Digital Twin – Lab 3

Devika Shaj Kumar Nair[1], Darsh Patel[2], Aman Milind Dhale[3]
[1]Ira Fulton School of Engineering, Arizona State University, Arizona, United States of America
[2]Ira Fulton School of Engineering, Arizona State University, Arizona, United States of America
[3]Ira Fulton School of Engineering, Arizona State University, Arizona, United States of America

Article Info

Article history:

Received October 13, 2024

Keywords:

MyCobot 600 Pro
ROS
Homogeneous Transformation
Digital Twin

ABSTRACT

This report aims to conduct a forward kinematic analysis of the MyCobot Pro 600 robot. The goal is to develop a kinematic model, derive the necessary homogeneous transformation matrices, and create a digital twin of the robot in MATLAB using Simscape/robotics Toolbox and perform forward kinematics. Additionally, the MyCobot Pro 600's model will be imported into ROS and Rviz for visualization and simulation purposes. The final output involves performing a comparison between the actual robot's end-effector (e.e.) position and the corresponding Rviz simulation.

Corresponding Author:

Devika Shaj Kumar Nair
Ira Fulton School of Engineering, Arizona State University
Tempe, Arizona, United States of America
Email: dshajkum@asu.edu

1. INTRODUCTION

Mycobot 600 Pro is a 6 axis collaborative robot that was designed for both commercial and educational purposes. It has an operating radius of 600mm, with an effective load capability of 2kg. The robot weighs around 8.8kg and it utilises a raspberry pi microprocessor embedded with roboFlow visual programming software. It has 6 joints with the following joint angle limits[1].

Table 1: Joint angle range for Mycobot 600 Pro

Joint	Angle Range
J1	+/- 180°
J2	-270~90°
J3	+/- 150°
J4	-260~80°
J5	+/- 168°
J6	+/- 174°

The main aim of this lab report is to walkthrough the process followed for conducting the forward kinematic analysis of Mycobot 600 pro and generate a digital twin of the same in MATLAB and ROS. We also compare the results with the output we got from the physical model of the robot.

2. KINEMATIC MODEL AND HOMOGENEOUS TRANSFORMATIONS

Homogeneous transformation matrices are derived from the configuration diagram of the robot model. Here, the homogeneous transformation matrices of the MyCobot 600 pro robot is generated manually from the configuration diagram shown in figure 1: and the corresponding transformation matrices obtained are as follows:

```
H01 = [cos(t1)      0    sin(t1)    0;
       sin(t1)      0   -cos(t1)    0;
       0            1      0       210;
       0            0      0       1];

H12 = [-cos(t2)  -sin(t2)    0      -250*cos(t2);
       -sin(t2)   cos(t2)    0      -250*sin(t2);
        0          0        -1      76.2;
        0          0         0      1];

H23 = [-cos(t3) -sin(t3) 0       250*cos(t3);
       -sin(t3)  cos(t3) 0       250*sin(t3);
       0         0       -1      76.2;
       0         0        0      1];
```

```
H34 = [-cos(t4)   0  sin(t4)    0;
       -sin(t4)   0  -cos(t4)   0;
        0        -1      0     76.2;
        0         0      0     1];

H45 = [cos(t5)   0   sin(t5)    0;
       sin(t5)   0  -cos(t5)    0;
       0         1      0       107;
       0         0      0       1];

H56 = [-cos(t6)   sin(t6) 0     0;
       -sin(t6)  -cos(t6) 0     0;
       0         0        1     109.5;
       0         0        0     1];
```

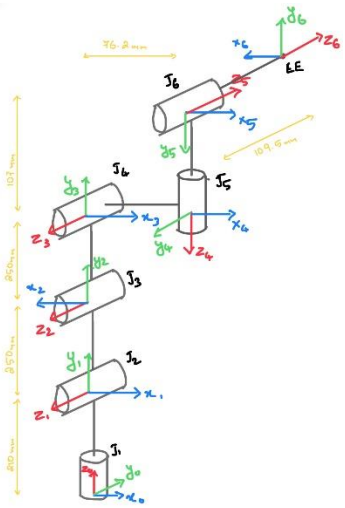

Figure 1: Configuration Diagram for Mycobot 600 Pro

The above homogeneous transformations yielded the following coordinates in matlab.

```
>> Homogeneous_Transformation
H06 =
   -1.0000         0         0         0
         0         0   -1.0000 -185.7000
         0   -1.0000         0  817.0000
         0         0         0    1.0000

X =
     0

Y =
  185.7000

Z =
   817
```

```
>> Homogeneous_Transformation
H06 =
         0         0    1.0000  185.7000
   -0.2588    0.9659         0 -521.6122
   -0.9659   -0.2588         0  479.1751
         0         0         0    1.0000

X =
 -185.7000

Y =
  521.6122

Z =
  479.1751
```

```
>> Homogeneous_Transformation
H06 =
    1.0000         0         0 -500.0000
         0         0   -1.0000 -185.7000
         0    1.0000         0  103.0000
         0         0         0    1.0000

X =
   500

Y =
  185.7000

Z =
   103
```

Figure 2: Homogeneous transformation results for Joint angles [0, -90, 0, -90, 0, 0], [90, -45, 30, -90, 0, 0] and [0,0,0,0,0,0] respectively

3. RESULTS FROM PHYSICAL MODEL

The Mycobot 600 pro model was actuated using the roboFlow visual programming software in the lab and the end effector coordinates corresponding to 3 sets of joint angles were captured as tabulated in Table:2. The figure 3 – 5 shows the configurations of the robot.

Table 2: End Effector positions from physical model

Joint Angle Set (deg)	X coordinate (mm)	Y coordinate (mm)	Z coordinate (mm)
[0, -90, 0, -90, 0, 0]	-11	185.64	816.90
[90, -45, 30, -90, 0, 0]	-190.80	515.72	485.72
[0, 0, 0, 0 ,0, 0]	499.448	187.211	99.948

Figure 3: Physical Bot configuration for joint angles [0, -90, 0, -90, 0, 0]

Figure 4:Physical Bot Configuration for joint angles [90, -45, 30, -90, 0, 0]

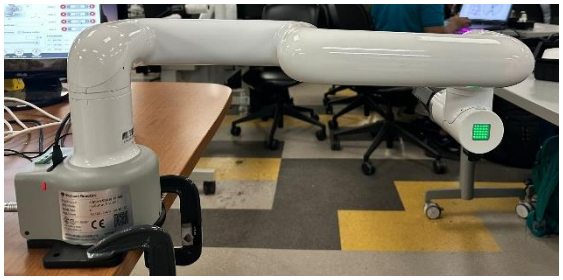

Figure 5: Physical Bot Configuration for joint angles [0, 0, 0, 0 ,0, 0]

4. DIGITAL TWIN USING ROS

Inorder to create the digital twin using ROS, it was essential to build a basic development environment in Ubuntu 20.4, which included ROS, MoveIt and Git version manager. Since there is no ROS software source in Ubuntu software list, we configured the ROS software source into the software repository and configured a public key so that the files download is facilitated smoothly before ROS installation. Later, the following command was used to install ROS noetic (the version corresponding to Ubuntu 20.4).

```
sudo apt install ros-noetic-desktop-full
```

As the second step, MoveIt and Git were also installed using the following commands.

```
sudo apt-get install git
sudo apt-get install ros-noetic-moveit
```

Post the basic setup of ROS, moveIT and Git were completed, catkin workspace was created and the necessary packages and URDF files for the mycobot 600 pro were downloaded from the github repository using the following commands [2].

```
cd ~/catkin_ws/src
git clone https://github.com/elephantrobotics/mycobot_ros.git
cd
catkin_make
source devel/setup.bash
```

Post this, the model was imported into Rviz and the joint angles were set and the following table 3 tabulates the results received in ROS.

Table 3: End effector coordinates from ROS

Joint Angle Set (deg)	X coordinate (mm)	Y coordinate (mm)	Z coordinate (mm)
[0, -90, 0, -90, 0, 0]	-0.003	189.4	807.32
[90, -45, 30, -90, 0, 0]	-187.51	521.34	469.36
[0, 0, 0, 0 ,0, 0]	500.16	181.01	88.82

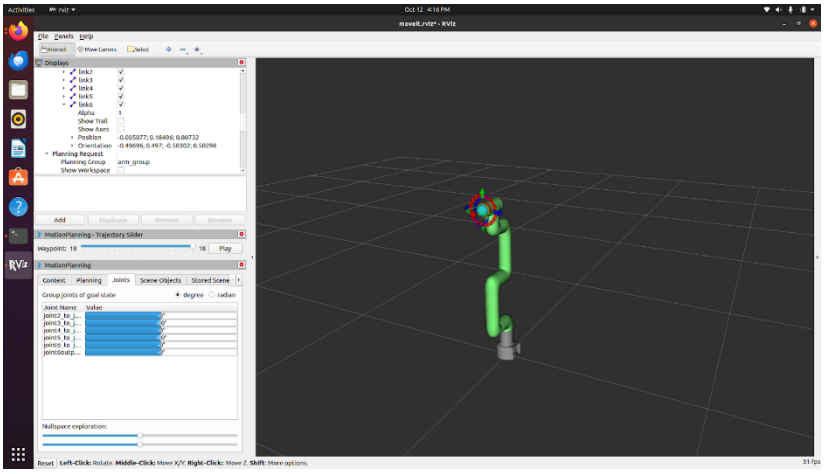

Figure 6: Digital Twin result for joint angle set [0, -90, 0, -90, 0, 0]

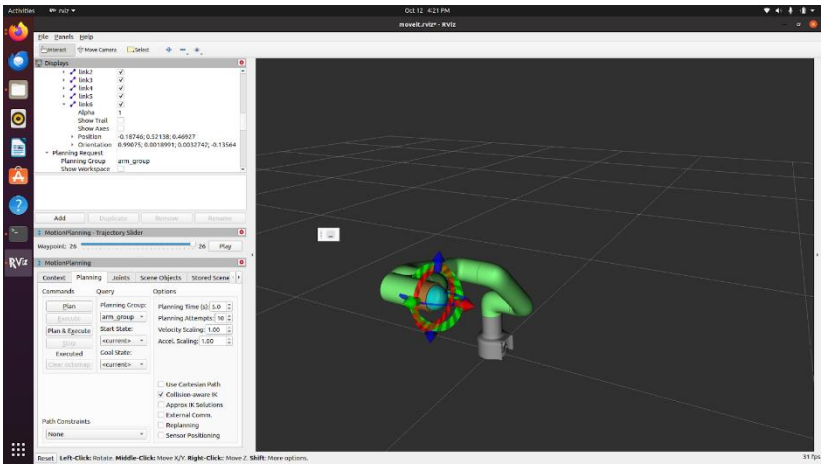

Figure 7: Digital Twin result for joint angle set 90, -45, 30, -90, 0, 0

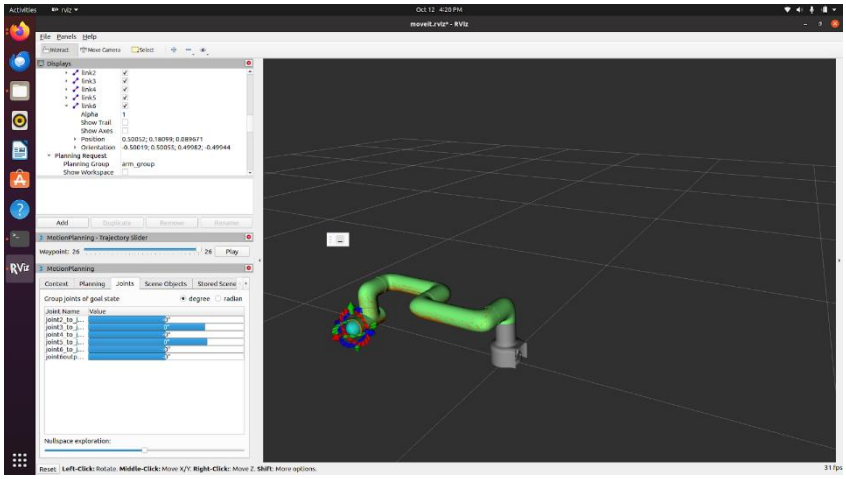

Figure 8: Digital Twin result for joint angle set 0, 0, 0, 0 ,0, 0

5. DIGITAL TWIN USING SIMSCAPE

The simulation of the MyCobot Pro 600 started with downloading the CAD model from the Elephant Robotics website. This model was imported into SolidWorks for further processing and the following steps were followed.

- Each joint in the assembly file was separated into a separate part and saved as a STEP file to allow independent movement of each joint during the simulation.
- After separating the joints, the parts were reassembled in SolidWorks. Joints were stacked in sequence from the base to the end effector, with appropriate mating of surfaces to ensure unhindered rotational motion as shown in figure: 9.

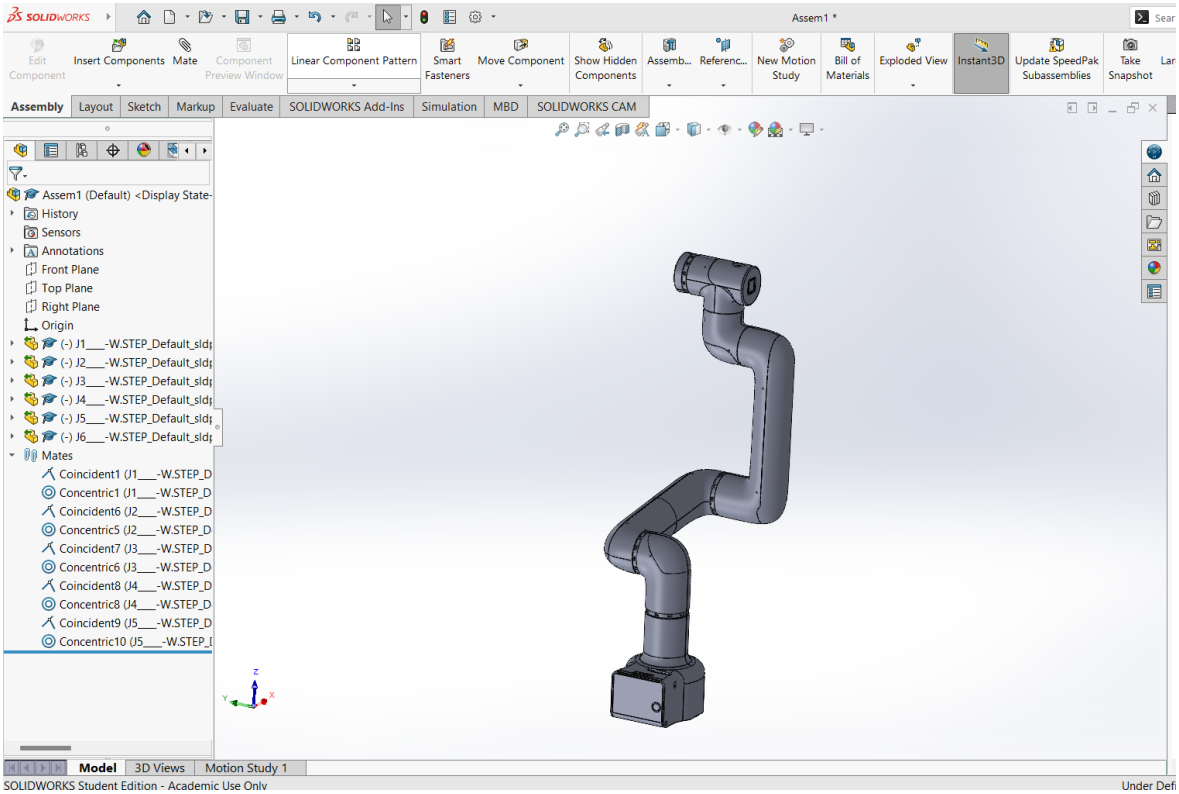

Figure 9: Assembly file after adding the joints and connecting them

- Local coordinate systems were assigned to each joint, defining the rotational axes necessary for tracking motion in the simulation.

- The assembled model was then exported as an XML file using the Simscape Multibody plugin within SolidWorks. This file format is required for importing the model into MATLAB for simulation.
- In MATLAB's Simscape environment, the XML file was imported (as shown in figure: 10), and slider gain blocks were added to the revolute joints. These blocks provided manual control over the joint angles, allowing for precise input during simulation.

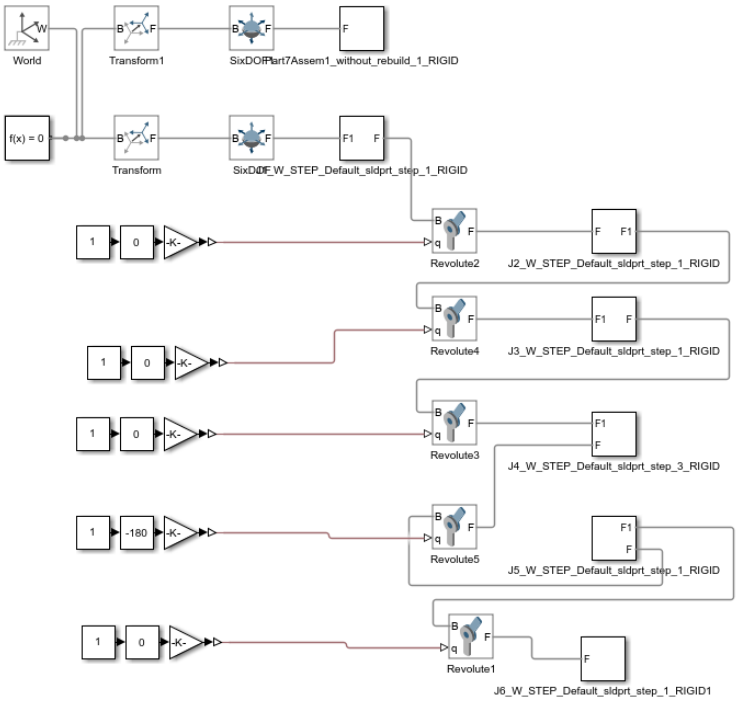

Figure 10: Simulink model obtained by importing the .xml file from Solidworks

- Three sets of joint angles were applied for testing: [0, -90, 0, -90, 0, 0], [90, -45, 30, -90, 0, 0], and [0, 0, 0, 0, 0, 0]. The robot's movement and configuration for each angle set were observed in Simscape and the corresponding results were recorded as shown in figure 11 – 13.

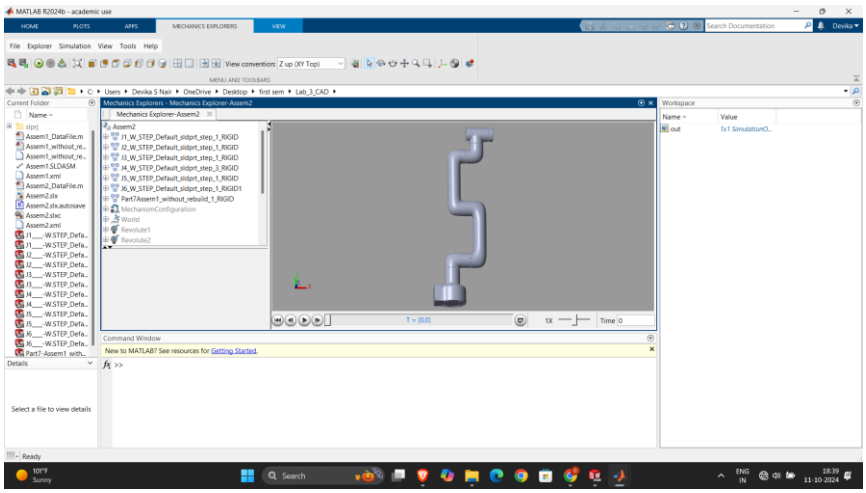

Figure 11: Configuration obtained for joint angle set 0, -90, 0, -90, 0,0

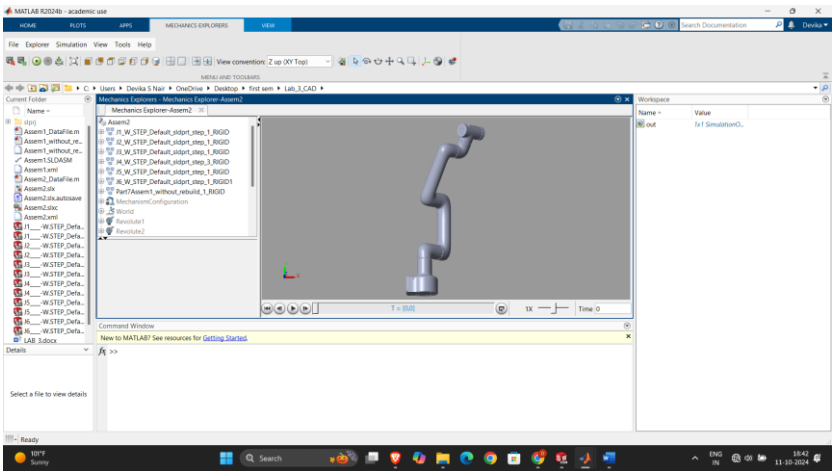

Figure 12: Configuration obtained for joint angles set 90, -45, 30, -90, 0, 0

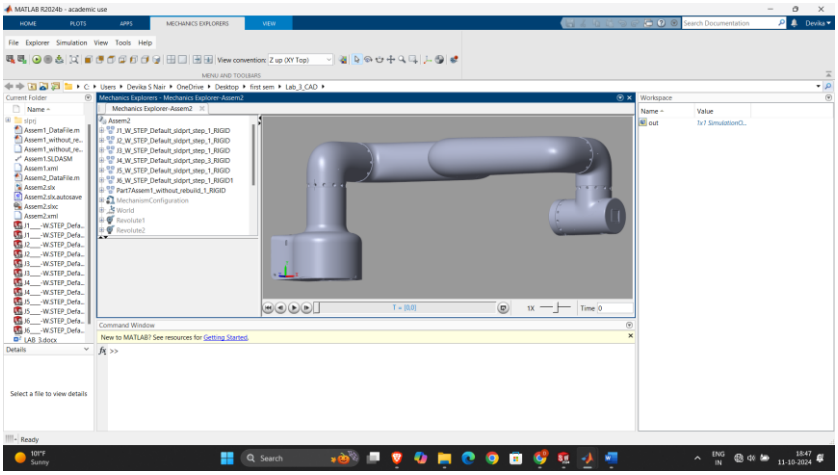

Figure 13: Configuration obtained for joint angle set 0, 0, 0, 0 ,0, 0

- The configurations generated in the simulation were then compared with the physical MyCobot Pro 600. The results showed that the simulated robot's movements matched the physical robot's joint configurations closely.
- The accurate replication of the physical robot's behaviour validated the simulation, confirming that the forward kinematics and joint control were implemented correctly in the digital twin.

6. RESULT COMPARISON

In this section, we compare the results we got from the physical robot and the ROS model. The following table illustrates the comparison between the end effector positions we got for the 3 sets of joint angles.

Table 4: Comparison of results

Joint Angle Set (deg)	Physical Model			ROS digital twin		
	X (mm)	Y (mm)	Z (mm)	X (mm)	Y (mm)	Z (mm)
[0, -90, 0, -90, 0, 0]	-11	185.64	816.90	-0.003	189.4	807.32
[90, -45, 30, -90, 0, 0]	-190.80	515.72	485.72	-187.51	521.34	469.36
[0, 0, 0, 0 ,0, 0]	499.448	187.211	99.948	500.16	181.01	88.82

The results obtained from the physical model and the digital twin simulations show a close correlation but with some noticeable differences in the final positions of the end effector for all the tested joint angle sets.
These variations can be attributed to factors like modelling approximations, numerical errors, or slight differences in how the physical robot and the digital twin handle kinematic constraints and transformations. However, the overall similarity in results indicates that the digital twin accurately represents the physical robot's movements with minor deviations.

7. CONCLUSION

The homogeneous transformation matrices calculated in MATLAB provided end-effector positions that closely matched the results from both the ROS digital twin and the physical robot. In the ROS simulation, the digital twin's end-effector coordinates were almost identical to those of the physical robot, with only minor differences across the tested joint angles. Similarly, the MATLAB Simscape simulation showed that the virtual robot's configurations aligned with those of the physical model, further confirming the accuracy of the simulation. Overall, both MATLAB and ROS simulations effectively mirrored the physical robot's behavior and movements.

REFERENCES

[1] Elephant Robotics, "MyCobot Pro 600 Documentation," Available: https://docs.elephantrobotics.com/docs/gitbook-en/2-serialproduct/2.3-myCobot_Pro_600/2.3-myCobot_Pro_600.html. Accessed: Sep. 14, 2024.

[2] Elephant Robotics, "Environment Building for ROS1," Available: https://docs.elephantrobotics.com/docs/pro600-en/12-ApplicationBaseROS/12.1-ROS1/12.1.2-EnvironmentBuilding.html. Accessed: Oct. 12, 2024.