Virtual Reality-Based Control and Simulation of a Real Robotic Arm

Eda Derya Toper, Fırat Kaçar, Serap Cekli

**Electrical-Electronics Engineering Department, Istanbul University - Cerrahpasa, Istanbul, Turkey*
(edaderya.toper@iuc.ogr.edu.tr, fkacar@iuc.edu.tr, serap.cekli@iuc.edu.tr)

Abstract: Industry 4.0 and digital factories leverage key technologies such as virtual reality (VR) to design, simulate, optimise, and interact with physical production systems remotely or collaboratively. Recent advancements in industrial and aerospace applications demonstrate that integrating VR with versatile robotic control techniques can unlock new opportunities. VR offers an "extended arm" for physical environments by providing a user-friendly human-machine interface, enhancing sustainability, testability, and functionality in robotic control systems. This research focuses on a widely used and continuously evolving robotic arm, designed to operate in an integrated system that functions in both real and virtual environments. A 3-degree-of-freedom (3DOF) robotic arm was initially developed and later upgraded to a 5DOF system. The robotic arm is integrated with Virtual Reality (VR) interfaces based on the Robot Operating System (ROS) and Unity3D platforms. In the real-world scenario, inverse kinematics is applied for the robot's movement mechanism, while gradient descent algorithms are used for the virtual model. Forward kinematics using homogeneous transformation matrices has been employed to verify the accuracy of the robotic arm's movements. VR headsets and controllers are used to direct the virtual control of the robotic arm. The goal of this study is to develop integrated systems that enable control of real-world operations via remote observation, addressing challenges such as prototyping costs and safety risks. By offering an innovative approach to robotic arm control through VR integration, this work contributes to the advancement of Industry 4.0 technologies. The findings suggest potential applications across various industries, opening up new possibilities for remote operation and training in complex manufacturing environments.

Keywords: Robotic arm, Virtual reality, Robot operating system, Kinematics, Robot manipulator.

1. INTRODUCTION

The concept of Industry 4.0, introduced at the 2011 Hannover Fair, emphasises machine-to-machine communication and human-machine collaboration, allowing robots to become an integral part of production processes through real-time data exchange. Today, robots and robotic arms with varying degrees of freedom (DOF) are at the centre of production processes in many sectors. Initially used in industrial applications in the 1960s to undertake hazardous and risky tasks, robots (Mikolajczyk et al., 2022) have since taken on important roles in sectors such as healthcare, agriculture, shipbuilding, and construction, alongside the rapid development of computer control and artificial intelligence (Lin et al., 2020).

Industrial robots not only enhance product quality but also reduce costs. However, users who are unfamiliar with a robot's purpose and goals may feel uneasy even with safe robots. Furthermore, when considering the safety risks and installation costs of designing and implementing robotic production lines, the importance of robot simulation becomes clearer. The "digital twin" model (Grieves, 2016), introduced in 2002 by the University of Michigan, optimises processes by creating digital copies of physical systems. Virtual reality technology facilitates critical aspects such as sustainability, testability (Bolano et al., 2019), and optimisation in robot control systems, while also paving the way for new user-friendly human-machine interfaces (Freund and Rossmann, 1999). This technology enables the development of new, user-friendly interfaces for human-robot interaction (HRI) and human-robot collaboration (HRC) (Freund and Rossmann, 1999; Dianatfar et al., 2021).

Studies in the field of VR and collaborative robot control, similar to our research, aim to offer a new perspective on virtual reality and collaborative robot control. In the literature, several studies can be found, including investigations into cloud-based robotic systems and multi-user interactions (Mizuchi and Inamura, 2017), long-distance data communication between ROS and Unity (Codd-Downey et al., 2014), co-simulation and interaction between digital twins and virtual reality (Havard et al., 2019), system optimization and monitoring (Sita et al., 2017), optimization of assembly processes (Lin et al., 2020; Marzano et al., 2015), and the VR-based management of multi-robot systems (Roldán et al., 2018). Although each study focuses on different aspects, they all serve as forward-looking examples. Their goal is to reduce costs and time, improve safety, and provide a user-friendly interface without intervening with existing real-world systems. It is clear that VR techniques hold great potential for programming and remotely controlling robotic systems in the future.

In the study (Islam et al., 2014), a mathematical model and design of a robust control strategy for a multi-degree-of-freedom (DOF) manipulator are investigated. Kinematic and dynamic models of the robotic arm have been derived, and the model has been validated by simulation. Also, in (Dobiš et al., 2021), two motion planning approaches for industrial manipulators are offered. Some metrics of both designs were measured and presented.

This study aims to develop a system that integrates both virtual and real environments using a model created in both. The chosen model is a robotic arm, a widely used mechanism that continues to evolve today. The goal is to enable users of robot systems, particularly those requiring an operator, to work remotely. This would allow for tasks that traditionally require on-site supervision to be performed remotely. Additionally, conducting tests, trials, or experimental work in a virtual environment aims to reduce costs, time, and safety risks associated with creating physical prototypes.

The rest of the paper is organised as follows. Section 2 gives the basic principles and methodology. Section 3 explains the assembly, calibration, and forward and inverse kinematics calculations of the robot manipulator design. Section 4 underlines the kinematic calculations, control methods, and how data communication with the real world is achieved for the robot manipulator created in the virtual environment. Section 5 and Section 6 deal with the interaction between the motion of the robot arm and the virtual world. Besides, these sections discuss the 5DOF real robot manipulator and virtual environment, respectively. Finally, the results are discussed in Section 7.

2. OVERVIEW AND METHODOLOGY

This study aims to integrate robot manipulators in both real and virtual environments. First, the design, the programs used, and the working setup for a 3 Degrees of Freedom (3DOF) robot manipulator (Kim and Song, 2014) are discussed in detail. Then, a model with 5 Degrees of Freedom (5DOF) has been developed to perform tasks that are more complex and increase the system's flexibility.

This work is the product of a master's thesis study, primarily focused on examining the bidirectional interaction and control mechanisms between virtual and real environments. For this purpose, a standard articulated robotic arm topology, widely used in the literature, was adopted. Even the upgrade of the robotic arm from 3DOF to 5DOF aimed to increase the system's flexibility while remaining within this standard topology. Therefore, the core contribution of this study is detailing the process and results of successfully integrating virtual environment control (Unity/Gradient Descent) with real-world kinematic control (Arduino/Inverse Kinematics) using this common topology. The physical manipulator's mechanical design was optimised to minimise friction, allowing the study to focus on kinematic control and virtual-real integration challenges.

The main reason for choosing this methodology is to ensure seamless interaction between real and virtual environments and to create a remotely controllable robotic system. The 3DOF model serves as an ideal starting point for understanding the basic principles and testing system integration, while the 5DOF model is used to simulate more complex movements and tasks. The workflow of the study is shown in Figure 1.

According to the working principle, the user employing virtual reality consoles first moves the target object in the virtual environment. The end-effector of the virtual robot manipulator moves according to the gradient descent algorithm and operates to capture the target object. When the virtual robot manipulator captures the target object, the coordinate data is transmitted to the real robot manipulator via serial communication. The real-world robot manipulator then moves based on the received coordinates using the inverse kinematics algorithm. In the following sections, a detailed explanation of each step and the results obtained will be presented.

3. ROBOT MANIPULATOR MODEL AND KINEMATICS IN THE REAL WORLD

Fig. 1. Workflow diagram.

This section will detail the design, assembly, calibration, and forward and inverse kinematics calculations of the robot manipulator built in a real environment. Figure 2 shows the 3DOF manipulator position when the joint angles are set to 0° in the real world.

The design of the real-world robot manipulator model has been done using Autodesk Fusion 360. This software has been chosen due to its simple interface and ease of use. The manipulator parts have been printed with the help of a 3D printer. To control the motors used in the manipulator's axes, Arduino Mega and Arduino Due development boards are utilised. For this reason, the Arduino IDE software interface is used for the calibration and inverse kinematics calculations. For forward kinematics calculations and other mathematical operations, such as matrices, the Python programming language is used.

Firstly, the real-world robot manipulator model has been calibrated, and its extended workspace has been determined. Calibration is crucial to ensure that the robot arm is not mechanically overloaded and that the motors reach the desired coordinates at their start and end positions. At this stage, calibration codes have been developed to observe the rotation directions and positions of the axles attached to the motor heads that form the joints of the robot.

Fig. 2. Position of 3DOF manipulator at θ_1=0°, θ_2=0°, θ_3=0° joint angles.

For servomotors, providing values in microseconds rather than angles makes it easier to align with the duty cycle values found in the motor's datasheet. This ensures more accurate and precise movements. Expressing the pulse values of the angles in microseconds algebraically on an x-y coordinate system, we obtain (1). The X-axis represents the maximum and minimum angle values, while the Y-axis shows the corresponding pulse widths. This formula calculates the PWM signals corresponding to specific angles of the servomotors, allowing the motors to perform movements based on these signals.

$$Y - Y_1 = \frac{Y_2 - Y_1}{X_2 - X_1}(X - X_1) \tag{1}$$

The following table summarises the angle and pulse values for the servomotors. Based on the working area, the angles of the third servo motor were set between -90° and +90°. These limits will not affect the kinematic diagrams when the joint angles are at (0,0,0). The algebraic Eq. (1) was processed into the Arduino calibration code according to Table 1.

Table 1. Motor angles and pulse equivalents.

Motor - Joint	Minimum Angle	Pulse Equivalent	Maximum Angle	Pulse Equivalent
Servo Motor θ_1	0°	450	180°	2950
Servo Motor θ_2	0°	450	180°	2450
Servo Motor θ_3	-90°	450	90°	2450

3.1 Forward Kinematic Calculations and Code Integration

Figure 3 shows the kinematic diagram of the 3DOF manipulator that was created. The kinematic diagram (Spong et al.) demonstrates how the links and joints are connected when the joint angle values are 0°. In robot manipulator applications, the position and orientation (rotation) of the end-effector are crucial. When a servomotor moves, it alters both the position and rotation of the end-effector. Forward kinematics (Kuçuk and Bingul, 2004) is the process of determining the position and orientation of the robot's end-effector based on the joint parameters (angles or displacements). The positions of the end-effector are expressed through "Displacement Vectors (Bajd et al., 2010)," denoted as "d". The rotation (orientation) of the end-effector is represented by the "Rotation Matrix (Hock et al., 2014)," which describes how a point defined in one frame appears in another, and is denoted by "R".

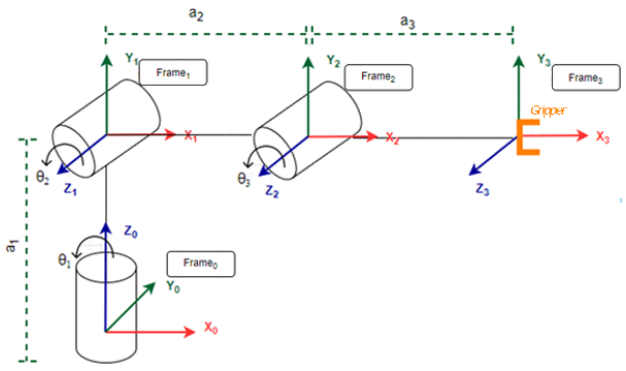


Fig. 3. Kinematic diagram of the 3DOF manipulator.

Homogeneous Transformation Matrices used for forward kinematic calculations. A Homogeneous Transformation Matrix (HTM) [17] is a 4×4 matrix that expresses both the position and orientation of an object in a single frame within 3D space. These matrices contain both rotational (R) and translational (d) information and are denoted by T or H. To find the Homogeneous Transformation Matrices, the relationships between the frames, as shown in the kinematic diagram of the manipulator, are examined. The HTM is derived using rotation matrices and displacement vectors as follows:

$$R_1^0 = \begin{bmatrix} cos\theta_1 & -sin\theta_1 & 0 \\ sin\theta_1 & cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos90 & -sin90 \\ 0 & sin90 & cos90 \end{bmatrix} = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 \\ sin\theta_1 & 0 & -cos\theta_1 \\ 0 & 1 & 0 \end{bmatrix} \quad d_1^0 = \begin{bmatrix} 0 \\ 0 \\ a_1 \end{bmatrix}, \tag{2}$$

$$R_2^1 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 \\ sin\theta_2 & cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 \\ sin\theta_2 & cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad d_2^1 = \begin{bmatrix} a_2 cos\theta_2 \\ a_2 sin\theta_2 \\ 0 \end{bmatrix}, \tag{3}$$

$$R_3^2 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 \\ sin\theta_3 & cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 \\ sin\theta_3 & cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad d_3^2 = \begin{bmatrix} a_3 cos\theta_3 \\ a_3 sin\theta_3 \\ 0 \end{bmatrix}. \tag{4}$$

In this study, HTM was also calculated using Denavit-Hartenberg (D-H) (Hayat et al., 2013) parameters. The Denavit-Hartenberg method is widely used in robot kinematics and describes the general transformation between two joints using four parameters (Spong et al., 2005; Hayat et al., 2013). According to the D-H Parameter Table, the number of rows corresponds to one less than the number of manipulator frames. The columns contain the rotation parameters θ and α, and the translation parameters r and d. Based on the obtained data for the 3DOF manipulator, the D-H table is presented as shown in Table 2.

Table 1. 3DOF manipulator D - H table.

n	θ_n	α_n	r_n	d_n
1	θ_1	90°	0	a_1
2	θ_2	0	a_2	0
3	θ_3	0	a_3	0

The general representation of the D-H method within the Homogeneous Transformation Matrix (HTM) is as follows:

$$T_n^{n-1} = \begin{bmatrix} cos\theta_n & -sin\theta_n cos\alpha_n & sin\theta_n sin\alpha_n & r_n cos\theta_n \\ sin\theta_n & cos\theta_n cos\alpha_n & -cos\theta_n sin\alpha_n & r_n sin\theta_n \\ 0 & sin\alpha_n & cos\alpha_n & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

The HTM obtained using both methods is as follows:

$$T_1^0 = \begin{bmatrix} cos\theta_1 & 0 & sin\theta_1 & 0 \\ sin\theta_1 & 0 & -cos\theta_1 & 0 \\ 0 & 1 & 0 & a_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

$$T_2^1 = \begin{bmatrix} cos\theta_2 & -sin\theta_2 & 0 & a_2 cos\theta_2 \\ sin\theta_2 & cos\theta_2 & 0 & a_2 sin\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

$$T_3^2 = \begin{bmatrix} cos\theta_3 & -sin\theta_3 & 0 & a_3 cos\theta_3 \\ sin\theta_3 & cos\theta_3 & 0 & a_3 sin\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

As a result, the HTM T_3^0 for the 3DOF manipulator was obtained as follows:

$$T_3^0 = T_1^0\, T_2^1\, T_3^2 \quad (9)$$

After completing the necessary adjustments through calibration and mechanical assembly, the maximum and minimum points that the robotic arm could reach along the X, Y, and Z axes were examined to define its workspace. For this purpose, the HTM calculations were utilised. The dimensions of each axis (link) of the physical robotic arm model (a_1, a_2, a_3) were gauged, and these data were integrated into the Python software used for forward kinematics calculations. The end-effector position data obtained from HTM calculations at specific angles are compared with the movements of the real robotic manipulator. An example study was conducted to observe the consistency between the calculations and practical results shown in Figure 4.

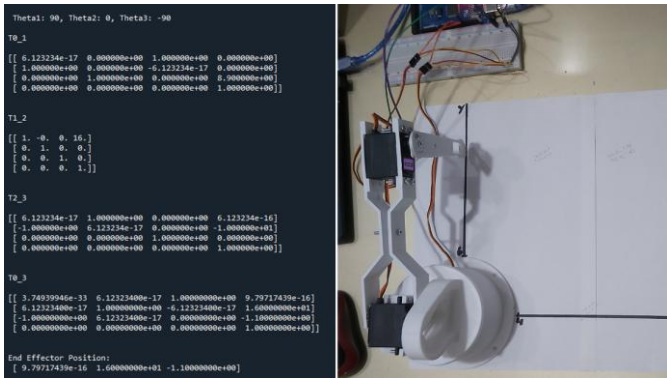


Fig. 4. The result of the HTM and the position of the robot manipulator at angles θ_1=90°, θ_2=0, θ_3=-90°.

3.2. Inverse Kinematic Calculations and Code Integration

In robotic systems, the problem of determining the position and orientation of the end-effector given the joint angles is known as "forward kinematics". On the other hand, determining the required joint angles for the robot to reach a given end-effector configuration (D'Souza et al., 2001) is referred to as the "inverse kinematics (Lee and Ziegler, 1984)" problem. In inverse kinematics calculations, unlike in forward kinematics, the direction of the X, Y, and Z axes is not significant. The inverse kinematics diagram can be drawn in two different perspectives:

• Top View – The XY plane as seen from the Z-axis (a top-down perspective).

• Side View – The space that includes the Z-axis.

For the top view of our robotic arm, to avoid errors in the inverse kinematics calculations, the angle θ_1 is considered within the range $0 < \theta_1 < 90°$, as depicted in Figure 5. As seen, the top view does not affect angles θ_2 and θ_3. Here, only the angle θ_1 is computed:

$$\tan\theta_1 = \frac{Y_3^0}{X_3^0} \quad \theta_1 = \tan^{-1}\frac{Y_3^0}{X_3^0} \quad (10)$$

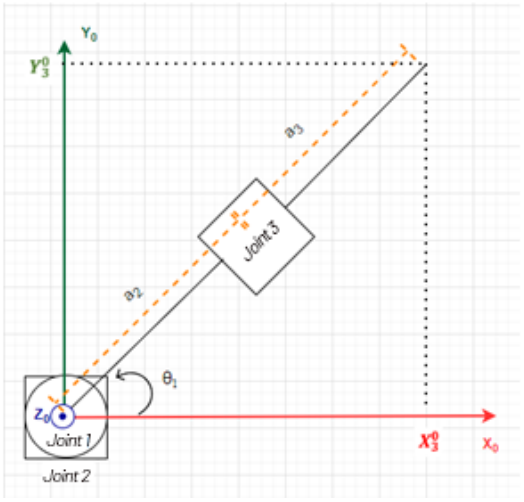


Fig. 5. 3DOF Manipulator top view.

In the side view of the robot arm reaching the target position, as shown in Figure 6a, two different configurations emerge. That is, depending on the values of angles θ_2 and θ_3, the robot arm can reach the target with the elbow positioned either above or below. The "elbow up" configuration is the most suitable for the robotic arm model and workspace used in this study. Therefore, inverse kinematic calculations were carried out considering $0 < \theta_2 < 90°$ and $-90° < \theta_3 < 0$ ranges, as shown in Figure 6b.

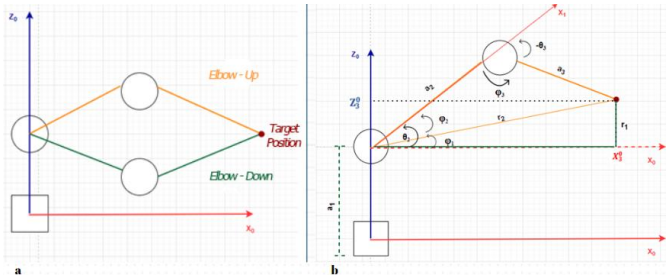


Fig. 6. 3DOF Manipulator side view.

The equations for θ_2 and θ_3 angles based on the kinematic diagram are obtained as follows

$$\theta_2 = \varphi_1 + \varphi_2 \quad (11)$$

$$r_1 = Z_3^0 - a_1 \quad (12)$$

$$\varphi_1 = \tan^{-1}\frac{r_1}{X_3^0},\ r_2 = \sqrt{r_1^2 + (X_3^0)^2} \quad (13), (14)$$

The values for angles φ_2 and φ_3 are calculated using the cosine theorem:

$$\varphi_2 = \cos^{-1}\frac{a_2^2 + r_2^2 - a_3^2}{2a_2 r_2} \quad (15)$$

$$\varphi_3 = \cos^{-1}\frac{a_2^2+a_3^2-r_2^2}{2a_2a_3} \quad (16)$$

$$\theta_3 = \varphi_3 - 180 \quad (17)$$

To enable the robot arm to calculate the values of θ_1, θ_2, and θ_3 required to reach desired target positions, the derived equations have been incorporated into the Arduino IDE program as code. The end-effector position data obtained using the Homogeneous Transformation Matrix (HTM) method has been entered into the software for the robot manipulator's inverse kinematic calculations, corresponding to the X, Y, and Z values. The expected position that the end-effector of the robot manipulator is supposed to reach, based on inverse kinematics, has been observed with the given joint angles.

An important aspect of this study is the use of the "elbow-up" method in the inverse kinematic calculations. This choice limits the workspace, ensuring that the θ_3 angle has a maximum of 0 degrees and a minimum of -90°. If this constraint is ignored, discrepancies will visibly arise in the forward and inverse kinematic tests. Figure 7 illustrates that different joint angles can be utilised to reach a similar target position according to the HTM method. When the elbow-up method is applied in the inverse kinematic calculations, the joint angles obtained are $\theta_1 = 40°$, $\theta_2 = 70°$, and $\theta_3 = -15°$. If the elbow-down method had been used, these angle values would have been $\theta_1 = 40°$, $\theta_2 = 60°$, and $\theta_3 = 10°$.

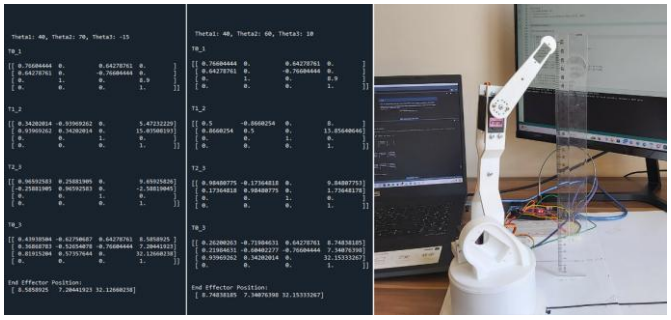


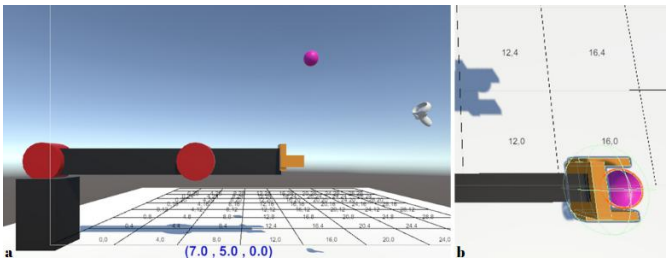
Fig. 7. The HTM end-effector position and robot movement at angles (40°, 70°, -15°) and (40°, 60°, 10°).

4. ROBOT MANIPULATOR MODEL AND CONTROL IN THE VIRTUAL ENVIRONMENT

This section details the design, kinematic calculations, control methods, and how data communication with the real world is achieved for the robot manipulator created in the virtual environment.

The design of the manipulator in the virtual environment was created using Unity3D (Lin et al., 2020; Grieves, 2016). The kinematic calculations for the manipulator were coded in C# within Unity3D. Since a comprehensive model was not required for the manipulator in the virtual environment, ready-made components provided by Unity3D have been used. The dimensions of the virtual manipulator were set to be $^1/_4$ times the real manipulator's size to allow for easier control and movement. This adjustment was made because Unity operates with units measured in meters, whereas the dimensions of the real-world robotic arm are computed in centimetres or millimetres.

In Figure 8a, the position of the virtual robot manipulator is shown with joint angles at $\theta_1 = 0$, $\theta_2 = 0°$, and $\theta_3 = 0°$. The virtual robotic arm is designed with three joints, matching the real-world model. Initially, a grid was created in the Unity scene using C# to visualise the position of the end-effector clearly. The pink sphere (target object) moved using a virtual reality console, and the virtual robot manipulator, controlled by the gradient method, attempts to reach this target.



a **b**

Fig. 8. a. Position of 3DOF manipulator at θ_1=0°, θ_2=0°, θ_3= 0° joint angles, b. End-effector target interaction area.

As shown in Figure 8b, once the target object enters the interaction area of the robot arm's end-effector, the end-effector's coordinate information is transmitted to the real-world robotic arm. Before these coordinates were sent, they were adjusted through the necessary transformations to ensure compatibility with the real-world system. Since the coordinate systems in the real world and the virtual environment differ, it is crucial to apply the correct transformations during data transmission and reception. Unity3D uses a left-hand coordinate system, while the real world typically follows the right-hand rule. The difference between these two systems necessitates transforming the X, Y, Z coordinates in the virtual world into X, Z, Y for proper alignment when transmitted to the real world.

4.1. Robot Manipulator Motion Mechanism and Code Integration

In the simulation, the Gradient Descent Algorithm (Sharma, 2018) was used for geometric calculations, unlike the real-world robot arm. This method was chosen because there is a target in the simulation, and the robot arm tries to reach this target, effectively. This algorithm aims to minimise the distance between the robot arm's end-effector and the target. It continuously forces the robotic manipulator to move, keeping the distance between the end-effector and the target object as minimal as possible.

The Gradient Descent Algorithm works by optimising the angle of each joint in the robot arm to minimise the distance to the target position (Oke and Istefanopulos, 2001). By iteratively updating each joint's angle using gradient information, the algorithm attempts to minimise the error function. This process ensures that the robotic arm reaches the designated target position as quickly and accurately as possible.

Typically, when calculating the gradient for each joint, the derivative of the position error with respect to the relevant joint angle is taken. However, calculating the derivative requires the function to meet specific mathematical properties, which are

not always guaranteed for arbitrary problems. This situation complicates what might otherwise be considered a simple problem, and obtaining the actual derivative can be impossible. Therefore, in this study, making a rough estimate of the value is considered essential.

For a better understanding of gradient calculations, a manipulator consisting of one joint and one end-effector, along with a target, was created as shown in Figure 9.

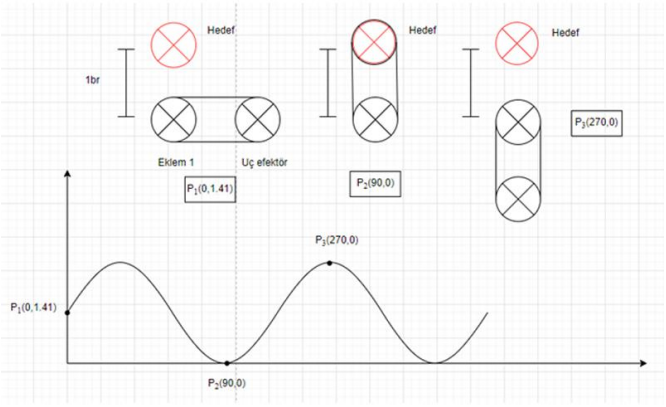


Fig. 9. Diagram for gradient calculation of the virtual 3DOF manipulator.

In Figure 9, the lengths are expressed in units because this is how length definitions are represented in the Unity3D program used for the simulation. The distance between the target and the end effector is illustrated in the visual to help better understand the gradient calculation (De Luca and Oriolo 1990).

The slope calculations were performed using a rough estimation method to optimise the function's derivative between 0° and 90°. As seen in the Figure, the distance increases again once the angle exceeds 90°. Therefore, slope calculations were made for angle values between 0° and 90°. The slope approximation is calculated based on the points $P_1(0, 1.41)$ and $P_2(30, 1)$, as shown in the equation. The general formula for the approximate slope (Gradient (De Luca and Oriolo 1990)) calculation:

$$\nabla f(p) = \frac{f(p+\Delta x) - f(p)}{\nabla x} \quad (18)$$

The slope formula applied in our study:

$$\nabla f(\theta) = \frac{f(\theta+\epsilon) - f(\theta)}{\epsilon} \quad (19)$$

Gradient descent update step:

$\theta = \theta - \alpha \cdot \nabla f(\theta)$ $f(\theta)$: angle distance θ: current joint angle

$$\theta = \theta - \alpha \frac{f(\theta+\epsilon) - f(\theta)}{\epsilon}$$ ϵ: small step size α: learning rate (20)

In (18-20), $f(\theta)$ represents the angle distance, and θ is the current joint angle. The value of ϵ, expressed as the small step size, was set to 0.01 in the software. The epsilon value (ε=0.01) was determined through empirical testing. Values ranging from 0.001 to 0.1 were tested. Smaller values (ε<0.005) resulted in slower convergence times (>5 seconds), while larger values (ε>0.05) caused oscillations around the target position. The selected value of 0.01 provided an optimal balance between convergence speed (average 2.3 seconds) and positioning accuracy (±0.03 units). The value of α, which represents the learning rate, is assigned as 0.1. In the Unity C# code, the red cylindrical objects, referred to as "joints" in the software, are assigned based on the gradient slope calculation to facilitate movement.

4.2 Control Tools and Communication Settings in the Virtual Environment

Due to the current lack of widespread use of virtual reality, experimental studies have been conducted using mobile accelerometers. Figure 10 illustrates the control of the target object using a mobile accelerometer and the corresponding movements of the robotic manipulators.

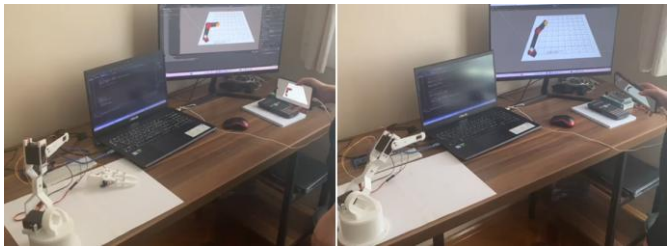


Fig. 10. Movement of 3DOF Manipulator with mobile accelerometer.

As shown, to enable movement along the axes with the mobile accelerometer, it is necessary to tilt the mobile phone.

Since virtual reality consoles are devices specifically created for virtual environments, the coordinate systems are designed to be compatible. Movement to the left or right occurs on the X-axis, up and down on the Y-axis, and forward and backwards on the Z-axis. This not only provides ease of use but also facilitates movement. The software control tools that come with the installation of XR packages were used, and no additional work was necessary. The basic settings are as shown in Figure 11.

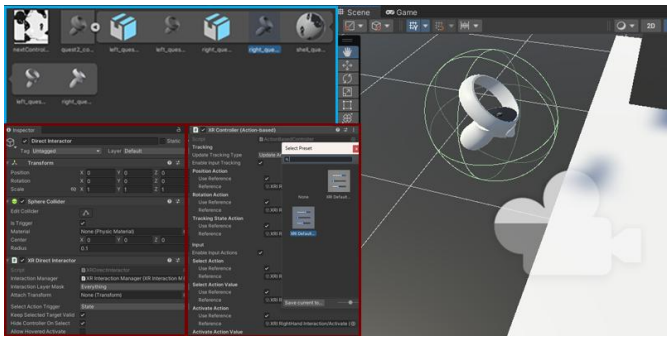


Fig. 11. Design and settings of the VR Console in the Unity program.

Communication between the robot arms in the virtual environment and the real world was achieved through serial communication in this study. The “System.IO.Ports” library is used in the Unity C# software file to establish serial communication with the Arduino development kit.

The virtual manipulator was designed at 1/4 scale of the real manipulator to facilitate easier control and movement within

the Unity environment. Therefore, the coordinates must be scaled up before transmission to the real robot:

Unity3D uses a left-handed coordinate system (X-right, Y-up, Z-forward), whereas the real-world robotic arm follows the right-handed convention (X-forward, Y-left, Z-up). This requires a coordinate axis permutation given as Eq. (21).

$$T_{virtual \to real} = \begin{bmatrix} 4 & 0 & 0 & X_0 \\ 0 & 0 & 4 & Y_0 \\ 0 & 4 & 0 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (21)$$

This 4×4 matrix combines the scaling (diagonal elements), coordinate swap (permutation of rows), and translation (last column) into a single operation. When the virtual manipulator's end-effector enters the interaction zone and captures the target object, these transformed coordinates are transmitted via serial communication to the Arduino controller, which then calculates the necessary joint angles using inverse kinematics as described in Section 3.2.

5. INTERACTIVE MOTION OF ROBOT ARMS IN THE REAL AND VIRTUAL WORLDS

The following Figure 12 and Figure 13, illustrate the movement of the robot manipulator in the real world based on the coordinate data received when the robot manipulator in the virtual environment captures the target object. The left-side image aims to show the similarity between the joint movements of the robot arms in both environments. The right-side image presents a different camera view from the virtual environment, showing the positions of both the target object and the virtual reality console. The angles taken by the robot arm joints at these positions were observed, and the accuracy of the homogeneous transformation matrices was examined.

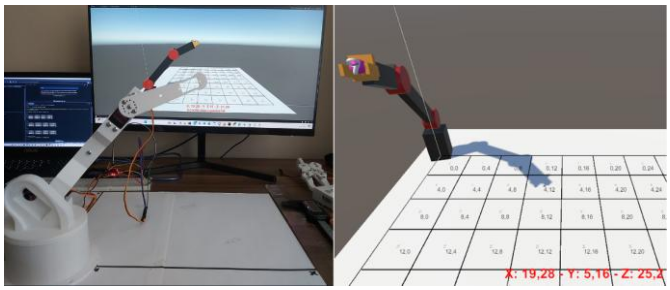


Fig. 12. Position of manipulator at θ_1=15°, θ_2=45°, θ_3= -15° joint angles.

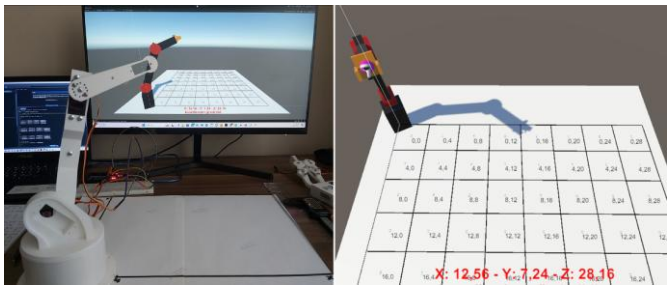


Fig. 13. Position of manipulator at θ_1=30°, θ_2=70°, θ_3=-45° joint angles.

The application was tested using Meta Quest 2 virtual reality glasses, and the visuals were captured from the Unity program screen to ensure clear and high-quality imagery.

The manipulator in both the real world and the virtual environment is tested within its workspace. During the tests, the angle values entered using the forward kinematics method were compared with those taken in both environments. For the real-world robot arm, the angle values calculated using the inverse kinematics method were taken, though for the virtual environment, the angle values computed using the gradient descent method were used.

Similarly, the coordinate data assessed using the forward kinematics method was compared with the coordinate data taken in both environments. Measurement tools are used to gauge the coordinates of the robot arm in the real world, while the position of the robot arm in the virtual environment is taken using the created grid system.

Figure 14 compares the angle values used in the forward kinematics (HTM) calculation of the robot arm's movement with the angle values obtained from the inverse kinematics method in the real world. It presents the input values for θ_1, θ_2, and θ_3 used for forward kinematics and compares them with the measured values using inverse kinematics.

Figure 15 compares the angle values used in the forward kinematics (HTM) calculation of the robot arm's movement with the angle values obtained using the gradient method in the virtual environment. It presents a comparison between the input and taken values for θ_1, θ_2, and θ_3.

The graph in Figure 16 shows that the coordinates obtained using the forward kinematics method (HTM) were compared with the accuracy of the robot arm's position as it moved according to these coordinates using the inverse kinematics in the real environment. The accuracy of the entered and obtained X, Y, and Z positions was assessed.

The graph in Figure 17 shows that the coordinates obtained using the forward kinematics method (HTM) were compared with the accuracy of the robot arm's position as it moved according to these coordinates in the virtual environment. The accuracy of the entered and gauged X, Y, and Z positions was assessed.

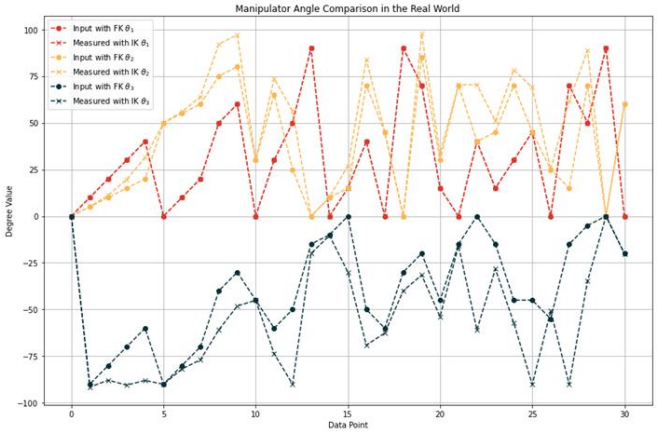


Fig. 14. Comparison of entered and measured angles for the manipulator in the real environment.

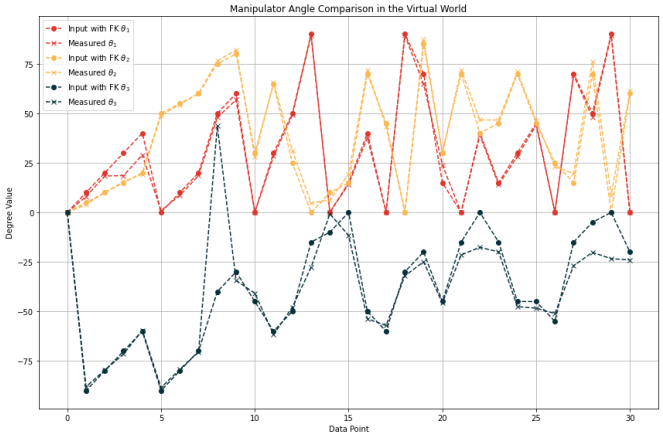


Fig. 15. Comparison of entered and measured angles for the manipulator in the virtual environment.

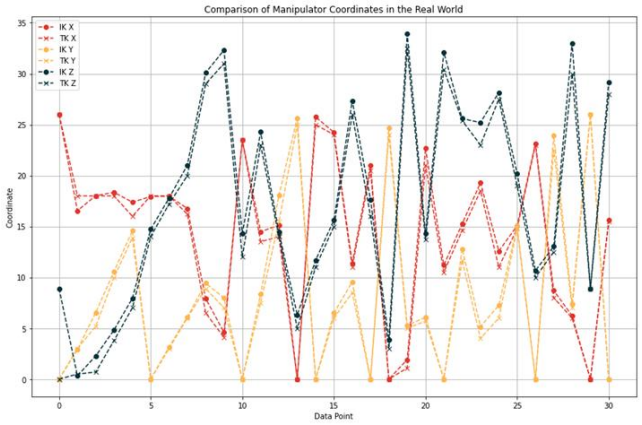


Fig. 16. Comparison of entered and measured coordinates for the manipulator in the real environment.

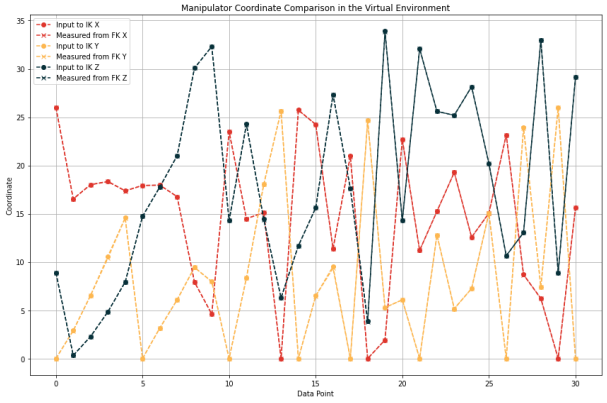


Fig. 17 Comparison of entered and measured coordinates for the manipulator in the virtual environment.

6. 5DOF ROBOT MANIPULATOR IN THE REAL AND VIRTUAL ENVIRONMENT

The 5-degree-of-freedom (5DOF) robot manipulator is designed with two additional joints to provide flexibility suited to the model's purpose. The fourth and fifth joints move within a 0°-180° angle range. The fifth joint, which is added for the gripping function, moves angularly when it reaches a specific position. To meet the requirements of the working model used here, sensors such as accelerometers, limit switches, and Hall Effect sensors could be used to ensure more precise and reliable movements. However, sensor usage was not deemed necessary in this study.

The necessary calibration adjustments, workspace determination, forward kinematic calculations, and inverse kinematic calculations were performed in the same manner as described for the 3DOF robot manipulator. For the robot manipulator in the virtual environment, only the design part was modified, as the software part had already been configured to accommodate the possibility of additional joints. After making the necessary adjustments, it was observed that the system worked smoothly. The following Figure 18 shows the positions of the robot manipulators at different angles.

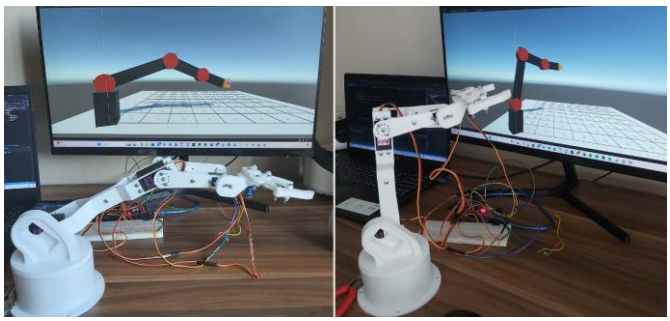


Fig. 18. The position of $(20°, 20°, -40°, 0, 50°)$ and $(45°, 80°, -90°, 10°, 100°)$.

7. CONCLUSIONS

Inverse kinematic calculations for robotic arms can vary depending on the model, and certain geometric configurations may present challenges in angle calculation. To improve accuracy, incorporating sensors and developing application-specific models can be beneficial. Previous research has extensively explored adaptive control mechanisms (Wang et al., 2017; Saleem et al., 2018), kinematic modelling strategies (Islam et al., 2014), and motion planning algorithms (Dobiš et al., 2021) for physical robotic systems with complex dynamic compensation and optimisation frameworks. In contrast, this study focused on the interaction between virtual and real environments using a standard articulated robotic arm topology, without requiring extensive dynamic modelling or adaptive parameter tuning. The emphasis was placed on practical implementation challenges of bidirectional virtual-real synchronisation rather than controller optimisation for a single physical system.

While VR-integrated robotic systems have been explored through various approaches - including cloud-based multi-robot collaboration (Mizuchi and Inamura, 2017), ROS-Unity teleoperation protocols (Codd-Downey et al., 2014), digital twin co-simulation (Havard et al., 2019), and virtual laboratory environments (Lin et al., 2020) - this work demonstrates continuous bidirectional real-time synchronisation between a physical manipulator and its virtual counterpart, addressing practical integration challenges not extensively covered in prior simulation-focused or cloud-based studies. The core contribution lies in tackling coordinate transformation (left-handed Unity to right-handed physical systems), scaling management (virtual meters to physical millimetres), and heterogeneous control algorithm synchronisation (gradient descent in virtual, inverse kinematics in real) for VR-based

remote operation. This study did not involve additional prototyping beyond the standard robotic arm platform.

Initially, control in the virtual environment was attempted using a mobile accelerometer, but it proved difficult for users. The study shifted to using a VR console, which was more effective, although a mouse, directional keys, or game controllers could also provide reliable control. There are dimensional differences between the virtual (in meters) and real (in millimetres or centimetres) environments, requiring the virtual space to be scaled to match the real-world working area. Additionally, discrepancies arose in movement due to differing coordinate system definitions; for instance, rotational movement followed different rules in the virtual environment compared to the real one. Coordinate graph analysis showed that the virtual manipulator was generally compatible with the real model, with deviations mostly between 0.01 and 0.03, peaking at 0.19. The real-world manipulator, however, displayed deviations of 0.5 to 3 cm. While some angle differences were noted, especially in the third joint, they deemed them acceptable as the end effector reached the correct positions.

In the virtual environment, manipulator movements were more predictable and consistent due to the absence of real-world factors like electronic heating, communication delays, mechanical friction, and motor current variability, resulting in more stable outcomes compared to the real-world manipulator.

Acknowledgements: This work was supported by Istanbul University-Cerrahpasa Research Fund with the project code FYL-2023-37183. The authors would like to thank Istanbul University-Cerrahpasa Research Fund for this financial support.

REFERENCES

Bajd, T., Mihelj, M., Lenarcic, J., Stanovnik, A., Munih, M. (2010). Homogenous transformation matrices. doi:10.1007/978-90-481-3776-3_2.

Bolano, G., Juelg, C., Roennau, A., Dillmann, R. (2019). Transparent Robot Behavior Using Augmented Reality in Close Human-Robot Interaction. *28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. doi:10.1109/ro-man46459. 2019.8956296.

Codd-Downey, R., Mojiri Forooshani, P., Speers, A., Wang, H., Jenkin M., (2014). From ROS to Unity: Leveraging Robot and Virtual Environment Middleware for Immersive Teleoperation. doi:10.1109/icinfa.2014.693-2785.

De Luca, A., Oriolo, G. (1990). The Reduced Gradient Method for Solving Redundancy in Robot Arms. *IFAC Proceedings*, Vols, doi:10.1016/s14746670(17)51725-5.

Dianatfar, M., Latokartano, J., Lanz, M. (2021). Review on Existing VR/AR Solutions in Human–Robot Collaboration. *Procedia CIRP*, pp. 407-411. 2020.05.25910.1016/j.procir.2020.05.259.

Dobiš, M. , Dekan, M., Duchoň, F., Beňo, P., Kohút, M. (2021). A Comparison of Motion Planners for Robotic Arm. *CEAI, Journal of Control Engineering and Applied Informatics*, vol. 23, no. 2 pp. 87-94.

D'Souza, A., Vijayakumar, S., Schaal, S. (2001). Learning Inverse Kinematics. *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium*. doi:10.1109/iros.2001.973374.

Freund, E., Rossmann, J. (1999). "Projective Virtual Reality: Bridging the Gap Between Virtual Reality and Robotics. *IEEE Transactions on Robotics and Automation*, 15(3), 411-422. doi:10.1109/70.768175.

Grieves, M. (2016). Origins of the Digital Twin Concept. doi:10.13140/RG.2.2.26367. 61609.

Havard, V., Jeanne, B., Lacomblez, M., Baudry, D. (2019). Digital Twin and Virtual Reality: A Co-Simulation Environment for Design and Assessment of Industrial Workstations. *Production & Manufacturing Research*, 7(1), 472-489. doi:10.1080/21693277.2019. 1660283.

Hayat, A. A., Chittawadigi, R. G., Udai, A. D., Saha, S. K. (2013). Identification of Denavit-Hartenberg Parameters of an Industrial Robot. *Proceedings of Conf. on Adv. in Robotics-AIR'13*, doi:10.1145/2506095.2506121.

Hock, O., Drgona, P., Paskala, M. (2014). Simulation Model of Adjustable Arm Using Denavit-Hartenberg Parameters. doi:10.1109/elektro.2014.6+6/847.

Islam, R.U, Iqbal J, and Khan, Q. (2014). Design and Comparison of Two Control Strategies for Multi-DOF Articulated Robotic Arm Manipulator. *CEAI, Journal of Control Engineering and Applied Informatics*, vol. 16, no. 2 pp. 28-39.

Kim, H.-S., Song, J.-B. (2014). Multi-DOF Counterbalance Mechanism for a Service Robot Arm. *IEEE/ASME Transactions on Mechatronics*, 19(6), 1756-1763. doi:10.1109/tmech. 2014.2308312.

Kuçuk, S., Bingul, Z. (2004). The Inverse Kinematics Solutions of Industrial Robot Manipulators. *Proceedings of the IEEE International Conference on Mechatronics*. doi:10.1109/ ICMECH.2004.1364451.

Lee, C. S. G., Ziegler, M. (1984). Geometric Approach in Solving Inverse Kinematics of PUMA Robots. *IEEE Transactions on Aerospace and Electronic Systems*, AES-20(6), 695-706, doi:10.1109/taes.1984.310452.

Lin, M., Sun, L., Ding, Y. (2020). Construction of Robotic Virtual Laboratory System Based on Unity3D. *IOP Conference Series: Materials Science and Engineering*. doi:10.1088/1757-899X/768/7/072084.

Marzano, A., Friel, I., Erkoyuncu, J. A., Court, S. (2015). Design of a Virtual Reality Framework for Maintainability and Assemblability Test of Complex Systems. *Procedia CIRP*, 37, pp. 242-247. doi:10.1016/-j.procir.2015.08.067.

Mikolajczyk, T., Mikołajewska, E., Al-Shuka, H.F.N. (2022). Recent Advances in Bipedal Walking Robots: Review of Gait, Drive, Sensors and Control Systems. *Sensors*, 22(12). doi:10.3390/s22124440.

Mizuchi, Y., Inamura, T. (2017). Cloud-Based Multimodal Human-Robot Interaction Simulator Utilizing ROS and Unity Frameworks. *IEEE/SICE International Symposium on System Integration (SII)*. doi:10.1109/sii.2017.8279345.

Oke, G., Istefanopulos, Y. (2001). Gradient-Descent Based Trajectory Planning for Regulation of a Two-Link

Flexible Robotic Arm. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics. Proceedings*. doi:10.1109/aim.2001.936807.

Roldán, J. J., Peña-Tapia, E., Garzón-Ramos, D., de León, J., Garzón, M., del Cerro, J., Barrientos, A. (2018). Multi-Robot Systems, Virtual Reality and ROS: Developing a New Generation of Operator Interfaces. *Robot Operating System (ROS)*, pp. 29-64. doi:10.1007/978-3-319-91590-6_2.

Sharma, A. (2018). Guided Stochastic Gradient Descent Algorithm for inconsistent datasets. *Applied Soft Computing*. doi:10.1016/j.asoc.2018.09.038.

Sita, E., Horvath, C. M., Thomessen, T., Korondi, P., Pipe, A. G. (2017). ROS-Unity3D Based System for Monitoring of an industrial robotic process. *IEEE/SICE International Symposium on System Integration (SII)*. doi:10.1109/sii.2017.8279361.

Spong, M.W., Hutchinson, S., Vidyasagar M. Robot Modeling and Control", https://www.researchgate.net/profile/-Mohamed_Mourad_Lafifi/post/How_to_avoid_singular_conFigureurations/attachment/59d6361b79197b807799389a/AS%3A386996594855942%401469278586939/download/Spong+-+Robot+modeling+and+Control.pdf

Wang, H.P., Tian, Y., Vasseur, C. (2017). State Feedback Trajectory Tracking Control of Nonlinear Affine in Control System with Unknown Internal Dynamics and Disturbances. CEAI, Journal of Control Engineering and Applied Informatics, 19(3), 22-30.

Saleem, O., Abbas, F., Khan, M.U., Imtiaz, M.A., Khalid, S. (2018). Adaptive Collaborative Position Control of a Tendon-Driven Robotic Finger. CEAI, Journal of Control Engineering and Applied Informatics, 20(2), 87-99.