Development of Augmented Reality Application for Teaching Mechanism and Robotics in ROS

Efe Kurdoglu, Oguzhan Unlu, Fatih Cemal Can, Mertcan Kocak

Faculty of Engineering and Architecture, Mechatronics Engineering Department
Izmir Katip Celebi University
Havaalani Sosesi Cd. No:33/2, Izmir, Turkey

efekurdoglu@outlook.com, unlu.oguzhan.1998@gmail.com, fatihcemal.can@ikcu.edu.tr, mertcan.kocak@ikcu.edu.tr

ABSTRACT

In this research, we present an approach to development of augmented reality application for teaching mechanism and robotics in ROS (Robot Operating System). Our approach uses Unity (Vuforia), ROS and SolidWorks that allows a user to specify high-level requests to the 6 degree of freedom robot manipulator, to build, approve or modify the computed robot motions. The proposed approach exploits requiring low-level motion specifications provided by the user. Furthermore, the robot can be used in education purposes. In the end, we present a discussion of its applicability in collaborative environments.

Keywords

ROS (Robot Operating System), Augmented Reality, Unity, Vuforia

1. INTRODUCTION

*ROS is an open-source, meta-operating system and can be robotics middleware [8]. It provides almost all the requirements we would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management [1,9]. ROS is a well-contributed and one of the most used software in robot industry all around the world. ”Since its introduction, the Robot Operating System (ROS), has gained popularity among robotics researchers as an open-source framework for the development of robotics applications.” [14]. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behavior across a wide variety of robotic platforms [2].

*Available: http://wiki.ros.org/ROS/Introduction

Recently, with the development of ROS and robot manipulators, there has been an increasing interest in robotic systems that can share their workspaces and collaborate [9] with humans. New technologies for robot, ROS and human interaction have begun to be developed. Augmented reality is one of the most popular technologies. Moreover, the Industry 4.0 paradigm triggered the use of AR in networks of connected physical systems and human-machine communication [3]. The most common definition of AR by Azuma states that in AR “3D virtual objects are integrated into a 3D real environment in real-time”.

So, how can we integrate robots into an AR system? Unity3d supports integration between robot-AR. “The exchange of data packets between the Unity server and the ROS client was achieved by TCP/IP protocol [8]. On the client side, a ROS node was developed *unitysocket,* which communicates to the present IP address of the Unity server.” [9]. Unity is a game engine that is used for many popular 2D, 3D, and Virtual/Augmented/Mixed Reality applications. It has a built-in physics engine that can handle contact dynamics, as well as material simulation (such as water, sand, or cloth). It supports integration with most common VR (and AR) hardware, and provides a shader language for writing custom GPU shaders [4].

We designed a 6 Dof serial manipulator in Solidworks. The manipulator was simulated in ROS Gazebo with the help of rviz and moveit configuration. Unity was used to transform our design into augmented reality. This paper is arranged as following sections. In section 2, design and kinematic analysis are presented by using Denavit-Hartenberg parameters and transformation matrices. ROS basics including topics, services, and actions are presented in section 3. Then, ROS visualization tools such as Rviz, Gazebo, and Moveit are illustrated using simulation figures of our robot in section 4. In section 5, we present process of transforming our serial manipulator from ROS to Unity. Finally, we draw a conclusion for this research in section 6.

2. DESIGN AND ANALYSIS

2.1 Denavit-Hartenberg Homogenous Transformation Matrices

The transformations between each two successive joints can be written by simply substituting the parameters from the parameters table into the transformation matrix as follows,

$$T_i^{i-1} = T(d_i)T(\theta_i)T(a_i)T(\propto_i)$$

$$\boldsymbol{T_i^{i-1}} = \begin{vmatrix} c\theta_i & -\boldsymbol{c} \propto_i s\theta_i & \boldsymbol{s} \propto_i s\theta_i & \boldsymbol{a_i} c\theta_i \\ s\theta_i & \boldsymbol{c} \propto_i c\theta_i & -\boldsymbol{s} \propto_i c\theta_i & \boldsymbol{a_i} s\theta_i \\ \boldsymbol{0} & \boldsymbol{s} \propto_i & \boldsymbol{c} \propto_i & \boldsymbol{d_i} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} \end{vmatrix}$$

D-H steps were used to draw kinematic structure of the manipulator. Kinematic structure of manipulator is shown in Figure 1.

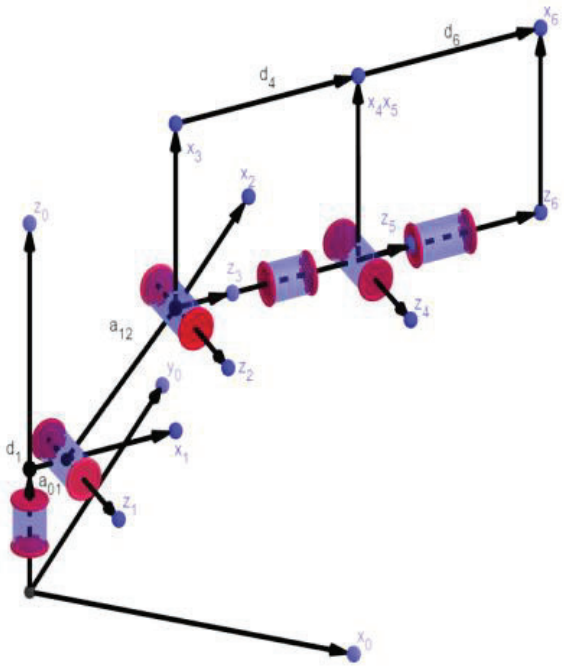

Figure 1. Simple Kinematics Structure

According to kinematics structure of the manipulator, Denavit-Hartenberg table is defined in Table 1. In Table 1, $\propto_{i-1,i}$ indicates twist angles between link i-1 and i, $a_{i-1,i}$ is defined as link length, d_i is offset of joint i, $\theta_{i\,(v)}$ is joint angle of joint i. Integer numbers from 1 to 6 describes link number of the manipulator. All joint angles are variable.

Table 1. Denavit-Hartenberg

	$\propto_{i-1,i}$	$a_{i-1,i}$	d_i	θ_i
1	90	$a_{0,1}$	d_1	$\theta_{1(v)}$
2	0	$a_{1,2}$	0	$\theta_{2(v)}$
3	90	0	0	$\theta_{3(v)}$
4	-90	0	d_4	$\theta_{4(v)}$
5	90	0	0	$\theta_{5(v)}$
6	0	0	d_6	$\theta_{6(v)}$

2.2 Forward Kinematics Analysis

Kinematic analysis mainly separates into two parts which are the forward kinematic analysis and the inverse kinematic analysis. The forward kinematic analysis means that the joint space is known to find any location and pose of the end of robot [5]. T_6^0 represents a vector that begins at base joint to end effector.

$$T_6^0 = \begin{vmatrix} n_X & o_X & a_X & p_X \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

where;

$n_X = s\theta_6 * [c\theta_4 * s\theta_1 - s\theta_4 * c\theta_1 * c(\theta_2 + \theta_3)] + c\theta_6 * [c\theta_5 * (s\theta_1 * s\theta_4 + c\theta_4 * c\theta_1 * c(\theta_2 + \theta_3)) - s\theta_5 * c\theta_1 * s(\theta_2 + \theta_3)]$

$n_y = -s\theta_6 * [c\theta_1 * c\theta_4 + s\theta_4 * s\theta_1 * c(\theta_2 + \theta_3)] - c\theta_6 * [c\theta_5 * (c\theta_1 * s\theta_4 - c\theta_4 * s\theta_1 * c(\theta_2 + \theta_3)) + s\theta_5 * s\theta_1 * s(\theta_2 + \theta_3)]$

$n_Z = c\theta_6 * [s\theta_5 * c(\theta_2 + \theta_3) + c\theta_4 * c\theta_5 * s(\theta_2 + \theta_3)] - s\theta_4 * s\theta_6 * s(\theta_2 + \theta_3)$

$o_X = c\theta_6 * [c\theta_4 * s\theta_1 - s\theta_4 * c\theta_1 * c(\theta_2 + \theta_3)] - s\theta_6 * [c\theta_5 * (s\theta_1 * s\theta_4 + c\theta_4 * c\theta_1 * c(\theta_2 + \theta_3)) - s\theta_5 * c\theta_1 * s(\theta_2 + \theta_3)]$

$o_y = s\theta_6 * [c\theta_5 * (c\theta_1 * s\theta_4 + c\theta_4 * s\theta_1 * c(\theta_2 + \theta_3)] + s\theta_5 * s\theta_1 * s(\theta_2 + \theta_3)] - c\theta_6 * [c\theta_1 * c\theta_4 + s\theta_4 * s\theta_1 * c(\theta_2 + \theta_3)]$

$o_Z = -s\theta_6 * [s\theta_5 * c(\theta_2 + \theta_3) + c\theta_4 * c\theta_5 * s(\theta_2 + \theta_3)] - c\theta_6 * s\theta_4 * s(\theta_2 + \theta_3)$

$a_X = s\theta_5 * [s\theta_1 * s\theta_4 + c\theta_4 * c\theta_1 * c(\theta_2 + \theta_3)] + c\theta_5 * c\theta_1 * s(\theta_2 + \theta_3)$

$a_y = c\theta_5 * [s\theta_1 * s(\theta_2 + \theta_3)] - s\theta_5 * [c\theta_1 * s\theta_4 - c\theta_4 * s\theta_1 * c(\theta_2 + \theta_3)]$

$$a_z = c\theta_4 * s\theta_5 * s(\theta_2 + \theta_3) - c\theta_5 * c(\theta_2 + \theta_3)$$

$$p_X = d_6 * \left[s\theta_5 * \left(s\theta_1 * s\theta_4 + c\theta_4 * c\theta_1 * c(\theta_2 + \theta_3)\right) + c\theta_5 * c\theta_1 * s(\theta_2 + \theta_3)\right] + d_4 * c\theta_1 * s(\theta_2 + \theta_3) + a_1 * c\theta_1 + a_2 * c\theta_1 * c\theta_2$$

$$p_y = d_4 * s\theta_1 * c(\theta_2 + \theta_3) - d_6 * \left[s\theta_5 * \left(c\theta_1 * s\theta_4 - c\theta_4 * s\theta_1 * c(\theta_2 + \theta_3)\right) - c\theta_5 * s\theta_1 * s(\theta_2 + \theta_3)\right] + a_1 * s\theta_1 + a_2 * c\theta_2 * s\theta_1$$

$$p_z = d_1 - d_4 * c(\theta_2 + \theta_3) + a_2 * s\theta_2 - d_6 * [c\theta_5 * c(\theta_2 + \theta_3) - c\theta_4 * s\theta_5 * s(\theta_2 + \theta_3)]$$

c: Cosine function s: Sine function

The transformation matrix consists of rotation and position of end effector of the manipulator. Here, n_X, n_y, n_z, …. are related to orientation of the end effector whereas p_X, p_y, p_z are position components of the end effector.

2.3 Inverse Kinematics Analysis

Inverse kinematic analysis is the opposite of the forward kinematic analysis. “Inverse kinematics of the manipulator must be solved to control motion of the end effector.” [12,5]. The inverse kinematic analysis means that the pose of the end effector is known to find the joint space. Every value of θ_n represents the degree of joints that is illustrated in Figure 1 and Table1.

2.4 Dynamics Analysis

In this section, dynamic problems are solved for serial manipulator using Newton-Euler formulation. Newton-Euler consists of forward and backward computations. Forward computation is used to compute angular velocity, angular acceleration, linear velocity, and linear acceleration of each link. Backward computation is utilized to compute joint torques and forces [5].

Torques are calculated for each joint using Newton-Euler formulation. And then, *Roboanalyzer program [16] is used to verify our results. We used cosine trajectory for joint angles of the manipulator without external load. We applied these joint inputs from 0 to 2 second. Each joint torques are shown in Figure 2.

*Available: http://www.roboanalyzer.com/

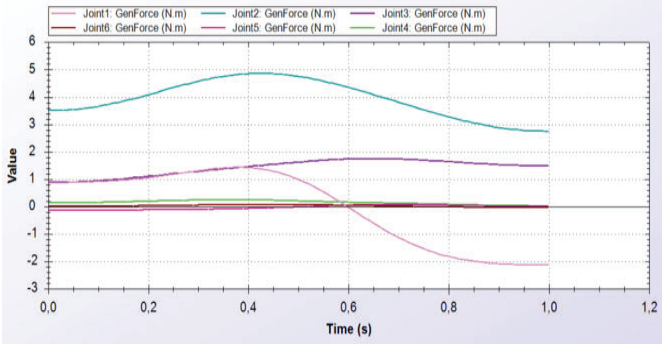

Figure 2. Torque of Joints (N.m)

2.5 Mechanical Design

In this section, 6 Degree of Freedom serial robot is created by using SolidWorks. Serial manipulator consists of 6 revolute joints and a gripper as shown in Figure 3. Computer Aided Design (CAD) and Engineering skills allow us to design any complex systems and at any scale [7].

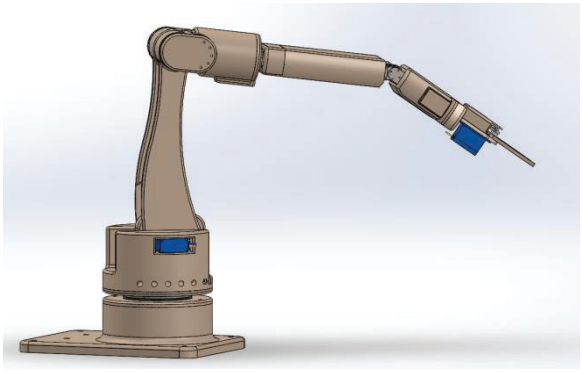

Figure 3. Collapsed Side View in SW

To control each joints of serial manipulators, servo motors were used. Serial manipulator has two type servo motors as dual shaft servo motor and digital servo motor. There are six digital servo motors and two dual shaft servo motors. Also, bearings were used to decrease applied load on servo motors. In this way, servo motor will not draw so much current and will have a long life. Exploded back and side views are shown in Figure 4 and 5, respectively.

Table 2. List of Assembly Components

Components	Technical Properties	Items
High Torque Robot Servo (Dual Shaft Servo) (7.4V)	35kg.cm 0.11sec/60 degree 180º	1
High Torque Robot Servo (Dual Shaft Servo) (8.4V)	70kg.cm 0.13sec/60 degree 180º	1
High torque Robot servo (8.4V)	60kg.cm 0.13sec/60 degree 180º, 270º	5
Trust Ball Bearing (51115)	75x100x19 (mm) C_0: 134 kN C: 44.2 kN	1
Deep Groove Bearings (6004)	20x42x12 C_0: 5 kN C: 9.95 kN	2
Deep Groove Bearings (6005)	25x47x12 C_0: 6.55 kN C: 11.9 kN	1
Deep Groove Bearings (6007)	35x62x14 C_0: 10.2 kN C: 16.8 kN	1

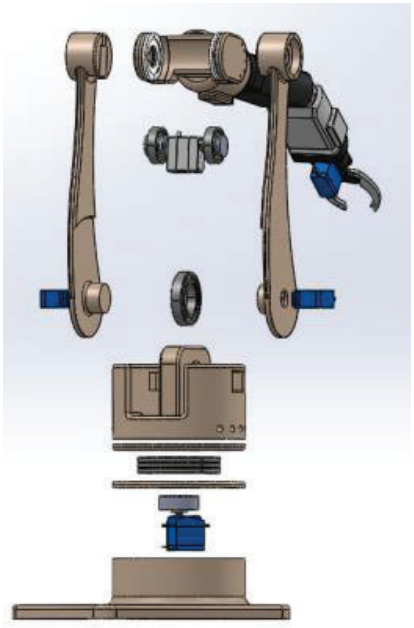

Figure 4. Exploded Back View in SW

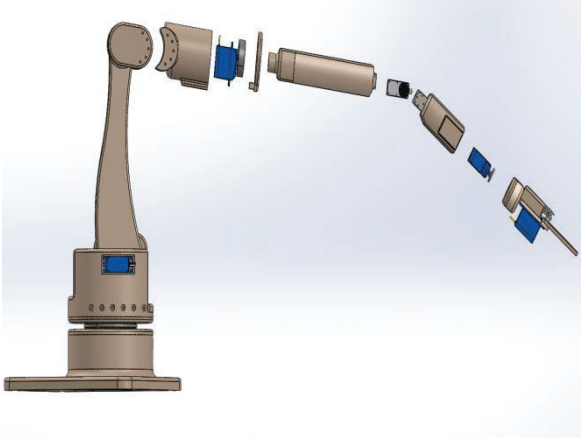

Figure 5. Exploded Side View in SW

2.6 Export to URDF

In order to implement the model into ROS, we should get the URDF file which include properties of the robot. To do that, a *URDF exporter must be installed in SolidWorks. And then, all the links, joints should be configured and organized. Reference axes, coordinate systems and joint types are detected automatically. After previewing the URDF package, you can arrange limits of joints, dynamical parameters such as friction and damping, safety controller, etc.

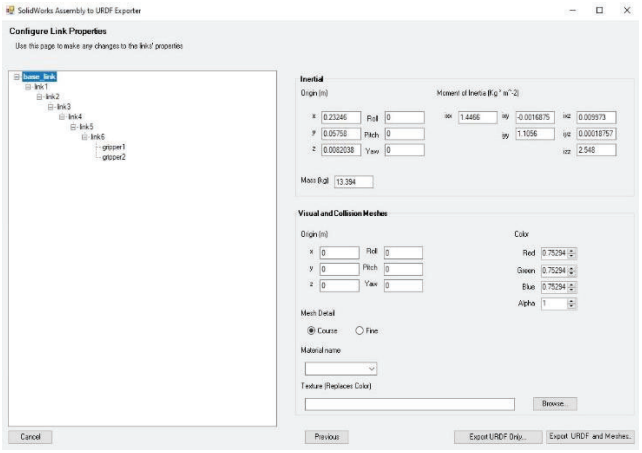

Figure 6. Configuration of Parameters in URDF Exporter

*http://wiki.ros.org/sw_urdf_exporter

3. ROS BASICS

ROS is an open source that provides to control robots via computer and used as a robotics middleware. "Running sets of ROS-based processes are represented in a graph architecture where processing takes place in nodes that may receive, post and multiplex sensor, control, state, planning, actuator, and other messages. Despite the importance of reactivity and low latency

in robot control, ROS, itself, is not a real-time OS (RTOS), though it is possible to integrate ROS with real-time code. Community addressed the lack of support for real-time systems in ROS 1.0, and they are improving it in the creation of ROS 2.0" [6]

3.1 Nodes

A node is a smallest processor unit that provides ROS to communicate with other nodes. There are quite a few ROS functionalities which could be included within a single directory named a package, and each capability within each package can be taken into particular scripts called as nodes [15,1]. In this regard, there are several nodes with different name and examples of nodes are shown in Figure 7.

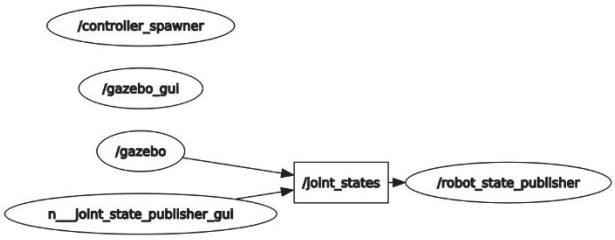

Figure 7. RQT plot for a node

3.2 Roscore

To connect all the nodes as a chain, a service is needed to be provided, whose name is roscore. All the nodes which are created needed ROS packages to form a direct peer-to-peer connection with the other nodes. So, it helps all the nodes find each other and use the necessary requirements by ROS for the projects [2, 15].

3.3 Topics

A topic is a name for a stream of messages with a defined type. Topics implement a publish/subscribe communication mechanism. Publish is the transmission of messages corresponding to the topic. Subscribe is the receiving of messages corresponding to the topic [1].

3.4 Services

Services are just synchronous remote procedures calls; they allow one node to call a function that executes in another node. The server (which provides the service) specifies a callback to deal with the service request and advertises the service. The client (which calls the service) then accesses this service through a local proxy. "The ROS action server on the Real-Time Server shares with the 1kHz real-time control loop a common memory buffer. This allows us to efficiently pass data, like controller parameters, between these two processes without costly serialization and deserialization" [13]

3.5 Action

Action is another type of message communication management used for bidirectional communication. Action is used where it takes longer to respond after a messaging request is received and communication must continue until a result is received. Action consists of goal, feedback, and result [1].

3.6 Single Board Computer

Microcontrollers have been used to take control of robots for many years. However, ROS and some other application need even much more power according to past. Currently, robot manipulator model provides an *NVIDIA Jetson Nano 4GB Developer Kit running on Ubuntu 18.04 LTS ROS Melodic. Furthermore, this single-board computer will run all the software essentials as a brain of the manipulator.

*Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

4. ROS VISUALIZATION TOOLS

4.1 Gazebo

Gazebo is a multirobot simulator for complex indoor and outdoor robotic simulation. Complex robots, robot sensors, and a variety of 3D objects can be simulated by creating real environment. Gazebo provides us numerously high-performance physical engines [11, 8]. It already has many libraries such as Kdl, ODE, Bullet, Simbody, DART and so on. Furthermore, they are all open source, can be used and contributed. So that, we can create a real physical environment as expected. Thus, the possible movements in real world can be predicted in this environment. To launch our manipulator in Gazebo (see Figure 8), terminal command is used as follows,

$roslaunch ikc_robot ikc_robot.launch

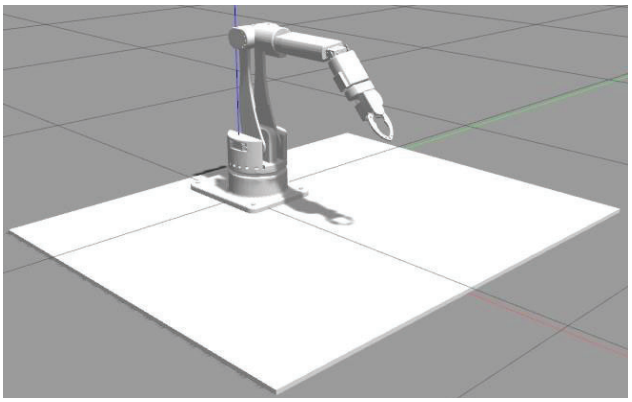

Figure 8. Gazebo simulation environment

4.2 Rviz

Rviz is a 3D visualization tool for ROS applications. It provides a view of your robot model, capture sensor information from robot sensors, and replay captured data. It can display data from camera, lasers, from 3D and 2D devices including pictures and point clouds [1]. In order to visualize the robot, there should be 'base_link' as a Fixed Frame. Furthermore, RobotModel and MotionPlanning should be added into Rviz environment to launch the robot in Rviz (see Figure 9). To launch our manipulator in Rviz, terminal command is used as follows,

$roslaunch ikc_robot display.launch

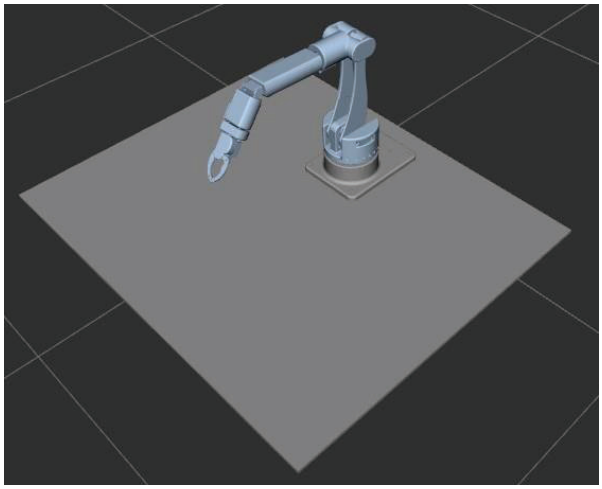

Figure 9. Rviz simulation environment

4.3 MoveIt

To simulate the robot, we need to create another package which is provided by *MoveIt. It is one of the most developed and used tools which can make motion planning [14], operation, 3B detection, inverse kinematics (in our case, we used KDLKinematicsPlugin) and integration of control and navigation algorithms [1]. "MoveIt is a ROS-based motion planning framework that provides an inverse kinematics solver using inverse Jacobian methods [14]. It allows the user to plan a collision-free path of the arm from point A to point B." [10]

*Available:docs.ros.org/en/melodic/api/moveit_tutorials/html/doc/getting_started/getting_started.html

$ roslaunch ikc_robot ikc_robot.launch

$ roslaunch ikc_robot_moveit_config move_group.launch

$ roslaunch ikc_robot_moveit_config moveit_rviz.launch

When all three commands are launched in the terminal, Motion Planning modules will be appeared as shown in Figure 10.

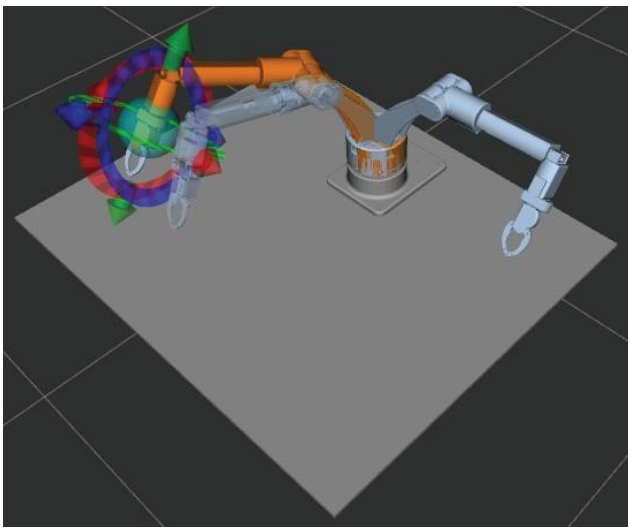

Figure 10. MoveIt simulation environment

5. ROS TO UNITY

URDF model of our robot manipulator is imported from ROS to Unity through ROSBridge Client in the course of converting the robot's Universal Robotic Description Format (URDF) to a Unity object by UrdfImporter. While ROS publishes some messages, Unity subscribes these messages [10].

5.1 RosBridge

Integration of ROS and Unity needs to be done for augmented reality application. So, ROS bridge libraries are necessary in order to connect. It is a Python library, for requesting JSON [8] strings and converting into ROS messages, and vice versa. It helps us to transport layer packages so that a WebSocket connection is created by the Rosbridge server package using the Python libraries for the conversion JSON to ROS [9,15].

5.2 Connecting to Unity

First, we need to create a server package. Two main files must be formed for getting files in binary and to save them in created folder. Then, we should check the IP address on computer. In Unity part, we need a ROS bridge client so that can be installed from Asset Store. Then, Robot Description is read and imported all the robot files which are simulated. Our manipulator was successfully imported into ROS as shown in Figure 11.

Available: https://github.com/siemens/ros-sharp/wiki/User_App_ROS_TransferURDFFromROS

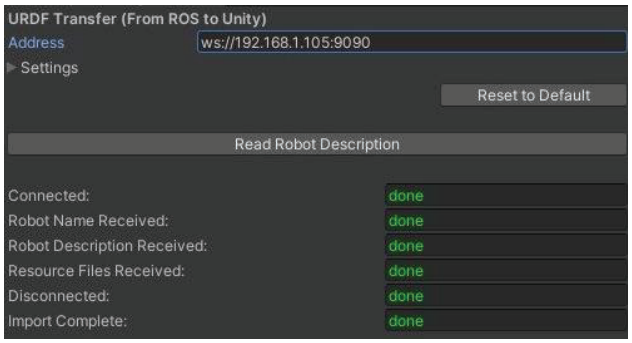

Figure 11. ROS Bridge Client

5.3 Augmented Reality (AR)

"A novel approach of using Augmented Reality (AR) creates a new design space and opportunity to facilitate more naturalistic teleoperation. On account of the ability of AR to augment the operator view and visualize the corresponding robotic task space, the operator can retrieve the robot's sensor state information (temperature level, position, joint angle, etc.) via graphic overlays" [10]. For this case, Vuforia Engine AR should be installed from Package Manager. And then, add an image for a target and ARCamera to capture the target into Hierarchy. In the end, when you capture the target, it will be visualized instantly (see Figure 12).

Figure 12. AR with Vuforia Engine

6. CONCLUSION

In this paper, we illustrate an innovative approach to use ROS with the Vuforia module in Unity3D engine. In education system, the model that is united ROS with the Unity3D engine gives a new perspective to produce advanced computer aided learning platform. Nowadays, AR projects, which present a new era, are employed so that it gets supported in ROS. With utilizing our own robot, we illustrate our novel interaction ROS and the Unity3D engine, design of 6 degree of freedom manipulator and kinematic & dynamic analyses of it.

This study is a base platform for teaching robots and mechanisms. Learning stages are going to be prepared based on this base platform. So that, students who have not a real robot can program and test robots and mechanisms in our platform. Finally, this project is still in the development process.

7. REFERENCES

[1] http://wiki.ros.org/ROS/Introduction

[2] Morgan Quigley (Author), Brian Gerkey (Author), William D. Smart (Author), 2015, Programming Robots with ROS: A Practical Introduction to the Robot Operating System,1st Edition, p.3

[3] Zhanat Makhataeva and Huseyin Atakan Varol, 17 February 2020, Augmented Reality for Robotics p. 1 of 28

[4] David Whitney, Eric Rosen, Daniel Ullman, Elizabeth Phillips, Stefanie Tellex, ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots, p.5 of 10, 2018

[5] Lung-Wen Tsai. February 1999, Robot Analysis and Design: The Mechanics of Serial and Parallel Manipulators.54-85

[6] R. Codd-Downey, P. Mojiri Forooshani, A. Speers, H. Wang and M. Jenkin, "From ROS to Unity: leveraging robot and virtual environment middleware for immersive teleoperation," in 2014 IEEE International Conference on Information and Automation, ICIA 2014, 2014.

[7] Florin GIRBACIA, Silviu BUTNARIU, April 26-27, 2012, An innovative approach to teaching mechanism using augmented reality technologies, The 8th International Scientific Conference eLearning and software for Education Bucharest, 140-143

[8] Edwin Babaians, Mohsen Tamiz, Yaser Sarfi, Amir Mogoei and Esmaeil Mehrabi Arsamrobotics Co. LTD. (2018), ROS2Unity3D; High-Performance Plugin to Interface ROS with Unity3d engine, 9th Conference on Artificial Intelligence and Robotics and 2nd Asia-Pacific International Symposium, 59-64

[9] Ahmed Hussein1, Fernando Garcia1 and Cristina Olaverri-Monreal2 Senior Member IEEE, 2018, ROS and Unity Based Framework for Intelligent Vehicles Control and Simulation p.3 of 6

[10] Chung Xue Er, Yuansong Qiao, niall Murray, "Enabling Human-Robot -Interaction for remote robotic operation via Augmented Reality", IEEE 21st International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), 2020

[11] Qian Zhang, Yongjia Zhao, "Implementation and Verification of a Virtual Testing System Based on ROS and Unity for Computer Vision Algorithms", 12th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), 2019

[12] Fatih Cemal Can, Önder Lapçin, Burak Ayan, Mehmet Çevik, 2018, Human Machine Interface Design for a 3 DoF Robot Manipulator, Universal Journal of Mechanical Engineering,

6(3): 47-53
[13] Kevin Zhang, Mohit Sharma, Jacky Liang, Oliver Kroemer, November 2020, A Modular Robotic Arm Control Stack for Research: Franka-Interface and FrankaPy. Available: https://arxiv.org/abs/2011.02398
[14] Andrea Testa, Andrea Camisa, Giuseppe Notarstefano, (Oct 2020), "ChoiRbot: Aros 2 Toolbox for Cooperative Robotics", IEEE Robotics and Automation Letters, Volume: 6, Issue: 2, April 2021. Available: https://arxiv.org/abs/2010.13431
[15] Corey Williams, Adam Schroeder, October 2020, "Utilizing ROS 1 and the Turtlebot3 in a Multi-Robot System", Technical Report of the Robotics and Automation Design Lab. Available: https://arxiv.org/abs/2011.10488
[16] Ratan Sadanand, Ravi Prakash Joshi, Rajeevlochana G. Chittawadigi, Subir Kumar Saha, CAD/CAM, Robotics and Factories of the Future pp 59-68, 2016