Figure 2. Modernity III rover (real object).

2. KINEMATIC MODEL OF THE MANIPULATOR

The kinematic model of the considered four-degree-of-freedom anthropomorphic manipulator is based on the geometric formulation presented in [29]. The model defines the geometric relationships between the manipulator links, the admissible configurations, and the position and orientation of the gripper. In the implemented control system, however, the inverse kinematics problem is solved online using the FABRIK algorithm. Therefore, the analytical relations presented in this section describe the manipulator kinematics and provide the geometric basis for the control model, while the real-time joint configuration is determined numerically by FABRIK.

The geometry of the considered manipulator is shown in Figure 3. The objective of the kinematic formulation is to determine the relationships between the joint variables ϑ_1 , ϑ_2 , ϑ_3 , and ϑ_4 , the prescribed position of the gripper point O_4 , and the inclination angle η of the last manipulator link. The length L_i denotes the distance between points O_{i-1} and O_i .

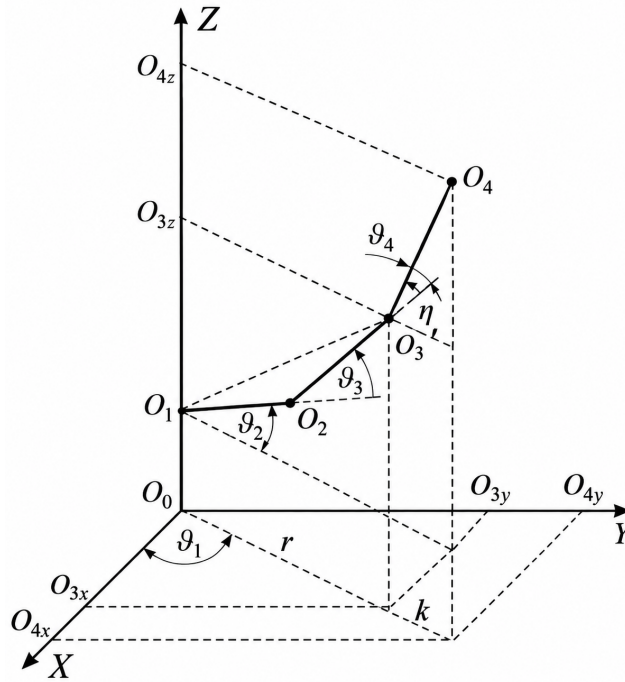


Figure 3. Diagram of the anthropomorphic manipulator [29].

The position of point O_3 is determined from the prescribed gripper position O_4 and the orientation of the last manipulator link. Considering the triangle $O_1O_2O_3$, the following relation can be written:

$$O_{3x}^2 + O_{3y}^2 + (O_{3z} - L_1)^2 = L_2^2 + L_3^2 + 2L_2L_3c_3, \quad (1)$$

where $c_3 = \cos(\vartheta_3)$. Therefore,

$$c_3 = \frac{O_{3x}^2 + O_{3y}^2 + (O_{3z} - L_1)^2 - L_2^2 - L_3^2}{2L_2L_3}. \quad (2)$$

The corresponding sine component is

$$s_3 = \pm \sqrt{1 - c_3^2}, \quad (3)$$

and the joint angle ϑ_3 is obtained as

$$\vartheta_3 = \text{atan2}(s_3, c_3). \quad (4)$$

The remaining angle ϑ_2 is determined from the geometric relations of the triangle $O_1O_2O_3$. Considering both possible signs of s_3 and the corresponding manipulator configurations results in multiple solutions of the inverse kinematics problem. The complete analytical derivation of these solutions is provided in [29].

The orientation of the gripper is related to the manipulator joint variables by

$$\eta = \vartheta_2 + \vartheta_3 + \vartheta_4, \quad (5)$$

and therefore

$$\vartheta_4 = \eta - \vartheta_2 - \vartheta_3. \quad (6)$$

Together with the possible orientations of the first joint ϑ_1 , four admissible manipulator configurations are considered in the geometric model. For the singular case $O_{4x} = 0$ and $O_{4y} = 0$, the angle ϑ_1 cannot be uniquely determined. These geometric relations define the manipulator structure and its admissible configurations, whereas the real-time inverse-kinematics solution used during operation is obtained using the FABRIK algorithm.

3. IMPLEMENTATION OF THE CONTROL SYSTEM

The developed control system consists of the physical planetary rover, the manipulator controller, the operator control panel, the virtual-reality interface, and the simulation module. Communication between the physical and virtual components enables the manipulator state to be reproduced in the virtual environment and allows control commands generated by the operator to be transferred to the rover. The overall architecture of the developed system is shown in Figure 4.

3.1 Hardware and Three-dimensional Model

The operator interface was based on Oculus Rift S VR goggles equipped with two Oculus Touch motion controllers (Figure 5). The headset provides a resolution of 2560×1440 pixels, a field of view of approximately 110° , and a refresh rate of 80 Hz. The Oculus Touch controllers were used to generate commands corresponding to the desired manipulator motion. To reduce the influence of unintended hand movements, an adjustable dead-zone parameter was introduced.

A StereoPi stereoscopic camera was used to provide visual feedback from the rover (Figure 6). The system consists of a Raspberry Pi-based platform and two cameras with a resolution of 3280×2464 pixels each and a baseline of 60 mm. The stereoscopic image provides additional depth information that assists the operator during manipulator positioning. Communication with the rover can be performed through Wi-Fi or Ethernet.

A three-dimensional model of the rover manipulator was prepared in Blender 2.90.0 [31] (Figure 7). Individual manipulator links were arranged in a hierarchical structure reflecting the physical kinematic chain. This structure allows the motion of individual links to be reproduced in the simulation environment. The completed model was subsequently exported to Unity and Unreal Engine.

3.2 Virtual Environment Implementation

The manipulator model was imported into Unity 2020.3 LTS [32], where its scale, joint locations, and hierarchical structure were verified. The XR Plugin and Oculus Integration libraries were used to provide support for the VR headset and Oculus Touch motion controllers.

Three main software modules were implemented: inverse kinematics, gripper control, and communication with the rover. The inverse-kinematics module uses the FABRIK algorithm to determine the manipulator configuration corresponding to the target position commanded by the operator through the VR controller. The geometric model described in Section 2 provides an analytical description of the manipulator geometry, link lengths, and admissible configurations, whereas FABRIK provides the numerical solution of the inverse kinematics problem during operation. The gripper module controls the end-effector motion, and the resulting manipulator configuration is transferred to the physical control system.

The stereoscopic image from the StereoPi camera is displayed in the Unity environment using two separate image planes, as shown in Figure 8. Two image streams are processed independently and displayed for the left and right eyes. The camera data are processed using Python and OpenCV [33] and transferred to the VR application.

An equivalent virtual environment was also implemented in Unreal Engine 4.26 [34] to enable comparison of the computational performance of both graphics engines. The same three-dimensional manipulator model and control assumptions were used. The VR interface and FABRIK-based manipulator control were implemented using the corresponding Unreal Engine components. Unlike the Unity implementation, the stereoscopic video stream could be handled using functionality available directly in the engine. The completed VR environment is shown in Figure 9.

3.3 Communication Between the Software and the Rover

Communication between the control computer and the rover was based on the UDP protocol, which was selected because of its low communication overhead and suitability for applications in which transmission delay is important [35]. The VR application

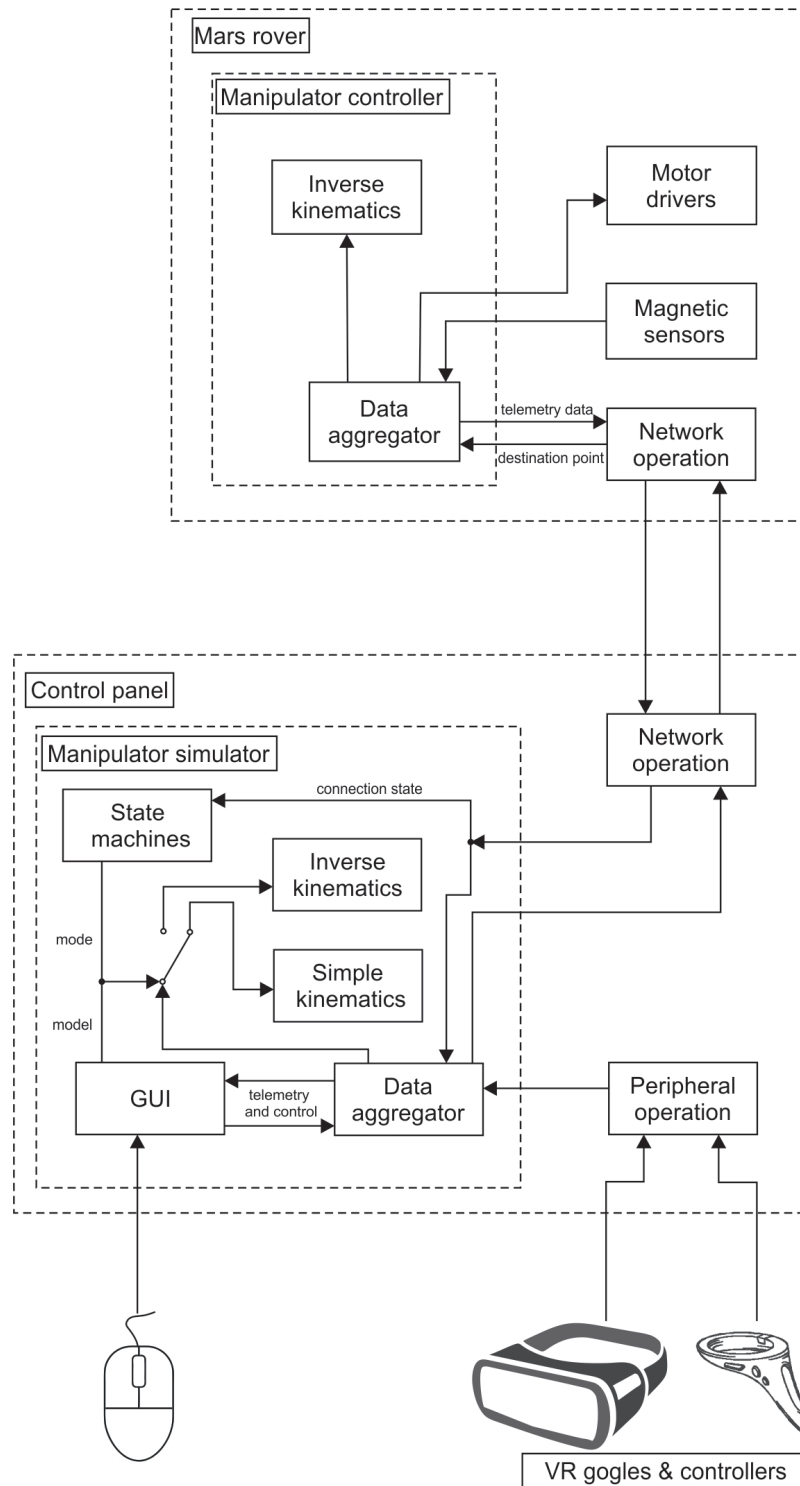


Figure 4. Architecture of the developed manipulator control and simulation system.



Figure 5. VR headset and controllers used in the developed control system.



Figure 6. StereoPi stereoscopic camera used in the control system [30].

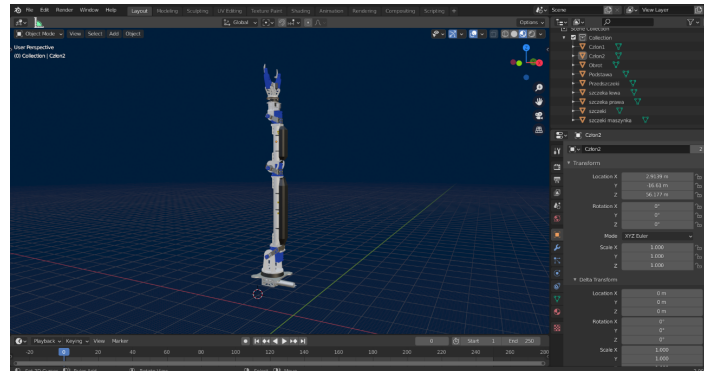


Figure 7. Manipulator model prepared in Blender [30].

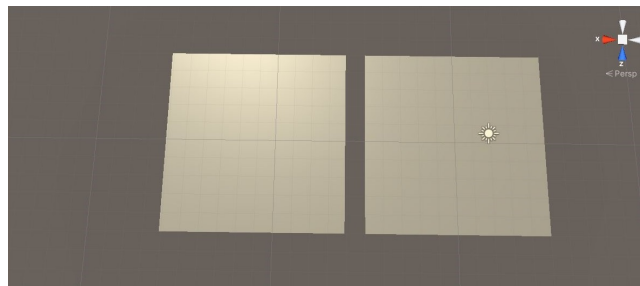


Figure 8. Image planes used to reproduce the stereoscopic camera view in Unity [30].

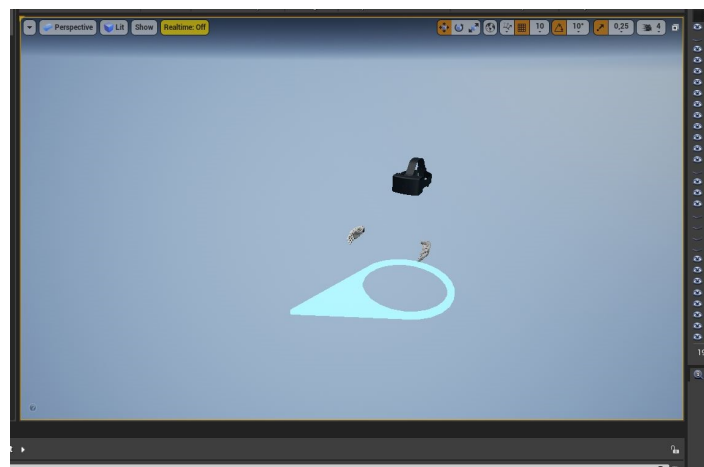


Figure 9. VR environment implemented in Unreal Engine.

communicates with the main rover control software through an internal socket, while control data are subsequently transmitted to the rover.

Asynchronous communication methods were used to minimize the computational overhead of data exchange. The integrity of transmitted control data was verified using a CRC-8 checksum [36]. The resulting communication structure is shown in Figure 10.

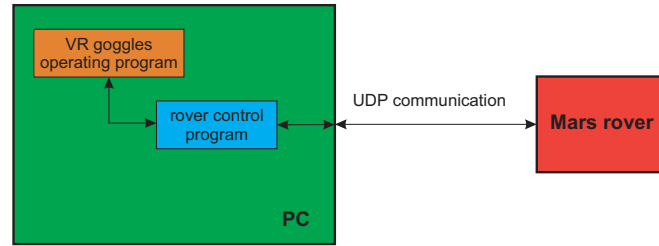


Figure 10. Communication between the VR application, rover control software, and planetary rover.

4. PERFORMANCE EVALUATION OF THE GRAPHICS ENGINES

The aim of this part of the study was to compare the computational performance of the Unity and Unreal Engine implementations of the developed virtual-reality environment. The evaluation was based on four performance indicators commonly used for graphics-engine comparison [37]:

- CPU usage;
- GPU usage;
- RAM usage;
- number of rendered frames per second (FPS).

The developed applications were tested on two computers with different hardware specifications (Table 1). The use of two hardware configurations allowed the performance of both graphics engines to be evaluated on a relatively high-performance computer and on an older, less powerful platform.

Table 1. Hardware specifications of the computers used in the performance tests.

	Computer I	Computer II
Type	Laptop	Laptop
CPU	Intel® Core™ i7-9750H 2.60 GHz	Intel® Core™ i5-5200U 2.20 GHz
GPU	NVIDIA GeForce GTX 1660 Ti 6GB	NVIDIA GeForce 940MX 4GB
RAM	2×16 GB	2×4 GB

4.1 Test Setup and Data Acquisition

The performance comparison was carried out using Unity 2020.3 LTS and Unreal Engine 4.26. Performance data were collected using the *cpu_stats* script and saved to CSV files for subsequent analysis. The measurements were additionally monitored using MSI Afterburner to independently verify the recorded CPU, GPU, and memory usage.

For both graphics engines, the same three-dimensional manipulator model and comparable virtual scenes were used. Both applications were evaluated on the same two hardware configurations listed in Table 1, thereby reducing the influence of differences in hardware and scene complexity on the performance comparison.

4.2 Performance Test Results

Three tests were performed for each combination of computer and graphics engine, and the averaged results are presented in Figures 11 and 12. CPU and GPU usage are shown in Figure 11, while RAM usage and the number of rendered frames per second are presented in Figure 12.

4.3 Discussion

The obtained results show clear differences in computational resource usage between the Unity and Unreal Engine implementations. However, the results should be interpreted in the context of the developed manipulator application and the selected hardware configurations. Direct comparison of absolute CPU and GPU usage values with other studies is limited because graphics-engine performance strongly depends on scene complexity, rendering settings, hardware, and the type of implemented application.

For Unity, the maximum CPU usage reached approximately 37% on Computer I and 69% on Computer II. The corresponding values for Unreal Engine were approximately 31% and 60%, respectively. The average CPU usage values are presented in Table 2. Unreal Engine required less CPU resources in both hardware configurations. The average CPU load decreased from 33.22% to 20.95% on Computer I and from 66.45% to 46.59% on Computer II. This corresponds to reductions of approximately 37% and 30%, respectively.

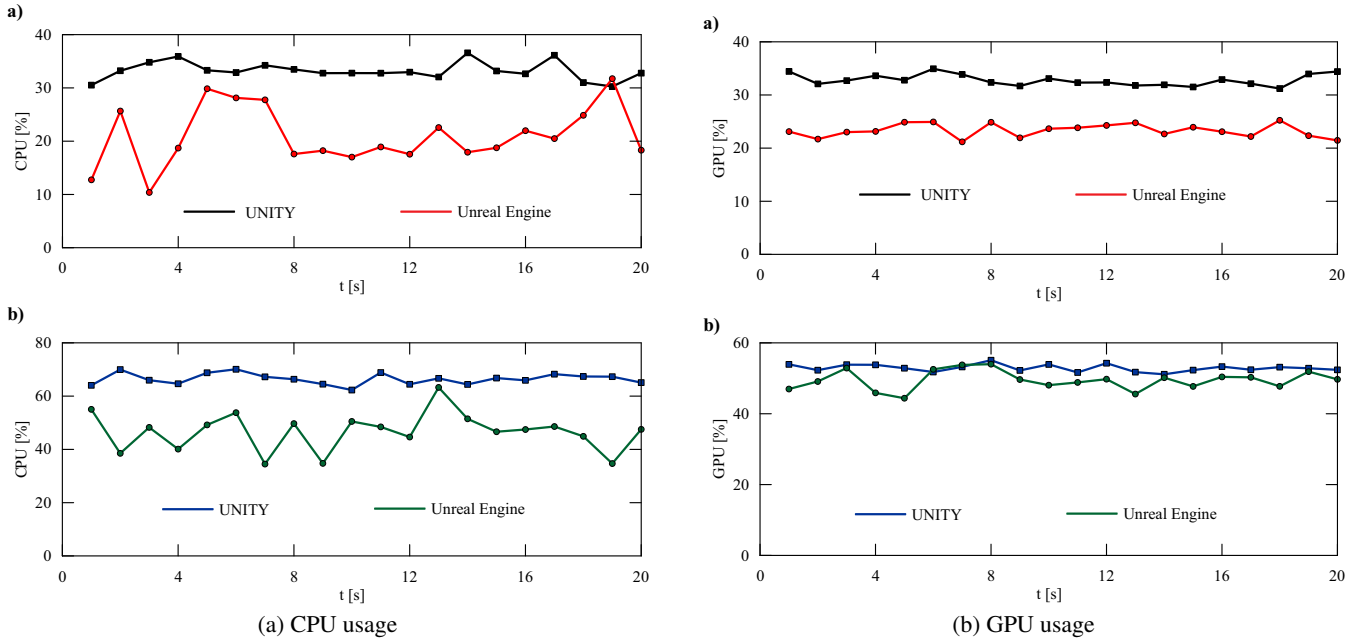


Figure 11. Processor and graphics-card usage during the performance tests: (a) CPU usage; (b) GPU usage. Each panel contains results for Computer I and Computer II.

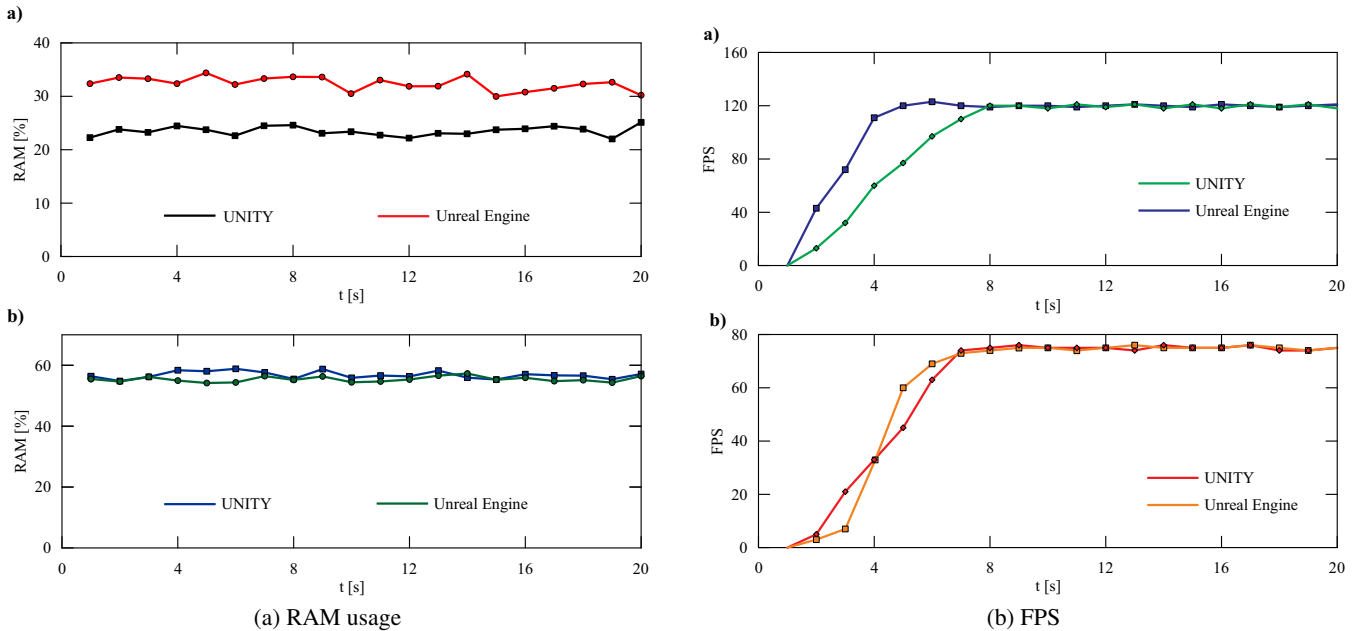


Figure 12. Memory usage and rendering performance: (a) RAM usage; (b) number of rendered frames per second. Each panel contains results for Computer I and Computer II.

A similar tendency was observed for GPU usage. For Computer I, the maximum GPU load was approximately 35% for Unity and 25% for Unreal Engine. For Computer II, the difference between the maximum GPU loads was considerably smaller. The average values presented in Table 2 confirm that Unreal Engine required less GPU resources in both configurations, although the difference was more pronounced for the higher-performance computer.

Table 2. Average CPU and GPU usage during the performance tests.

Engine	CPU usage [%]		GPU usage [%]	
	Computer I	Computer II	Computer I	Computer II
Unity	33.22	66.45	32.08	52.91
Unreal Engine	20.95	46.59	23.30	48.06

The observed tendencies differ partly from those reported in [37]. In that study, Unity showed more favorable CPU performance, whereas Unreal Engine was characterized by a higher GPU load. Such differences can be attributed to the different applications and hardware configurations used in the two studies. In the present work, the rendered object was a relatively compact manipulator model imported from a 3D modelling environment. In [37], a considerably more complex virtual museum

environment containing multiple detailed objects was investigated. Therefore, the comparison with the literature should be considered qualitative rather than as a direct comparison of absolute performance values.

The RAM usage results show a different relationship. On Computer I, Unity required approximately 9% less memory than Unreal Engine, whereas on Computer II the difference was only about 0.5%. The lower RAM usage of Unity on the more powerful computer is consistent with observations reported in [37] and [38]. For the lower-performance computer, the memory requirements of both engines were comparable.

The FPS results are presented in Figure 12(b). Both implementations reached a stable frame rate after the initial loading period. The average frame rate was approximately 120 FPS on Computer I and approximately 75 FPS on Computer II. On Computer II, Unity produced slightly fewer frames than Unreal Engine; however, the differences were not sufficient to identify a clear overall advantage of either engine based solely on the FPS criterion. A similar observation was reported in [37].

Overall, the results demonstrate that the choice of graphics engine involves a trade-off between different computational resources. For the developed VR manipulator application, Unreal Engine showed lower CPU and GPU usage on both tested computers, whereas Unity showed lower or comparable RAM usage. The FPS results were similar for both engines. Therefore, neither engine can be considered universally superior; the preferred solution depends on the available hardware resources and the requirements of the specific VR application.

5. TESTS ON A REAL OBJECT

After implementation of the developed control system, experimental tests were performed on the physical rover to verify the response of the manipulator to the operator's commands (Figure 13(a)). The test consisted of moving the manipulator gripper along a predefined path made of bent wire (Figure 13(b)). The objective was to complete the path while maintaining continuous contact between the gripper tip and the wire. Contact was indicated by a red LED, and the test was repeated if contact was lost.

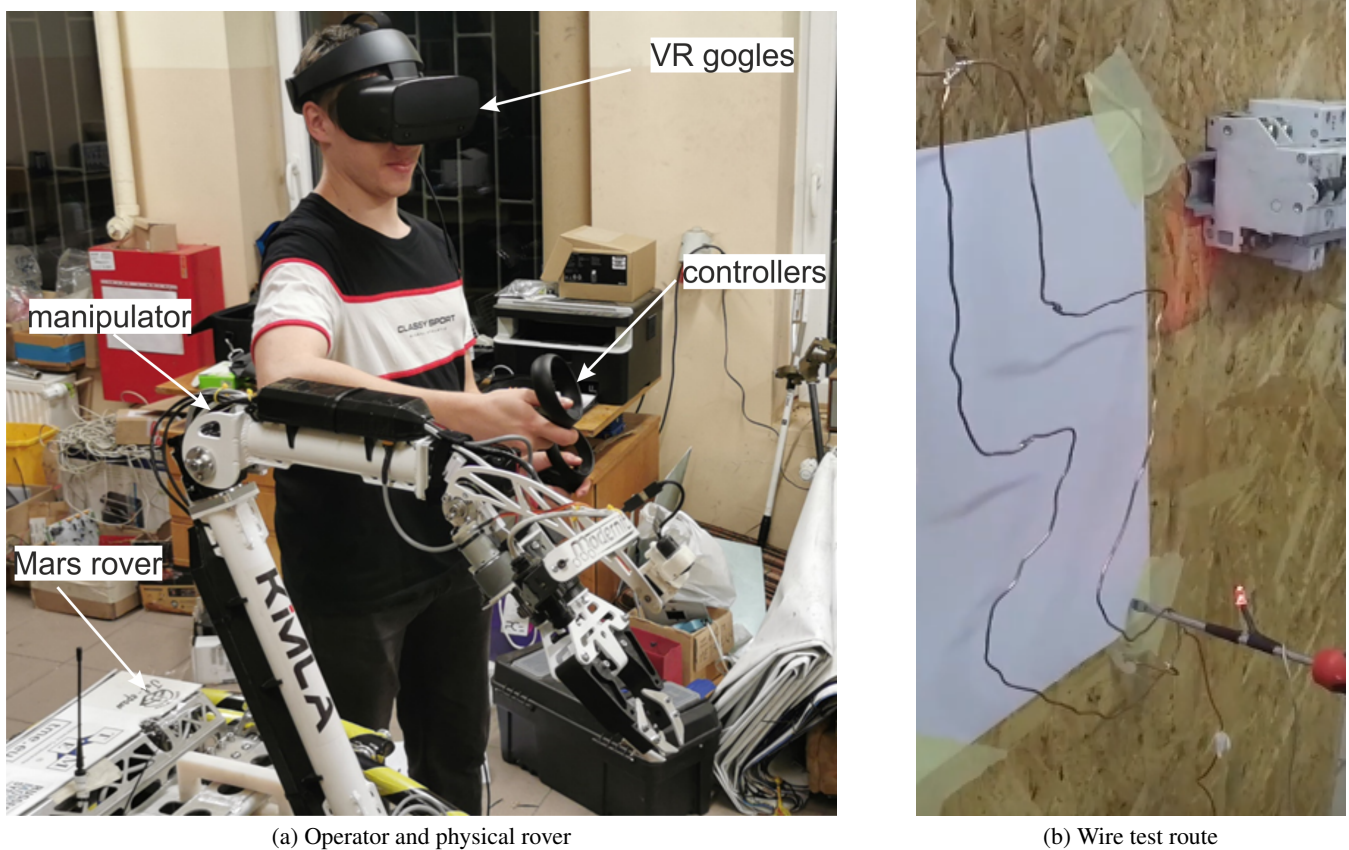


Figure 13. Experimental verification on the physical rover: (a) operator performing the test using the VR interface; (b) test route used for manipulator guidance.

The task was performed twice using the conventional joystick-based control system and twice using the proposed VR-based control system. The purpose of these trials was to provide a preliminary functional verification of the developed control concept on the physical rover rather than to perform a statistically representative human-operator study. The measured completion times are presented in Table 3. Within these preliminary trials, the average task-completion time was approximately 14.99 s for the conventional control system and 12.00 s for the VR-based system, corresponding to an approximately 20% reduction in completion time.

During the tests, the VR-based control also resulted in a smoother gripper trajectory than the conventional joystick-based control. Example trajectories obtained using both control methods are shown in Figure 14. The VR-based trajectory was shorter and contained fewer abrupt changes in direction.

The presented results should be treated as a preliminary experimental verification of the developed control concept. Due to the limited number of trials, the results are not sufficient for statistical evaluation of operator performance. A more extensive study involving a larger number of users and repeated trials would be required to quantitatively assess usability, learning effects, and repeatability.

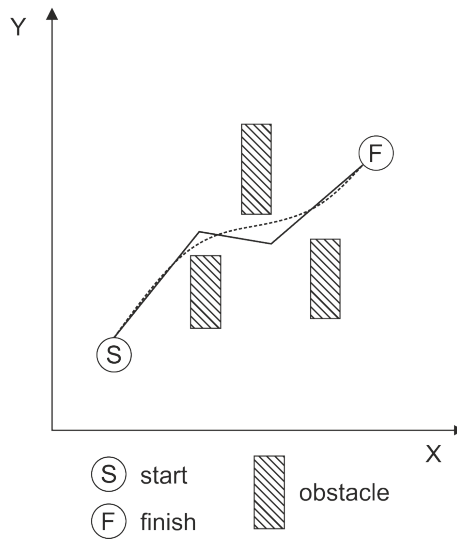


Figure 14. Example gripper motion paths obtained using conventional control (solid line) and VR-based control (dashed line).

Table 3. Comparison of manipulator test-route completion times.

Control method	Attempt 1	Attempt 2
Conventional joystick control	15.25 s	14.72 s
VR-based control	12.32 s	11.68 s

6. CONCLUSIONS

A virtual-reality-based control and simulation system for a planetary rover manipulator was developed and experimentally verified. The system integrates VR controllers, stereoscopic vision, inverse kinematics, and communication with the physical rover. The developed solution was successfully implemented on the Modernity III rover and was also used during the 2021 Virtual Rover Challenge.

The proposed VR-based interface was compared with the conventional joystick-based control system in preliminary tests performed on the physical manipulator. Within the limited set of preliminary trials, the average task-completion time decreased from approximately 14.99 s for conventional control to 12.00 s for VR-based control, corresponding to an approximately 20% reduction. The recorded gripper trajectories also indicated smoother motion and a shorter path for the VR-based control method. Due to the limited number of trials, these results should be interpreted as a preliminary validation of the proposed concept rather than a complete statistical evaluation.

The computational performance of the system was evaluated using implementations developed in Unity 2020.3 LTS and Unreal Engine 4.26. For the considered VR manipulator application, Unreal Engine showed lower CPU and GPU usage on both tested computers. The average CPU usage decreased from 33.22% to 20.95% on Computer I and from 66.45% to 46.59% on Computer II when Unreal Engine was used instead of Unity. A similar tendency was observed for GPU usage. In contrast, Unity required less or comparable RAM, while the FPS results did not indicate a clear overall advantage of either engine.

The obtained results show that the selection of a graphics engine should depend on the available hardware resources and the requirements of the specific VR application rather than on a single performance indicator. The developed control architecture can also be adapted to other mobile or stationary robotic platforms equipped with manipulators.

Future work should include a more extensive experimental evaluation involving a larger number of operators and repeated trials. Further optimization of the computational and communication components of the system may also improve the performance of the developed VR-based control architecture.

ACKNOWLEDGMENT AND FUNDING

This work has been carried out within statutory research of the Department of Mechanics and Machine Design Fundamentals of Czestochowa University of Technology. This research received no external funding.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

REFERENCES

- [1] H. Yang, Application technology of space manipulator, In *Manned Spacecraft Technologies*, Springer Singapore, 2021, 295–309.
- [2] L. G. Teoh, Y.-J. Tseng, and C.-S. Chen, Applying virtual reality to the control system for status judgment and arrangement of workpiece, *2019 IEEE International Conference on Computation, Communication and Engineering (ICCCE)*, Fujian, China, 2019, 40–43.
- [3] R. Li, M. van Almkerk, S. van Waveren, E. J. Carter, and I. Leite, Comparing human-robot proxemics between virtual reality and the real world, *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, Daegu, South Korea, 2019, 431–439.
- [4] L. Gammieri, M. Schumann, L. Pelliccia, G. D. Gironimo, and P. Klimant, Coupling of a redundant manipulator with a virtual reality environment to enhance human-robot cooperation, *Procedia CIRP*, 62, 2017, 618–623.
- [5] P. Ptak, K. Sokół, and D. Cekus, Design of the manipulator control unit for modernity mars rover, *Engineering Mechanics 2019*, Prague, 2019, 299–302.
- [6] N. Sreekanth, A. Dinesan, A. R. Nair, G. Udupa, and V. Tirumaladass, Design of robotic manipulator for space applications, *Materials Today: Proceedings*, 46, 2021, 4962–4970.
- [7] M. Pierzgalski, P. Ptak, D. Cekus, and K. Sokół, Modeling and stress analysis of a manipulator mounted on a mars rover, *Procedia Engineering*, 177, 2017, 121–126.
- [8] D. Skrobek and D. Cekus, Optimization of the operation of the anthropomorphic manipulator in a three-dimensional working space, *Engineering Optimization*, 51, 2019, 1997–2010.
- [9] T. Szczęsny and M. Rečko, Control of robotic arm for mars rover analogue, *2018 19th International Carpathian Control Conference (ICCC)*, Szilvásvárad, Hungary, 2018, 259–264.
- [10] A. P. Buczyłowski, M. Rečko, A. Wolniakowski, and K. Haraburda, A hand-held teleoperation device for the control of the mars rover analogue manipulator, *International Journal of Mechanical Engineering and Robotics Research*, 9, 2020, 1018–1023.
- [11] I. Kubik and P. Kwiaton, Forward kinematics algorithm for anthropomorphic manipulators, *Prace Naukowe Akademii im. Jana Długosza w Częstochowie. Technika, Informatyka, Inżynieria Bezpieczeństwa*, 5, 2017, 39–51.
- [12] P. Lu, Z.-W. Ren, and K.-T. Zhang, Control system design and kinematic analysis for a 4-dof light weight manipulator, *2017 Chinese Automation Congress (CAC)*, Jinan, China, 2017, 5673–5678.
- [13] F. Roperio, P. Muñoz, M. D. R-Moreno, and D. F. Barrero, A virtual reality mission planner for mars rovers, *2017 6th International Conference on Space Mission Challenges for Information Technology (SMC-IT)*, California, USA, 2017, 142–146.
- [14] M. F. Alam, S. Katsikas, O. Beltramello, and S. Hadjiefthymiades, Augmented and virtual reality based monitoring and safety system: A prototype IoT platform, *Journal of Network and Computer Applications*, 89, 2017, 109–119.
- [15] N. Zhang, T. Qi, and Y. Zhao, Real-time learning and recognition of assembly activities based on virtual reality demonstration, *Sensors*, 21, 2021, 6201.
- [16] G. Garg, V. Kuts, and G. Anbarjafari, Digital twin for FANUC robots: Industrial robot programming and simulation using virtual reality, *Sustainability*, 13, 2021, 10336.
- [17] L. Pérez, S. Rodríguez-Jiménez, N. Rodríguez, R. Usamentiaga, and D. F. García, Digital twin and virtual reality based methodology for multi-robot manufacturing cell commissioning, *Applied Sciences*, 10, 2020, 3633.
- [18] O. Liu, D. Rakita, B. Mutlu, and M. Gleicher, Understanding human-robot interaction in virtual reality, *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, Lisbon, Portugal, 2017, 751–757.
- [19] A. A. Malik, T. Masood, and A. Bilberg, Virtual reality in manufacturing: Immersive and collaborative artificial-reality in design of human-robot workspace, *International Journal of Computer Integrated Manufacturing*, 33, 2020, 22–37.
- [20] X. Sha, Z. Jia, W. Sun, Y. Hao, X. Xiao, and H. Hu, Development of mixed reality robot control system based on HoloLens, In *Intelligent Robotics and Applications*, Springer International Publishing, 2019, 571–581.
- [21] B. Bejczy, R. Bozyil, E. Vaičekas, S. B. K. Petersen, S. Bøgh, S. S. Hjorth, and E. B. Hansen, Mixed reality interface for improving mobile manipulator teleoperation in contamination critical applications, *Procedia Manufacturing*, 51, 2020, 620–626.
- [22] D. Cafolla, M. Russo, and M. Ceccarelli, Experimental validation of HeritageBot III, a robotic platform for cultural heritage, *Journal of Intelligent and Robotic Systems*, 100, 2020, 223–237.
- [23] T. Namba and Y. Yamada, Risks of deep reinforcement learning applied to fall prevention assist by autonomous mobile robots in the hospital, *Big Data and Cognitive Computing*, 2, 2018, 13.
- [24] A. Khadr and A. Houidi, Designing digital twin of a three wheeled robot using simscape for trajectory tracking, *Applications of Modelling and Simulation*, 10, 2026, 163–172.
- [25] A. Alkamachi, Adaptive pd-based control of robot manipulators with gravity compensation and rbf neural networks, *Applications of Modelling and Simulation*, 10, 2026, 108–119.

- [26] B. R. Galarza, P. Ayala, S. Manzano, and M. V. Garcia, Virtual reality teleoperation system for mobile robot manipulation, *Robotics*, 12, 2023, 163.
- [27] M. Gallipoli, S. Buonocore, M. Selvaggio, G. A. Fontanelli, S. Grazioso, and G. D. Gironimo, A virtual reality-based dual-mode robot teleoperation architecture, *Robotica*, 42, 2024, 1935–1958.
- [28] A. George, A. Bartsch, and A. B. Farimani, OpenVR: Teleoperation for manipulation, *SoftwareX*, 29, 2025, 102054.
- [29] D. Skrobek, *Modeling, Analysis and Optimization of the Work Cycle of Manipulators with Four Degrees of Freedom*, Ph.D. Thesis, Czestochowa University of Technology, 2021.
- [30] G. Ptak, *The Use of Virtual Reality to Control the Mars Rover's Manipulator*, M.Sc. Thesis, Czestochowa University of Technology, 2021.
- [31] Blender. <https://www.blender.org/>, (accessed 28 august 2026).
- [32] Unity. <https://unity.com/>, (accessed 28 august 2026).
- [33] OpenCV documentation. <https://docs.opencv.org/>, (accessed 28 august 2026).
- [34] Unreal engine. <https://www.unrealengine.com/>, (accessed 28 august 2026).
- [35] Y. Sahraoui, A. Ghanam, S. Zaidi, S. Bitam, and A. Mellouk, Performance evaluation of TCP and UDP based video streaming in vehicular ad-hoc networks, *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, El Qued, Algeria, 2018, 67–72.
- [36] Z. Zhong and W. Hu, Error detection and control of IIoT network based on CRC algorithm, *Computer Communications*, 153, 2020, 390–396.
- [37] A. Ciekankowska, A. K. Gliński, and K. Dziedzic, Comparative analysis of Unity and Unreal Engine efficiency in creating virtual exhibitions of 3d scanned models, *Journal of Computer Sciences Institute*, 20, 2021, 247–253.
- [38] E. Christopoulou and S. Xinogalos, Overview and comparative analysis of game engines for desktop and mobile devices, *International Journal of Serious Games*, 4, 2017, 21–36.