# Implementation of inverse kinematics algorithm for 6DoF robot arm in Unity

Jelena Vidaković, Andrija Dević, Nikola Živković
Robotics Department
Lola Institute
Belgrade, Serbia
jelena.vidakovic@li.rs

Vladimir Kvrgić
Robotics Department
Institute Mihajlo Pupin, University of Belgrade
vladimir.kvrgic@pupin.rs

Mihailo Lazarević
Department of Mechanics
Faculty of Mechanical Engineering, University of Belgrade
mlazarevic@mas.bg.ac.rs

*Abstract*— **In this paper, the implementation of the solution of the inverse kinematics problem for the 6DoF industrial robot arm in the Unity game engine is presented. Unity, one of the most popular game engines, is a very powerful tool and a leading platform for creating XR applications. Two different methods for the implementation of the solution of the inverse kinematics problem have been proposed: 1) Development and implementation of the inverse kinematics algorithm of a specific robot, and 2) Using inverse kinematics solvers by integration of Unity with dedicated robotics development frameworks. For verification of the proposed procedures, a serial robot with cylindrical joints RL15 is used.**

***Keywords-Virtual reality; augmented reality; robot arm; Unity; inverse kinematics;***

## I. Introduction

Integration of robotic solutions in manufacturing results in improved productivity and asset performance, reduced inefficiencies, and lower production and maintenance costs, while enhancing system agility and flexibility, adding value to the manufacturing process. In industrial settings, robotic arms with 6 degrees of freedom are common because they are often sufficient for many manufacturing tasks. Traditional approaches to programming robots, particularly in the manufacturing sector, are firmly established. Nevertheless, these methods tend to be time-intensive and frequently demand specialized expertise.

Extended reality (XR) is an umbrella term encompassing technologies of Augmented reality (AR), Virtual Reality (VR), and Mixed Reality (MR) have been increasingly used in human-machine communication, as the technological basis for the building of a new generation of Human-Machine Interfaces (HMI). XR is developing at a rapid pace and finding applications in various manufacturing aspects. Faster computers, sophisticated camera technologies, and innovative algorithms continue to inspire researchers to broaden the scope of XR applications. Recently, Extended reality (XR)-assisted solutions have been proposed in the robot programming domain [1].

With the increasing availability of game engines and their content, it is becoming more and more cost-effective for new simulators to use them as starting building blocks [2]. Unity, one of the most popular game engines, is a very powerful tool and leading platform for creating XR applications with high-quality 3D rendering which can be deployed across a variety of platforms and has the capability to integrate different XR SDKs [3].

To control the movements of the end-effector by a user in Unity in Cartesian space, a solution to the inverse kinematics problem has to be implemented in the robot's GameObject, a fundamental building block representing entities in the Unity scene. In this paper, the implementation of the solution of the inverse kinematics problem for the 6DoF industrial robot arm in the Unity game engine is presented. Two different methods for the implementation of inverse kinematics in Unity have been discussed. 6DoF industrial robot with cylindrical joints RL15 previously developed and installed at the Laboratory for Robotics and Machine Tools in Lola Institute [4-5] is used for demonstration of the proposed procedures.

## II. Implementation of Solution of Robot Inverse Kinematics Problem in Unity

A robot application program is a set of instructions that cause the robot system to move the robot's end effector in order to perform the desired robot task correctly. One of the essential ingredients of modern robot programming systems is the thorough usage of the frame concept. Robot links' poses and object locations as well as motions are expressed in accordance with human spatial intuition in terms of Cartesian coordinates [6]. The position of a rigid body is defined by the position vector of the origin of the frame rigidly attached to the body w.r.t. to the reference frame. For orientation, conventional robot programming systems commonly use an adopted form of Euler angles, a minimal representation of rigid body orientation, to define the orientation of the frame attached to the rigid body (end-effector) w.r.t. reference frame.

The solution of the inverse kinematics problem for 6DoF industrial robot problem integration in the robot's GameObject in Unity can be performed in two manners:

*1) Development and implementation of inverse kinematics algorithm of a specific robot;*

*2) Using inverse kinematics solvers by integration of Unity with dedicated robotics development frameworks.*

The first method is necessary within the design of standalone XR applications. The second method is related to integration of Unity as a simulator with a dedicated robotics development framework which enables advanced control, motion planning and programming features such as ROS [7], yielding the ROS-Unity robotics simulation suite. Other integrations are possible, such as with 3D simulation and offline/online programming environment RoboDK [8], which is presented here.

## A. *Robot RL15 description*

Robot RL15 is a 6DoF robot with cylindrical joints. In Fig.1, the actual robot installed at Lola Institute, and its model designed in SolidWorks is presented. Maximum rotational speeds for axes 1 to 6 are 3500, 2800, 3500, 3400, 3400, 3400 rev/min respectively.

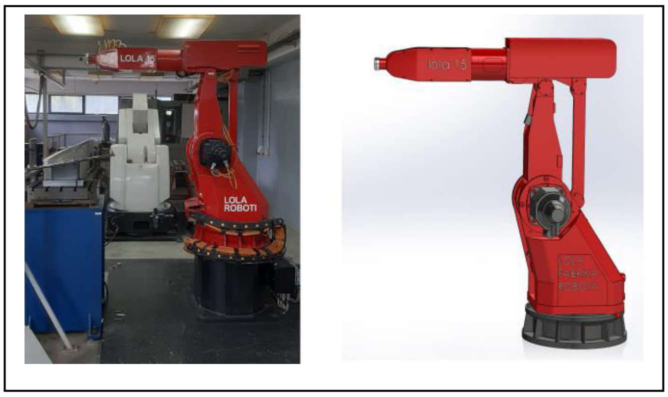

Figure 1. 6DoF industrial robot arm RL15

## B. *Integration of solution of inverse kinematics problem in Unity-RoboDK Suite*

RoboDK is a commercial cross-platform robot simulation and programming environment [8]. Integration of solution of inverse kinematics problem in Unity-RoboDK Suite is based on the designed CAD model which is exported into STEP (Standard for the Exchange of Product Data) file format. In order to import the robot model in Unity, the STEP file is converted into glTF file format [9], an open-source standard for the efficient transfer of 3D models and scenes from one application to another. Export of CAD robot model into glTF file format is performed using GLTFExporter add-on [10]. RoboDK's integrated inverse kinematics solver has been used for the created robot model [11]. It was necessary to write a script that allows interaction between Unity and RoboDK via RoboDK API. Code written in C# executes mapping of robot motion in RoboDK with robot motion in Unity in real-time. Given that RoboDK uses a right-handed orientation contrary to Unity, frame transformations between RoboDK and Unity have to be included, and they are presented below.

## C. *Derivation of solution for direct and inverse kinematics problem for Robot RL15*

The kinematic model of the robot RL15 is developed using Denavit–Hartenberg (D–H) convention, Fig.2. D–H parameters for RL15 are given in Table 1.

Figure 2. RL15: Frames and D-H parameters.

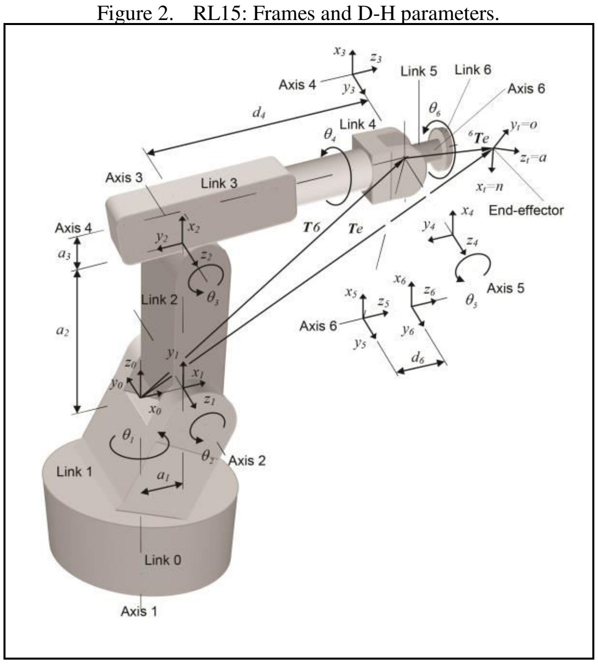

TABLE I. D–H PARAMETERS FOR THE RL15 LINKS

| Link | Variable [°] | *a* [mm] | *d* [mm] | α[°] |
|---|---|---|---|---|
| 1 | $q_1$ | 200 | 0 | 90 |
| 2 | $q_2$+90 | 600 | 0 | 0 |
| 3 | $q_3$ | 115 | 0 | 90 |
| 4 | $q_4$ | 0 | 825 | -90 |
| 5 | $q_5$ | 0 | 0 | 90 |
| 6 | $q_6$ | 0 | 0 | 0 |

The solution to the robot forward kinematics problem is used for motion definition, workspace definition, and the development of a solution to the inverse kinematics problem. The rotation matrix and position vector form 4×4 homogeneous transformation matrix (HMT) (1), widely used in the robotics community to define the poses of the frames attached to robot links.

$$ {}^{0}\mathbf{T}_1 = \left[\begin{array}{ccc|c} & {}^{0}\mathbf{R}_1 & & {}^{0}\mathbf{P}_1 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \tag{1} $$

In (1), HMT describing a pose of frame enumerated with 1 w.r.t. frame enumerated with 0 (e.g. BASE frame) is given. ${}^{0}\mathbf{R}_1$ is a rotational matrix, which can obtained using 12 different extrinsic and 12 different intrinsic Euler angle rotation sequences [12] for adopted frame orientation yielding the same numerical results w.r.t. adopted reference frame, while ${}^{0}\mathbf{P}_1$ is the position vector. From Fig. (2) and Table I, and by using formulation (1), HMTs describing a pose of frames $i$=1, 2,…,$n$ w.r.t. the BASE frame are given:

$$ {}^{0}\mathbf{T}_1=\begin{bmatrix} c_1 & 0 & s_1 & c_1a_1 \\ s_1 & 0 & -c_1 & s_1a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}^{0}\mathbf{T}_2=\begin{bmatrix} -c_1s_2 & -c_1c_2 & s_1 & c_1v_5 \\ -s_1s_2 & -s_1c_2 & -c_1 & s_1v_5 \\ c_2 & -s_2 & 0 & c_2a_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}, $$
$$ {}^{0}\mathbf{T}_3=\begin{bmatrix} -v_1 & s_1 & v_2 & X_6 \\ -v_3 & -c_1 & v_4 & Y_6 \\ c_{23} & 0 & s_{23} & Z_6 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}^{0}\mathbf{T}_4=\begin{bmatrix} v_{15} & -v_2 & v_{16} & X_6 \\ v_{17} & -v_4 & v_{18} & Y_6 \\ v_{19} & -s_{23} & v_{20} & Z_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2} $$
$$ {}^{0}\mathbf{T}_5=\begin{bmatrix} v_{21} & v_{16} & a_{x6} & X_6 \\ v_{23} & v_{18} & a_{y6} & Y_6 \\ v_{25} & v_{20} & a_{z6} & Z_6 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ {}^{0}\mathbf{T}_6=\begin{bmatrix} n_{x6} & o_{x6} & a_{x6} & X_6 \\ n_{y6} & o_{y6} & a_{y6} & Y_6 \\ n_{z6} & o_{z6} & a_{z6} & Z_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

where

$v_1=c_1s_{23},\ v_2=c_1c_{23},\ v_3=s_1s_{23},\ v_4=s_1c_{23},\ v_5=a_1-v_{13},v_{12}=c_{23}a_3+s_{23}d_4$
$v_{13}=s_2a_2,\ v_{14}=c_2a_2,\ v_{15}=-v_1c_4-s_1s_4,\ v_{16}=s_1c_4-v_1s_4,\ v_{17}=c_1s_4-v_3c_4,$
$v_{18}=-v_3s_4-c_1c_4,\ v_{19}=c_{23}c_4,v_{20}=c_{23}s_4,\ v_{21}=v_{15}c_5-v_2s_5,\ v_{23}=v_{17}c_5-v_4s_5,$
$v_{25}=v_{19}c_5-s_{23}s_5,\ v_{31}=d_4c_{23}-s_{23}a_3,\ n_{x6}=v_{21}c_6+v_{16}s_6,\ n_{y6}=v_{23}c_6+v_{18}s_6,$
$n_{z6}=v_{25}c_6+v_{20}s_6,\ o_{x6}=v_{16}c_6-v_{21}s_6,\ o_{y6}=v_{18}c_6-v_{23}s_6,\ o_{z6}=v_{20}c_6-v_{25}s_6$
$a_{x6}=v_{15}s_5+v_2c_5,\ a_{y6}=v_{17}s_5+v_4c_5,\ a_{z6}=v_{19}s_5+s_{23}c_5,\ X_6=c_1v_0,\ Y_6=s_1v_0,$
$Z_6=v_{12}+v_{14},\ v_0=v_5+v_{31}.$

and shorthand notations, $\sin(q_i) = s_i$, $\cos(q_i) = c_i$, $\sin(q_i + q_j) = s_{ij}$, and $\cos(q_i + q_j) = c_{ij}$ is used.

In ${}^{0}\mathbf{T}_6$ (2), $X_6$, $Y_6$, $Z_6$ represent coordinates of the frame attached to the sixth link (for simplicity, it is assumed that it coincides with end-effectors frame). Here, to define end-effector orientation w.r.t. BASE frame based on rotational matrix ${}^{0}\mathbf{R}_6$ in ${}^{0}\mathbf{T}_6$ (2), the Euler angles sequence ZYX (A, B, C) of intrinsic rotations to form rotation matrix in ${}^{0}\mathbf{T}_6$ (2) is used. This implies three successive rotations: rotation about axis $z$, for value $A$, followed by a rotation about axis $y$', of the newly rotated frame for value $B$, followed by a rotation about axis $x$'' of the newly rotated frame for value $C$, all positive in a counter-clockwise direction.

The inverse kinematics problem of a robot can be obtained as a numerical or analytical solution. An analytical solution can be obtained using either algebraic or geometric method. Here, an algebraic solution using manipulation of terms of the HTMs describing the relation between the successive links of the RL15 as well as inverses of the mentioned HTMs is used [5]. Manipulation of terms of (2), as well as HMTs ${}^{j}\mathbf{T}_i$ , ${}^{j}\mathbf{T}_i$ where ${}^{j}\mathbf{T}_i=({}^{j}\mathbf{T}_{j+1})^{-1}\times..\ {}^{0}\mathbf{T}_1^{-1}\times{}^{0}\mathbf{T}_6$, $i$=1..6, $j$=1..$i$, provides a solution for the inverse kinematics problem [5]. The algebraic method allows multiple solutions to the inverse kinematics problem. Here, the following is obtained:

$$ q_1=a\tan 2(Y_6,X_6), \tag{3} $$

$$ \begin{gathered} q_3=a\tan 2(pom,\sqrt{1-pom^2})-\beta, \\ pom=\frac{p^2+Z_6^2-l_1}{2a_2l},\ p=-s_2a_2+lc_{23\beta},\ \beta=\arctan(a_3/d_4), \end{gathered} \tag{4} $$

$$ \begin{gathered} q_2=a\tan 2(pZ_6+p_1\sqrt{p^2+Z_6^2-p_1^2},p_1^2-Z_6^2), \\ p_1=s_3a_3-c_3d_4, \end{gathered} \tag{5} $$

$$ \begin{gathered} q_4=a\tan 2(a'',a'), \\ a'=s_{23}a'''-c_{23}a_{z6},\ a''=s_1a_{x6}-c_1a_{y6},\ a'''=c_1a_{x6}+s_1a_{y6} \end{gathered} \tag{6} $$

$$ q_5=a\tan 2(s5,c5) \tag{7} $$

$$ \begin{gathered} q_6=a\tan 2((-n'n_{x6}-n''n_{y6}+n'''n_{z6}),(-n'o_{x6}-n''o_{y6}+n'''o_{z6})) \\ n'=-s_4c_1s_{23}-c_4,\ s_1n''=-s_4s_1s_{23}+c_4c_1,\ n'''=-s_4c_{23} \end{gathered} \tag{8} $$

The developed algorithm has to be implemented in Unity as C# scripts. Position and speed limits provided by actuators' manufacturers are included as well.

## *D. Frame transformation*

Using Unity editor as an integrated development environment for industrial robot arm 3D simulations carries certain challenges considering that, unlike specialized robotics development environments, Unity wasn't initially designed with the explicit purpose of accommodating robotics applications. Regarding one of the fundamental aspects such as frame orientation, Unity uses left-handed frame orientation. Unity Euler angles are $X$, rotation about the $x_U$ axis of the Unity frame, Fig. 3, $Y$, rotation about the $y_U$ axis of the Unity frame, Fig. 3, and $Z$, rotation about the $z_U$ axis of the Unity frame, Fig. 3. In practical terms, left-handed frame employed by Unity has to be translated to the right-handed frame used to define solutions of robot RL15 direct and inverse kinematics problems. Taking into account that conversion from the left to right-handed frames and vice versa is impossible by three successive rotations, here appropriate axes transformations have been carried out by visual axes matching, from Unity frame to inverse kinematics algorithm (IKa) frame used in ${}^{0}\mathbf{T}_6$ (2), and are graphically represented in Fig. 3.

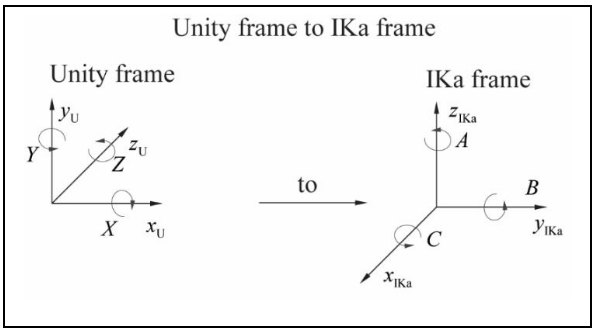

Figure 3. Unity GLOBAL frame and IKa BASE frame

From Fig. 3., it can be adopted that $A$=-$Y$; $B$=-$X$; $C$=$Z$ which yields:

$$\mathrm{Rot}(y_U, -Y)=\begin{bmatrix}\cos(-Y) & 0 & \sin(-Y)\\ 0 & 1 & 0\\ -\sin(-Y) & 0 & \cos(-Y)\end{bmatrix} \quad (9)$$

$$\mathrm{Rot}(x_U, -X)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos(-X) & -\sin(-X)\\ 0 & \sin(-X) & \cos(-X)\end{bmatrix} \quad (10)$$

$$\mathrm{Rot}(z_U, Z)=\begin{bmatrix}\cos(Z) & -\sin(Z) & 0\\ \sin(Z) & \cos(Z) & 0\\ 0 & 0 & 1\end{bmatrix} \quad (11)$$

By substituting (9-11) into the rotational sequence (12), numerical input ${}^{0}\mathbf{R}_6$ in ${}^{0}\mathbf{T}_6$ (2) is obtained.

$$^{\mathrm{IKaBASE}}\mathbf{R}_{\mathrm{EE}}=\mathrm{Rot}\left(z_{\mathrm{IKa}}, A\right)\mathrm{Rot}\left(y_{\mathrm{IKa}}, \mathrm{B}\right)\mathrm{Rot}\left(x_{\mathrm{IKa}}, C\right) \quad (12)$$

## III. VERIFICATION OF THE PROPOSED METHODS AND ALGORITHMS

In Fig.4., Unity-RoboDk Suite: programming and simulation of RL15 is presented.

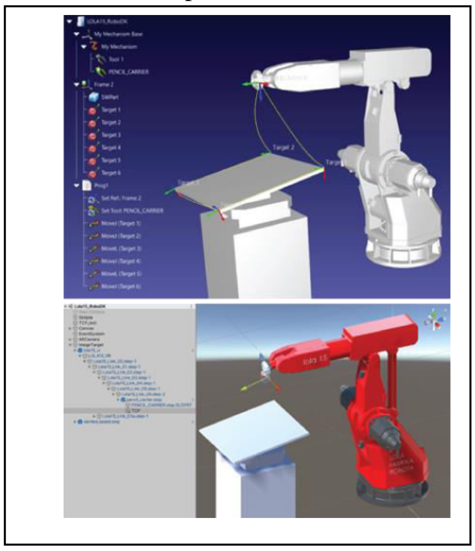

Figure 4. Unity-RoboDk Suite: programming and simulation of RL15 [11]

In Fig.5., the programming and simulation of RL15 in Unity editor based on the developed and implemented inverse kinematics algorithm presented herein is shown. In Fig.6, the developed AR application is given. The correctness of the pre-engineered algorithms was confirmed by simulations of programmed movements.

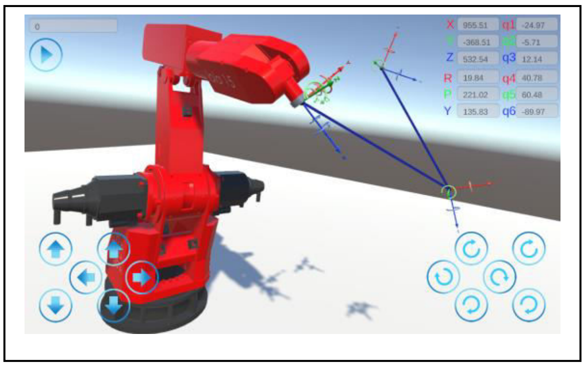

Figure 5. RL15 programming in Unity

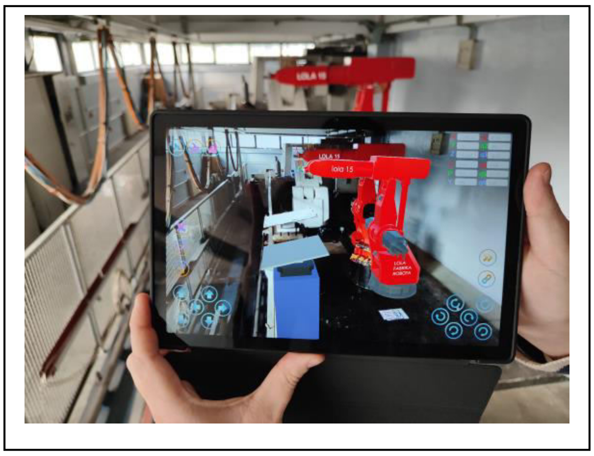

Figure 6. Developed AR Android app

## IV. CONCLUSON

In this paper, two distinct manners for implementation of solution of inverse kinematics problem for 6DoF industrial robot arm in Unity game engine are presented in order to create XR robot programming applications. The proposed procedures are verified using the serial robot with cylindrical joints RL15.

## ACKNOWLEDGMENT

This research has been supported by the research grants of the Serbian Ministry of Science, Technological Development and Innovations, grant No. 451-03-65/2024-03/200105 from 5.2.2024 and No. 451-03-66/2024-03/200066.

## REFERENCES

[1] S. K. Ong, A. W. W Yew, N. K Thanigaivel, A. Y .C. Nee, “Augmented Reality-Assisted Robot Programming System for Industrial Applications,” Robot. Comput. Integr. Manuf., vol. 61, pp. 101820, 2020.

[2] W. Garratt, S. Rithviik, F. Wang, “Achieving Interoperability Between Gaming Engines by Utilizing Open Simulation Standards,” In

Proceedings of the 2023 Simulation Innovation Workshop (SIW); Simulation Interoperability Standards Organization - SISO, 2023, pp. 446.

[3] Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine Available online: https://unity.com (accessed on 18 December 2023).

[4] V. Kvrgic, J. Vidakovic, "Efficient Method for Robot Forward Dynamics Computation," Mech. Mach. Theory., vol. 145, pp. 103680, 2020.

[5] V. Kvrgic, Development of intelligent system for control and programming of industrial robots, Doctoral dissertation, Faculty of Mechanical Engineering, University of Belgrade, Belgrade, 1998.

[6] F. M. Wahl, T. Ulrike, Robot programming-from simple moves to complex robot tasks, Institute for Robotics and Process Control, Technical University of Brawnschweig, 2002.

[7] ROS: Home Available online: https://www.ros.org/ (accessed on 11 January 2024).

[8] Simulator for Industrial Robots and Offline Programming - RoboDK Available online: https://robodk.com/ (accessed on 18 December 2023).

[9] glTF - Runtime 3D Asset Delivery Available online: https://www.khronos.org/gltf/ (accessed on 18 December 2023).

[10] GLTFExporter – Three.Js Docs Available online: https://threejs.org/docs/#examples/en/exporters/GLTFExporter (accessed on 18 December 2023).

[11] J. Vidaković, A. Dević, I. Lazarević, N. Živković, Design of Augmented Reality-Based Android App for Simulation and Programming of Industrial Robots.; New Trends in Engineering Research, Proceedings of the International Conference of Experimental and Numerical Investigations and New Technologies, CNNTech 2023, 2023.

[12] J. Diebel, "Representing attitude: Euler angles, unit quaternions, and rotation vectors," Matrix, vol. 58(15-16), pp. 1-35, 2006.