ENPM662 – INTRODUCTION TO ROBOT MODELING

PROJECT 2 REPORT

HIGH SPEED CAMERA USING A MOBILE MANIPULATOR

AUTHORS:

AAQIB BARODAWALA - 119348710
SANDEEP THALAPANANE - 119402535

INDEX

INTRODUCTION:

There are some complex camera movements that cannot be achieved by a cinematographer/director using hands or even stabilizing gimbals which are used by many studios for acquiring professional shots. A *High-speed Robotic Camera* can achieve such cinematic shots by programming the speed and providing an input to reach the desired target and orientation.

A robotic camera is a type of camera that is capable of being remotely operated and programmed to capture images and videos. It can be used in a variety of applications, from surveillance to filmmaking. Robotic cameras are typically equipped with a variety of features, such as panning, tilting, and zooming capabilities. They can also be programmed to follow a specific path or to capture images at predetermined intervals. This makes them ideal for capturing images in difficult-to-reach places or for capturing images over long periods of time. Robotic cameras are also used in the film and television industry. They are used to capture shots that would be difficult or impossible to capture with a traditional camera. For example, they can be used to capture shots from high angles or to capture shots from a distance. They can also be used to capture shots in low light or in extreme weather conditions. Robotic cameras are becoming increasingly popular due to their versatility and ease of use. They are becoming more affordable and are being used in a variety of applications. They are an invaluable tool for filmmakers, security professionals, and anyone who needs to capture images in difficult-to-reach places.

Robotic cameras have revolutionized the film industry, allowing filmmakers to capture shots that were previously impossible. Robotic cameras are automated camera systems that can be programmed to move and capture shots with precision and accuracy. They are often used in situations where a human operator would be unable to physically move the camera, such as in tight spaces or when tracking a moving object. Robotic cameras are typically mounted on a robotic arm, which can be programmed to move the camera in a variety of directions. This allows filmmakers to capture shots from angles and perspectives that would be impossible with a traditional camera. The robotic arm can also be programmed to move the camera at a specific speed, allowing for smooth tracking shots. Robotic cameras are also equipped with advanced features such as remote control, allowing filmmakers to control the camera from a distance. This allows for more creative shots, as the camera can be moved and adjusted without the need for a human operator. Robotic cameras are also more reliable than traditional cameras, as they are less prone to human error. This makes them ideal for capturing shots in difficult or dangerous environments, such as underwater or in extreme weather conditions. Overall, robotic cameras have revolutionized the film industry, allowing filmmakers to capture shots that were previously impossible. They are reliable, and precise, and offer a variety of features that make them ideal for capturing creative shots.

There are many advantages of Robotic cameras over traditional cameras. They are more precise and accurate, allowing for greater control over the shot. They can also be programmed to move in a certain way, allowing for more creative shots. Additionally, robotic cameras are more reliable and require less maintenance than traditional cameras. They are also more cost-effective, as they can be used for multiple shots without having to purchase additional equipment. Finally, robotic cameras are more versatile, as they can be used in a variety of settings and for a variety of purposes.

APPLICATION:

Robotic cameras are used in a variety of applications, from industrial automation to entertainment. In industrial automation, robotic cameras are used to monitor and inspect production lines, detect defects, and provide feedback to operators. In entertainment, robotic cameras are used to capture dynamic shots and create unique angles for film and television.

Robotic cameras are also used in medical imaging. They can be used to capture images of the human body for diagnosis and treatment. They can also be used to monitor surgical procedures and provide feedback to surgeons.

Robotic cameras are also used in scientific research. They can be used to capture images of the night sky, monitor wildlife, and provide data for climate change research. Robotic cameras are becoming increasingly popular due to their versatility and cost-effectiveness. They can be used in a variety of applications and provide a cost-effective solution for capturing images and real-time data.

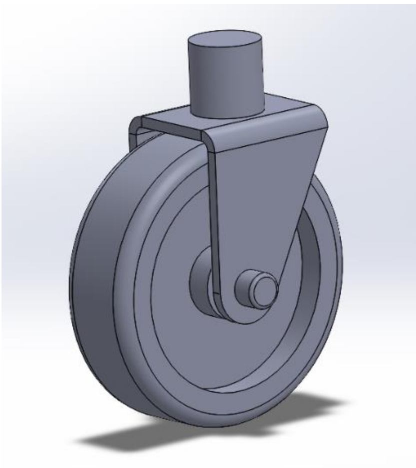
The robot we decided to model and simulate is designed for the application of capturing cinematic and professional shots.

Colossus, Kira, and Mia by Motorized Precision

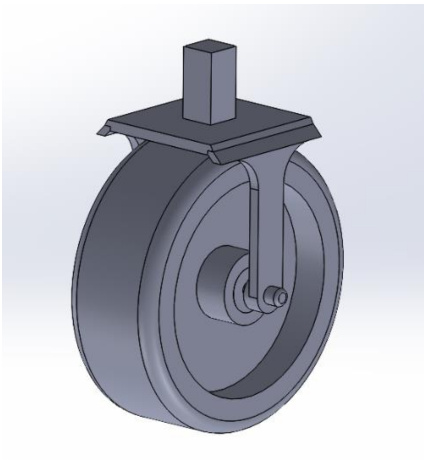
 

ROBOT TYPE AND DESCRIPTION:

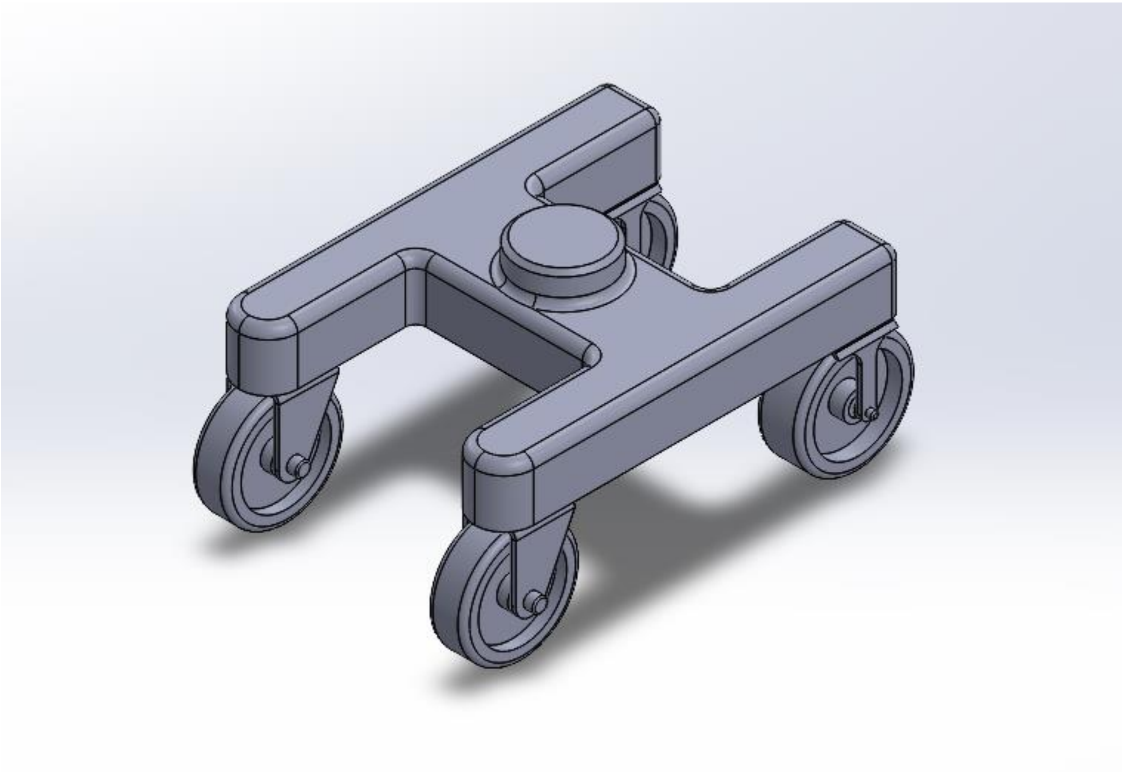
The robot we have designed has a 6-DOF UR3 manipulator mounted on a 4-wheel mobile platform. This makes the entire assembly an **8-DOF** robot. The mobile platform facilitates movement on a smooth flat surface, most likely the floor of a studio or even a flat road. Furthermore, the platform (which is in the shape of 'H') is specifically designed to achieve a reach that is lowest to the ground. This allows the robot to capture cinematic shots from the ground up.



Front Castor Wheel Assembly



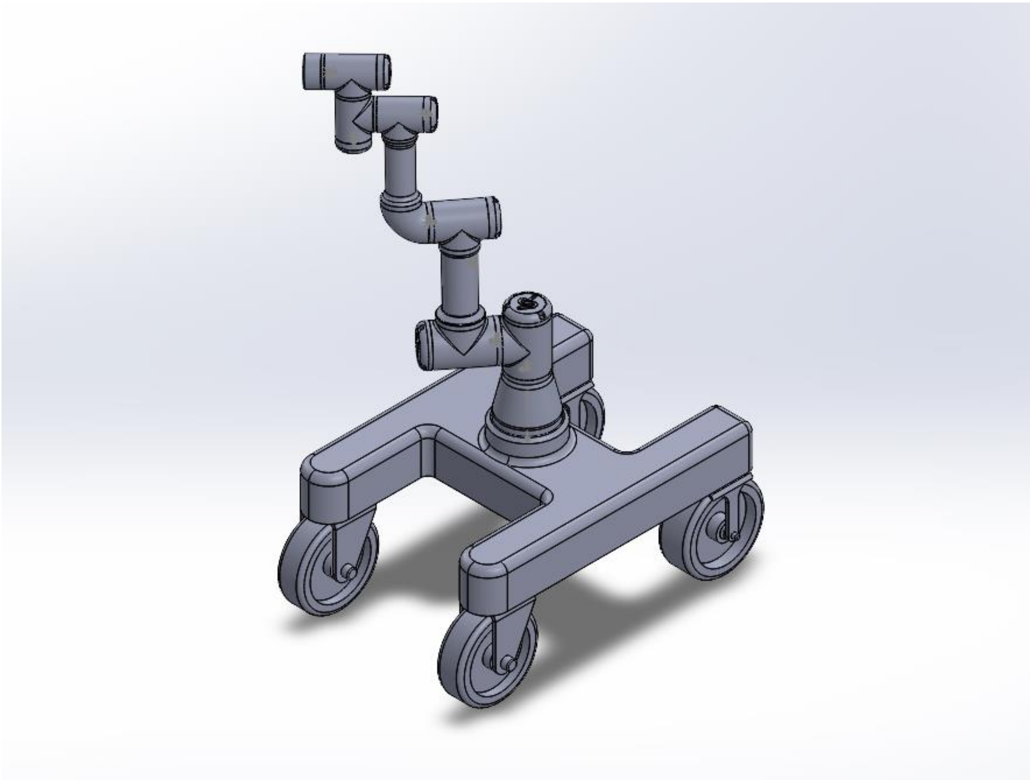
Front Castor Wheel Assembly



Mobile Platform Assembly

UR3 by Universal Robots

The Universal Robot UR3 is a lightweight, collaborative robot designed for use in a variety of applications. It can perform a wide range of tasks, from simple pick-and-place operations to assembly tasks that have complex trajectories. The UR3 is a 6-axis robotic arm with 6 revolute joints, a built-in vision system, and a user-friendly interface. It is designed to be easy to program and operate. It is designed to work safely alongside humans. The UR3 is ideal for small parts assembly, material handling, machine tending, and other light industrial applications. It is also capable of performing more complex tasks such as welding, gluing, and screw driving. The UR3 is designed to be flexible and reliable and is capable of working in a variety of environments. It is also equipped with a range of safety features, including force-sensing technology and a built-in emergency stop button. The UR3 is a powerful and versatile robot that can help increase productivity and reduce costs in a variety of applications.



High Speed Camera Assembly

Technical Specifications:

System Parameter	UR3
Degrees of Freedom	6 rotating joints
Payload	3 kg / 6.6 lbs
Repeatability	±0.1 mm / ±0.0039 in (4 mils)
Weight with cable	11 kg /24.3 lbs
Reach	500 mm / 19.7 in
Motion Range	Base: ± 360° Shoulder: ± 360° Elbow: ± 360° Wrist 1: ± 360° Wrist 2: ± 360° Wrist 3: Infinite
Maximum Speed	Base: ± 180°/Sec. Shoulder: ± 180°/Sec. Elbow: ± 180°/Sec. Wrist 1: ± 360°/Sec Wrist 2: ± 360°/Sec Wrist 3: ± 360°/Sec
Power Consumption	Min 90W, Typical 125W, Max 250W
Robot Mounting	Any
Ambient Temperature	0-50°*

Physical

Footprint	Ø 128mm
Materials	Aluminium, PP plastics
Tool connector type	M8
Cable length robot arm	6 m / 236 in
Weight with cable	11 kg /24.3 lbs

DH PARAMETERS:

The Denavit-Hartenberg convention has become a standard for selecting and assigning frames of reference in robot applications. For calculating the forward kinematics of the mobile manipulator, coordinate frames were assigned at each joint using Spong's naming convention.

The Denavit-Hartenberg (DH) parameters are a set of four parameters used to describe the relative position and orientation of two consecutive links in a robotic manipulator. The parameters are then used to define the relative position and orientation of the two links in a coordinate system. The parameters are the link length (d), the link twist (θ), the link offset (a), and the joint angle (α).

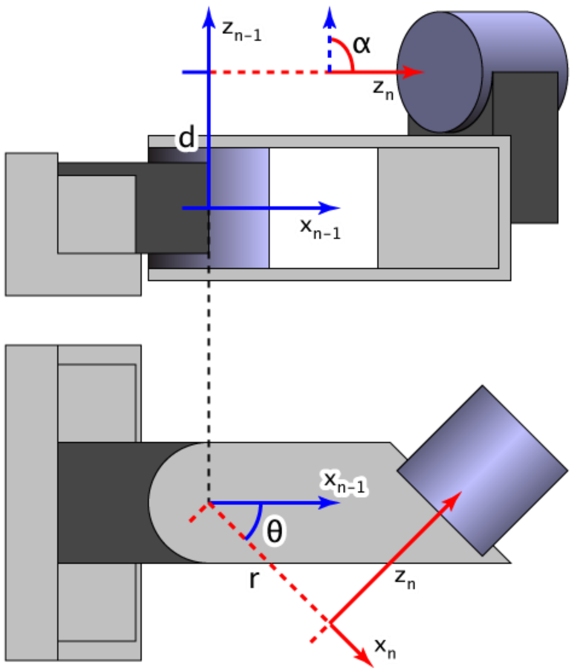
The link length (d) is the distance between origin of the first link and origin of the second link.

The link twist (θ) is the angle between z-axis of the first link and the z-axis of the second link.

The link offset (a) is the distance between x-axis of the first link and x-axis of the second link.

The joint angle (α) is the angle between x-axis of the first link and x-axis of the second link.

The DH parameters helps specify the relative location of frame n with respect to the previous frame. They are also used to calculate the forward and inverse kinematics of the robot, as well as the Jacobian matrix. The DH parameters are also used to define the motion of the robot, such as the trajectory of the end effector.



Illustrates the transformation parameters of a pair of reference frames laid out according to Denavit-Hartenberg convention

We can summarize the procedure for assigning coordinate frames based on the DH convention using the following steps:

Step 1: Locate and label the joint axes $z_0, \ldots, z_{n-1}$. The z-axis is the axis of rotation for a revolute joint

Step 2: Establish the base frame. Set the origin anywhere on the z_0-axis. The x_0 and y_0 axes are chosen conveniently to form a right-handed frame.

For $i = 1, \ldots, n-1$ perform Steps 3 to 5.

Step 3: Locate the origin o_i where the common normal to z_i and z_{i-1} intersects z_i. If z_i intersects z_{i-1}, locate o_i at this intersection. If z_i and z_{i-1} are parallel, locate o_i in any convenient position along z_i.

Step 4: Establish x_i along the common normal between z_{i-1} and z_i through o_i, or in the direction normal to the $z_{i-1} - z_i$ plane if z_{i-1} and z_i intersect.

Step 5: Establish y_i to complete a right-handed frame.

Step 6: Establish the end-effector frame $o_n x_n y_n z_n$. Assuming the n^{th} joint is revolute, set $z_n = a$ parallel to z_{n-1}. Establish the origin o_n conveniently along z_n, preferably at the centre of the gripper or at the tip of any tool that the manipulator may be carrying. Set $y_n = s$ in the direction of the gripper closure and set $x_n = n$ as $s \times a$. If the tool is not a simple gripper set x_n and y_n conveniently to form a right-handed frame.

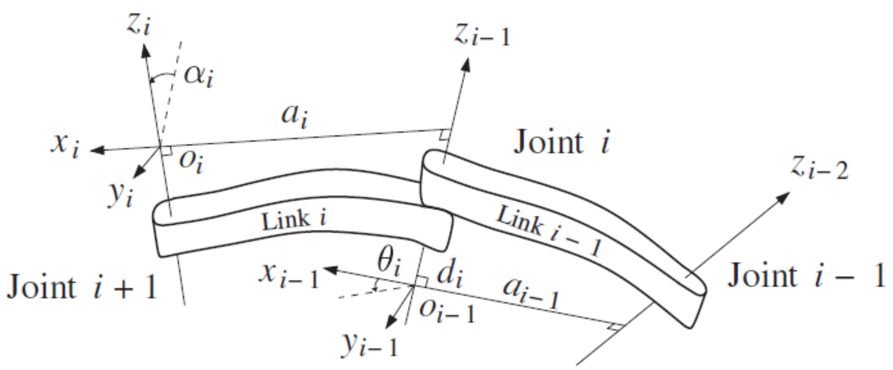
Step 7: Create a table of DH parameters $a_i, d_i, \alpha_i, \theta_i$.

$a_i =$ distance along x_i from the intersection of the x_i and z_{i-1} axes to o_i.

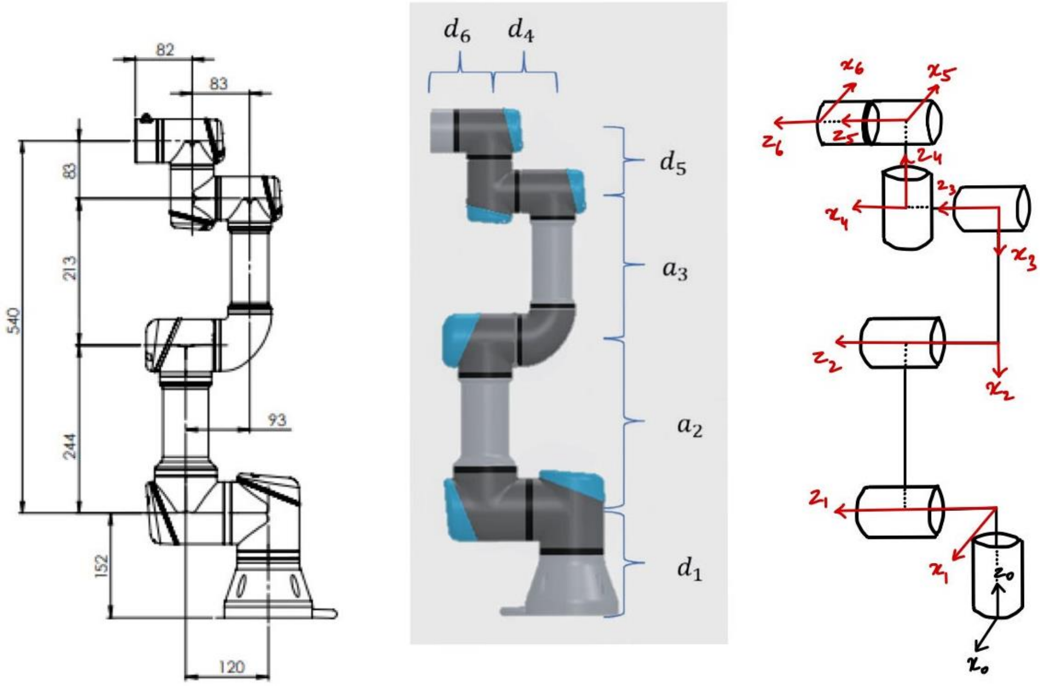
$d_i =$ distance along z_{i-1} from o_{i-1} to the intersection of the x_i and z_{i-1} axes. If the joint i is prismatic, then d_i is variable.

$\alpha_i =$ the angle from z_{i-1} to z_i measured about x_i.

$\theta_i =$ the angle from x_{i-1} to x_i measured about z_{i-1}. If joint i is revolute, θ_i is variable.



Denavit-Hartenberg frame assignment



All dimension is in mm

The computed DH table and coordinate frames for the UR3 manipulator is given as follows:

DH table:

Joint	a_i	d_i	α_i	θ_i	Offset
1	0	d_1	$\pi/2$	θ_1	0
2	a_2	0	0	θ_2	$-\pi/2$
3	a_3	0	0	θ_3	0
4	0	d_4	$\pi/2$	θ_4	$-\pi/2$
5	0	d_5	$-\pi/2$	θ_5	0
6	0	d_6	0	θ_6	0

FORWARD KINEMATICS AND VALIDATION:

Based on the DH convention, we can further derive the forward kinematics of a manipulator. With this convention, each individual homogeneous transformation A_i is represented as a product of four basic transformations:

$$\begin{aligned} A_i &= \mathrm{Rot}_{z,\theta_i}\mathrm{Trans}_{z,d_i}\mathrm{Trans}_{x,a_i}\mathrm{Rot}_{x,\alpha_i} \\ &= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i} & 0 & 0 \\ s_{\theta_i} & c_{\theta_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ &\quad \times \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c_{\alpha_i} & -s_{\alpha_i} & 0 \\ 0 & s_{\alpha_i} & c_{\alpha_i} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

$$= \begin{bmatrix} c_{\theta_i} & -s_{\theta_i}c_{\alpha_i} & s_{\theta_i}s_{\alpha_i} & a_i c_{\theta_i} \\ s_{\theta_i} & c_{\theta_i}c_{\alpha_i} & -c_{\theta_i}s_{\alpha_i} & a_i s_{\theta_i} \\ 0 & s_{\alpha_i} & c_{\alpha_i} & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We can then construct the A_i matrices by substituting the DH parameters in the above equation. The final transformation matrix is then given by the formula:

$$T_n^0 = A_1^0(q_1) \times A_2^1(q_2) \times A_3^2(q_3) \times \ldots \times A_n^{n-1}(q_n)$$

Where q_i represents the individual joint angles or θ_i's.

To calculate the forward kinematics, we used the help of a python script (given in appendix) to calculate and validate the final transformation matrix. We found that the end effector position and orientation match with the Peter Corke's MATLAB Toolbox's output with joint angles given as follows:

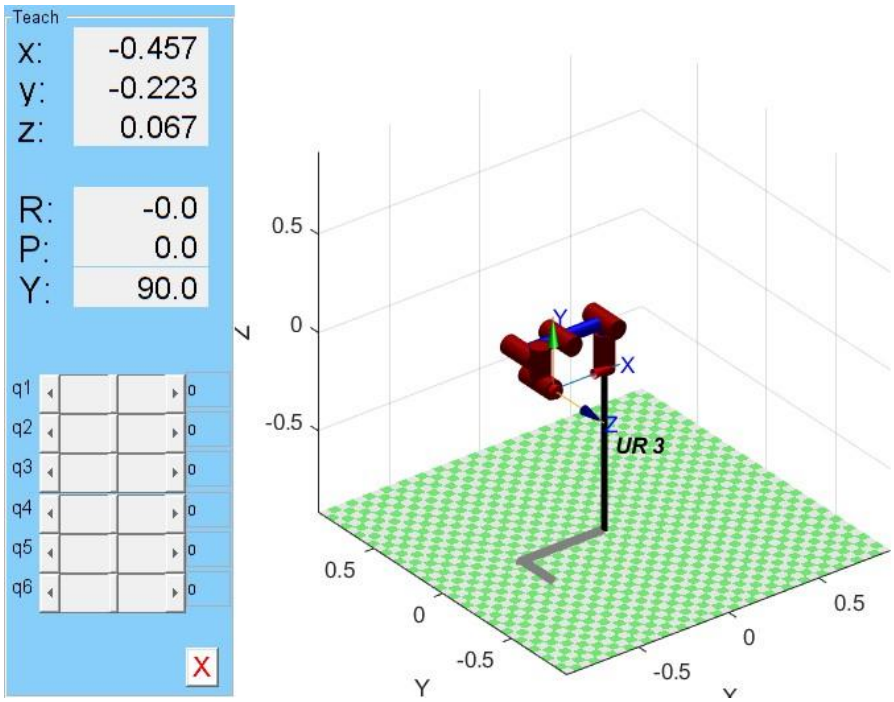
$$q_i = [0\ 0\ 0\ 0\ 0\ 0]$$

$$\text{and } q_i = [0\ -90\ 0\ -90\ 0\ 0] \text{ (with offset)}$$

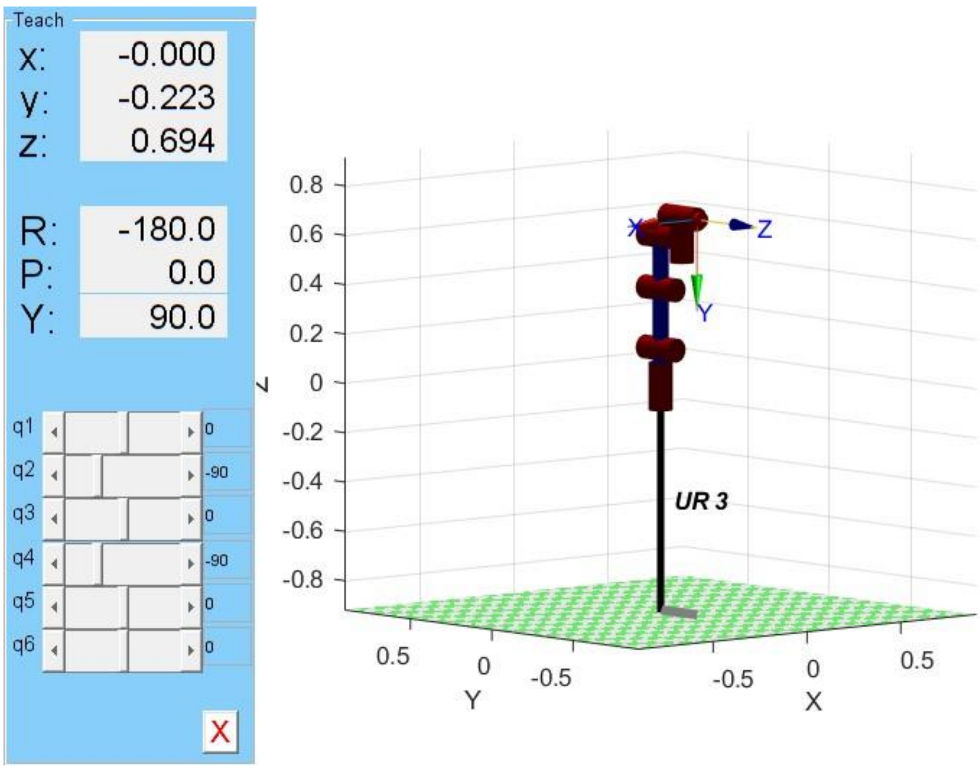
Python Script "forward kinematics" output for first set of joint angles:

End effector position:

[-0.457]
[-0.22315]
[0.0665]
Roll(R): 0
Pitch(P): 0
Yaw(Y): pi/2



UR3 in home position



UR3 with offset

Final Transformation matrix:

$[(((-\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1) + \cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_2)*\cos(\theta_1)*\cos(\theta_3) -$
$\sin(\theta_3)*\cos(\theta_1)*\cos(\theta_2))*\sin(\theta_4))*\cos(\theta_5) + \sin(\theta_1)*\sin(\theta_5))*\cos(\theta_6) + (-(-\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1)$
$+ \cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (-\sin(\theta_2)*\cos(\theta_1)*\cos(\theta_3) -$
$\sin(\theta_3)*\cos(\theta_1)*\cos(\theta_2))*\cos(\theta_4))*\sin(\theta_6)\ \ -(((-\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1) +$
$\cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_2)*\cos(\theta_1)*\cos(\theta_3) -$
$\sin(\theta_3)*\cos(\theta_1)*\cos(\theta_2))*\sin(\theta_4))*\cos(\theta_5) + \sin(\theta_1)*\sin(\theta_5))*\sin(\theta_6) + (-(-\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1)$
$+ \cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (-\sin(\theta_2)*\cos(\theta_1)*\cos(\theta_3) -$
$\sin(\theta_3)*\cos(\theta_1)*\cos(\theta_2))*\cos(\theta_4))*\cos(\theta_6)\ \ -((-\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1) +$
$\cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_2)*\cos(\theta_1)*\cos(\theta_3) -$
$\sin(\theta_3)*\cos(\theta_1)*\cos(\theta_2))*\sin(\theta_4))*\sin(\theta_5) + \sin(\theta_1)*\cos(\theta_5)\ \ a2*\cos(\theta_1)*\cos(\theta_2) -$
$a3*\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1) + a3*\cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3) + d4*\sin(\theta_1) + d5*((-$
$\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1) + \cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) - (-\sin(\theta_2)*\cos(\theta_1)*\cos(\theta_3) -$
$\sin(\theta_3)*\cos(\theta_1)*\cos(\theta_2))*\cos(\theta_4)) + d6*(-((-\sin(\theta_2)*\sin(\theta_3)*\cos(\theta_1) +$
$\cos(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_2)*\cos(\theta_1)*\cos(\theta_3) -$
$\sin(\theta_3)*\cos(\theta_1)*\cos(\theta_2))*\sin(\theta_4))*\sin(\theta_5) + \sin(\theta_1)*\cos(\theta_5))]$

[
]

$[(((-\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3) + \sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_1)*\sin(\theta_2)*\cos(\theta_3) -$
$\sin(\theta_1)*\sin(\theta_3)*\cos(\theta_2))*\sin(\theta_4))*\cos(\theta_5) - \sin(\theta_5)*\cos(\theta_1))*\cos(\theta_6) + (-(-\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3)$
$+ \sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (-\sin(\theta_1)*\sin(\theta_2)*\cos(\theta_3) -$
$\sin(\theta_1)*\sin(\theta_3)*\cos(\theta_2))*\cos(\theta_4))*\sin(\theta_6)\ \ -(((-\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3) +$
$\sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_1)*\sin(\theta_2)*\cos(\theta_3) -$
$\sin(\theta_1)*\sin(\theta_3)*\cos(\theta_2))*\sin(\theta_4))*\cos(\theta_5) - \sin(\theta_5)*\cos(\theta_1))*\sin(\theta_6) + (-(-\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3)$
$+ \sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (-\sin(\theta_1)*\sin(\theta_2)*\cos(\theta_3) -$
$\sin(\theta_1)*\sin(\theta_3)*\cos(\theta_2))*\cos(\theta_4))*\cos(\theta_6)\ \ -((-\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3) +$
$\sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_1)*\sin(\theta_2)*\cos(\theta_3) -$
$\sin(\theta_1)*\sin(\theta_3)*\cos(\theta_2))*\sin(\theta_4))*\sin(\theta_5) - \cos(\theta_1)*\cos(\theta_5)\ \ a2*\sin(\theta_1)*\cos(\theta_2) -$
$a3*\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3) + a3*\sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3) - d4*\cos(\theta_1) + d5*((-$
$\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3) + \sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) - (-\sin(\theta_1)*\sin(\theta_2)*\cos(\theta_3) -$
$\sin(\theta_1)*\sin(\theta_3)*\cos(\theta_2))*\cos(\theta_4)) + d6*(-((-\sin(\theta_1)*\sin(\theta_2)*\sin(\theta_3) +$
$\sin(\theta_1)*\cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (-\sin(\theta_1)*\sin(\theta_2)*\cos(\theta_3) -$
$\sin(\theta_1)*\sin(\theta_3)*\cos(\theta_2))*\sin(\theta_4))*\sin(\theta_5) - \cos(\theta_1)*\cos(\theta_5))]$

[
]

$[\qquad ((-\sin(\theta_2)*\sin(\theta_3) + \cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (\sin(\theta_2)*\cos(\theta_3) +$
$\sin(\theta_3)*\cos(\theta_2))*\cos(\theta_4))*\cos(\theta_5)*\cos(\theta_6) + ((-\sin(\theta_2)*\sin(\theta_3) + \cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) -$
$(\sin(\theta_2)*\cos(\theta_3) + \sin(\theta_3)*\cos(\theta_2))*\sin(\theta_4))*\sin(\theta_6)$
$-((-\sin(\theta_2)*\sin(\theta_3) + \cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (\sin(\theta_2)*\cos(\theta_3) +$
$\sin(\theta_3)*\cos(\theta_2))*\cos(\theta_4))*\sin(\theta_6)*\cos(\theta_5) + ((-\sin(\theta_2)*\sin(\theta_3) + \cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) -$

$(\sin(\theta_2)*\cos(\theta_3) + \sin(\theta_3)*\cos(\theta_2))*\sin(\theta_4))*\cos(\theta_6)$ $-((-\sin(\theta_2)*\sin(\theta_3) + \cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (\sin(\theta_2)*\cos(\theta_3) + \sin(\theta_3)*\cos(\theta_2))*\cos(\theta_4))*\sin(\theta_5)$
$a2*\sin(\theta_2) + a3*\sin(\theta_2)*\cos(\theta_3) + a3*\sin(\theta_3)*\cos(\theta_2) + d1 + d5*(-(-\sin(\theta_2)*\sin(\theta_3) + \cos(\theta_2)*\cos(\theta_3))*\cos(\theta_4) + (\sin(\theta_2)*\cos(\theta_3) + \sin(\theta_3)*\cos(\theta_2))*\sin(\theta_4)) - d6*((-\sin(\theta_2)*\sin(\theta_3) + \cos(\theta_2)*\cos(\theta_3))*\sin(\theta_4) + (\sin(\theta_2)*\cos(\theta_3) + \sin(\theta_3)*\cos(\theta_2))*\cos(\theta_4))*\sin(\theta_5)$
]

[
]

[
0
0
0
1
]

INVERSE KINEMATICS AND VALIDATION:

In the real-world applications, we know the position and orientation and we should calculate the joint angles to reach that pose. This process is called inverse kinematics. Below, we will be finding the joint angles using the final transformation matrix and geometric method.

$$ {}^{0}T_{6} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ {}^{0}_{1}T^{-1} \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = {}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T\,{}^{4}_{5}T\,{}^{5}_{6}T $$

$$ \begin{bmatrix} r_{11}C_1 + r_{21}S_1 & r_{12}C_1 + r_{22}S_1 & r_{13}C_1 + r_{23}S_1 & p_xC_1 + p_yS_1 \\ r_{31} & r_{32} & r_{33} & p_z \\ r_{11}S_1 - r_{21}C_1 & r_{12}S_1 - r_{22}C_1 & r_{13}S_1 - r_{23}C_1 & p_xS_1 - p_yC_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

$$ = \begin{bmatrix} C_{234}C_5C_6 - S_{234}S_6 & -S_{234}C_6 - C_{234}C_5C_6 & -C_{234}S_5 & d_5S_{234} + a_3C_{23} + a_2C_2 + d_6S_5C_{234} \\ C_{234}S_6 + S_{234}C_5C_6 & C_{234}C_6 - S_{234}C_5C_6 & -S_{234}S_5 & d_5C_{234} + a_3S_{23} + a_2S_2 - d_6S_5S_{234} \\ C_6S_5 & -S_5S_6 & C_5 & d_4 + d_6C_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Comparing the 3^{rd} row & 3^{rd} column elements of each equation & 3^{rd} row 4^{th} column elements of each equation to obtain θ_1.

C – cos

S – sin

$$r_{13} S_1 - r_{23} C_1 = C_5 \quad -①$$

$$p_x S_1 - p_y C_1 = d_4 + d_6 C_5 \quad -②$$

sub ① into ②

$$p_x S_1 - p_y C_1 = d_4 + d_6 (r_{13} S_1 - r_{23} C_1)$$

$$(p_x - d_6 r_{13}) S_1 + (d_6 r_{23} - p_y) C_1 = d_4$$

Assuming $\rho \sin\phi = d_6 r_{23} - p_y$

$$\rho \cos\phi = p_x - d_6 r_{13}$$

where $\phi = \text{atan2}(d_6 r_{23} - p_y,\ p_x - d_6 r_{13})$

$$\rho = (d_6 r_{23} - p_y)^2 + (p_x - d_6 r_{13})$$

$$\rho \cos\phi S_1 + \rho \sin\phi C_1 = d_4$$

$$\cos\phi S_1 + \sin\phi C_1 = \frac{d_4}{\rho}$$

$$\sin(\theta_1 + \phi) = \frac{d_4}{\rho}$$

$$\phi + \theta_1 = \text{atan2}\left(\frac{d_4}{\rho}, \pm\sqrt{1 - \frac{d_4^2}{\rho^2}}\right)$$

Combining ϕ and $\theta_1+\phi$ we get

$$\theta_1 = \operatorname{atan2}\left(d_4, \pm\sqrt{(d_6 r_{23} - p_y)^2 + (p_x - d_6 r_{13})^2}\right)$$
$$- \operatorname{atan2}(d_6 r_{23} - p_y, \; p_x - d_6 r_{13})$$

Next comparing 3rd row, column 1 and 3rd row 2nd column of each side to get θ_5

$$r_{11} S_1 - r_{21} C_1 = C_6 S_5 \quad - (3)$$
$$r_{12} S_1 - r_{22} C_1 = -S_5 S_6 - (4)$$

squaring both the eqns and adding

$$(r_{11} S_1 - r_{21} C_1)^2 + (r_{12} S_1 - r_{22} C_1)^2 = (C_6 S_5)^2 + (-S_5 S_6)^2$$
$$= (C_6^2 + S_6^2) S_5^2 = S_5^2$$

$$S_5 = \pm\sqrt{(r_{11} S_1 - r_{21} C_1)^2 + (r_{12} S_1 - r_{22} C_1)^2}$$

$$\theta_5 = \sin^{-1}\left(\pm\sqrt{(r_{11} S_1 - r_{21} C_1)^2 + (r_{12} S_1 - r_{22} C_1)^2}\right)$$

Next using (3) & (4)

$$\frac{r_{11} S_1 - r_{21} C_1}{S_5} = C_6$$

$$-\frac{r_{12} S_1 - r_{22} C_1}{S_5} = S_6$$

$$\theta_6 = \text{atan2}\left(\frac{r_{11} S_1 - r_{21} C_1}{S_5}, -\frac{r_{12} S_1 - r_{22} C_1}{S_5}\right)$$

Next $\theta_{234} = \theta_2 + \theta_3 + \theta_4$ is derived from comparing two new equations

$$^0T_4 = {}^0T_6 \cdot \left({}^5H_6^{-1}\right)\left({}^4H_5^{-1}\right)$$

The 3^{rd} row 1^{st} column of $^0T_4$ is $\sin\theta_{324}$

3^{rd} row 3^{rd} column of $^0T_4$ is $-\cos\theta_{234}$

3^{rd} row 1^{st} column of right side is

$$r_{31} C_5 C_6 - r_{33} S_5 - r_{32} S_6 C_5$$

3^{rd} row 3^{rd} column of right side is

$$-r_{32} C_6 - r_{31} S_6$$

$$\sin\theta_{234} = r_{31} C_5 C_6 - r_{33} S_5 - r_{32} S_6 C_5$$

$$\cos\theta_{234} = -r_{32} C_6 - r_{31} S_6$$

$$\theta_{234} = \text{atan2}\left(r_{31} C_5 C_6 - r_{33} S_5 - r_{32} S_6 C_5, -r_{32} C_6 - r_{31} S_6\right)$$

Finding q_2 and q_3 from the geometry of robot

The origin values of 4^{th} joint are

$$P_{x_4} = P_x - r_{31} d_6 + d_5 r_{21} C_6 + d_5 r_{11} S_6$$

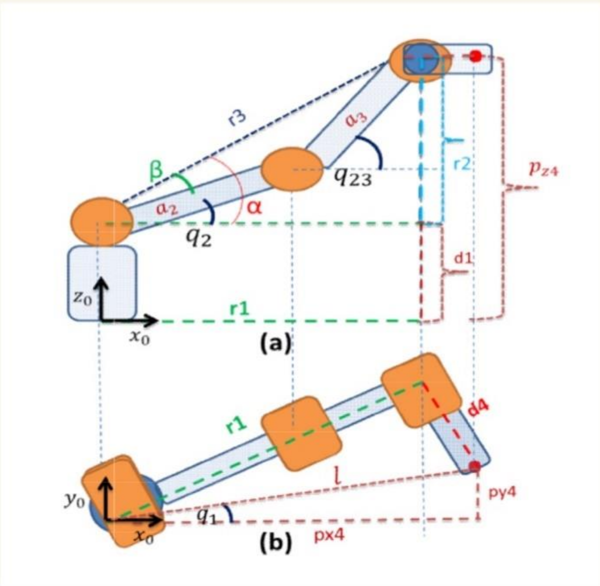
$$P_{y_4} = P_y - r_{32} d_6 + d_5 r_{22} C_6 + d_5 r_{12} S_6$$

$$P_{z_4} = P_z - r_{33} d_6 + d_5 r_{22} C_6 + d_5 r_{13} S_6$$

we can get solution of P_4 using θ_6.

From the geometry

$$L = \sqrt{P_{x_4}^2 + P_{y_4}^2}$$



$$r_1 = \sqrt{L^2 - d_4^2}$$

$$r_2 = P_{z_4} - d_1$$

$$r_3 = \sqrt{r_1^2 + r_2^2}$$

From the above figure we can get

$$r_1 = a_2 C_2 + a_3 C_{23}$$

$$r_2 = a_2 S_2 + a_3 S_{23}$$

Squaring the above eqns & adding

$$r_1^2 + r_2^2 = (a_2 C_2 + a_3 C_{23})^2 + (a_2 S_2 + a_3 S_{23})^2$$

$$= (a_2 C_2)^2 + (a_3 C_{23})^2 + 2 a_2 a_3 C_2 C_{23}$$

$$+ (a_2 S_2)^2 + (a_3 S_{23})^2 + 2 a_2 a_3 S_2 S_{23}$$

$$r_1^2 + r_2^2 - (a_2^2 + a_3^2) = 2 a_2 a_3 (C_2 C_{23} + S_2 S_{23})$$

$$= 2 a_2 a_3 \cos(\theta_{23} - \theta_2)$$

$$= 2 a_2 a_3 \cos(\theta_3)$$

$$\cos\theta_3 = \frac{r_1^2 + r_2^2 - (a_2^2 + a_3^2)}{2 a_2 a_3}$$

$$\theta_3 = \cos^{-1}\left(\frac{r_1^2 + r_2^2 - (a_2^2 + a_3^2)}{2 a_2 a_3}\right)$$

next finding θ_2

From the figure

$$\alpha = \text{atan2}(r_2, r_1)$$

$$\beta = \text{atan2}(a_3 S_3\, a_2 + a_3 C_3)$$

$$\theta_2 = \alpha - \beta$$

$$\theta_4 = \theta_{234} - \theta_3 - \theta_2$$

We will be getting 8 different solutione for the inverse kinematics.

So we are using inverse velocity kinematics to plot a circle.

$$\dot{q} = J^{-1}(\dot{x})$$

where $\dot{q}$ is derivative of joint angles

Finding Jacobian using the Second method

$$J = \begin{bmatrix} \frac{\partial X_p}{\partial q_1} & \frac{\partial X_p}{\partial q_2} & \frac{dX_p}{dq_3} & \frac{\partial X_p}{\partial q_4} & \frac{\partial X_p}{\partial q_5} & \frac{\partial X_p}{dq_6} \\ {}^{0}Z_1 & {}^{0}Z_2 & {}^{0}Z_3 & {}^{0}Z_4 & {}^{0}Z_5 & {}^{0}Z_6 \end{bmatrix}$$

where X_p is the 4^{th} column of the final transformation matrix.

${}^{0}Z_i$ is the 3^{rd} column of the ${}^{0}T_i$ transformation matrix

plotting a circle trajectory on the y plane of 0.025m radius

Circle equation in parametric form

$$y = 0.025\cos\theta$$

$$z = 0.025\sin\theta$$

$$V_y = \dot{y} = -0.025 \cdot \cos\theta \cdot \dot{\theta} \qquad \dot{\theta} = \frac{d\theta}{dt} = \frac{2\pi}{5}$$

$$V_z = \dot{z} = 0.025 \cdot \sin\theta \cdot \dot{\theta}$$

$$0, \omega_x, \omega_y, \omega_z = 0$$

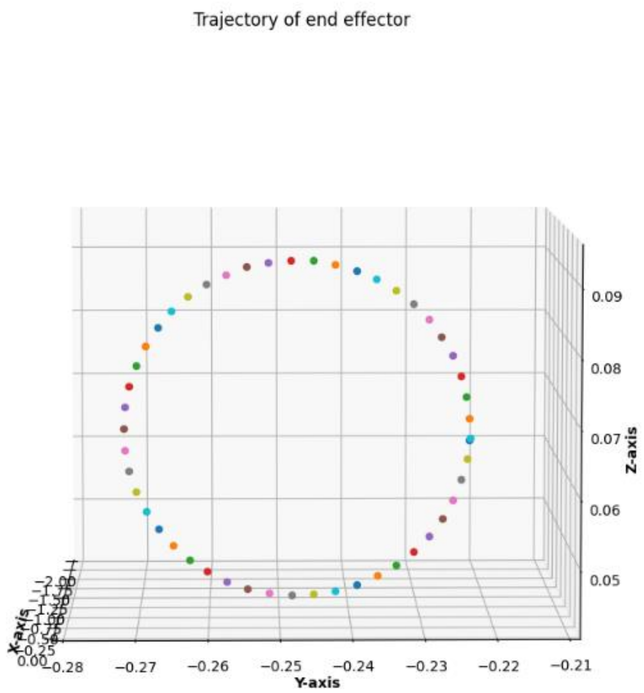
$$\dot{X} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$

$$q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \end{bmatrix} \qquad \dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix}$$

$$\dot{q} = J^{-1}(q) \cdot \dot{X} \qquad \Delta t = \frac{T}{N} = \frac{5}{50}$$

$$\dot{q} = q_{i-1} + \dot{q}_i \, \Delta t$$

The end effector trajectory is plotted using a python script (Appendix 2) using the jacobian matrix method.



Jacobian Matrix:

[(-0.0921*(sin(theta1)*sin(theta2)*sin(theta3) -
sin(theta1)*cos(theta2)*cos(theta3))*cos(theta4) -
0.0921*(sin(theta1)*sin(theta2)*cos(theta3) +
sin(theta1)*sin(theta3)*cos(theta2))*sin(theta4))*sin(theta5) +
(0.08535*sin(theta1)*sin(theta2)*sin(theta3) -
0.08535*sin(theta1)*cos(theta2)*cos(theta3))*sin(theta4) + (-
0.08535*sin(theta1)*sin(theta2)*cos(theta3) -
0.08535*sin(theta1)*sin(theta3)*cos(theta2))*cos(theta4) -
0.2132*sin(theta1)*sin(theta2)*sin(theta3) + 0.2132*sin(theta1)*cos(theta2)*cos(theta3) +
0.24355*sin(theta1)*cos(theta2) + 0.0921*cos(theta1)*cos(theta5) + 0.13105*cos(theta1)
(-0.0921*(sin(theta2)*sin(theta3)*cos(theta1) -
cos(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - 0.0921*(-
sin(theta2)*cos(theta1)*cos(theta3) -
sin(theta3)*cos(theta1)*cos(theta2))*cos(theta4))*sin(theta5) + (-
0.08535*sin(theta2)*sin(theta3)*cos(theta1) +
0.08535*cos(theta1)*cos(theta2)*cos(theta3))*cos(theta4) + (-
0.08535*sin(theta2)*cos(theta1)*cos(theta3) -
0.08535*sin(theta3)*cos(theta1)*cos(theta2))*sin(theta4) +
0.2132*sin(theta2)*cos(theta1)*cos(theta3) + 0.24355*sin(theta2)*cos(theta1) +
0.2132*sin(theta3)*cos(theta1)*cos(theta2) (-0.0921*(sin(theta2)*sin(theta3)*cos(theta1)
- cos(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - 0.0921*(-
sin(theta2)*cos(theta1)*cos(theta3) -
sin(theta3)*cos(theta1)*cos(theta2))*cos(theta4))*sin(theta5) + (-
0.08535*sin(theta2)*sin(theta3)*cos(theta1) +
0.08535*cos(theta1)*cos(theta2)*cos(theta3))*cos(theta4) + (-
0.08535*sin(theta2)*cos(theta1)*cos(theta3) -
0.08535*sin(theta3)*cos(theta1)*cos(theta2))*sin(theta4) +
0.2132*sin(theta2)*cos(theta1)*cos(theta3) + 0.2132*sin(theta3)*cos(theta1)*cos(theta2)
(0.0921*(-sin(theta2)*sin(theta3)*cos(theta1) +
cos(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - 0.0921*(-
sin(theta2)*cos(theta1)*cos(theta3) -
sin(theta3)*cos(theta1)*cos(theta2))*cos(theta4))*sin(theta5) + (-
0.08535*sin(theta2)*sin(theta3)*cos(theta1) +
0.08535*cos(theta1)*cos(theta2)*cos(theta3))*cos(theta4) -
(0.08535*sin(theta2)*cos(theta1)*cos(theta3) +
0.08535*sin(theta3)*cos(theta1)*cos(theta2))*sin(theta4) (-0.0921*(-
sin(theta2)*sin(theta3)*cos(theta1) + cos(theta1)*cos(theta2)*cos(theta3))*cos(theta4) -
0.0921*(-sin(theta2)*cos(theta1)*cos(theta3) -
sin(theta3)*cos(theta1)*cos(theta2))*sin(theta4))*cos(theta5) -
0.0921*sin(theta1)*sin(theta5)
0   ]

[
]

[(-0.0921*(-sin(theta2)*sin(theta3)*cos(theta1) +
cos(theta1)*cos(theta2)*cos(theta3))*cos(theta4) - 0.0921*(-
sin(theta2)*cos(theta1)*cos(theta3) -
sin(theta3)*cos(theta1)*cos(theta2))*sin(theta4))*sin(theta5) + (-
0.08535*sin(theta2)*sin(theta3)*cos(theta1) +
0.08535*cos(theta1)*cos(theta2)*cos(theta3))*sin(theta4) +
(0.08535*sin(theta2)*cos(theta1)*cos(theta3) +
0.08535*sin(theta3)*cos(theta1)*cos(theta2))*cos(theta4) +
0.0921*sin(theta1)*cos(theta5) + 0.13105*sin(theta1) +
0.2132*sin(theta2)*sin(theta3)*cos(theta1) - 0.2132*cos(theta1)*cos(theta2)*cos(theta3) -
0.24355*cos(theta1)*cos(theta2) (-0.0921*(sin(theta1)*sin(theta2)*sin(theta3) -
sin(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - 0.0921*(-
sin(theta1)*sin(theta2)*cos(theta3) -
sin(theta1)*sin(theta3)*cos(theta2))*cos(theta4))*sin(theta5) + (-
0.08535*sin(theta1)*sin(theta2)*sin(theta3) +
0.08535*sin(theta1)*cos(theta2)*cos(theta3))*cos(theta4) + (-
0.08535*sin(theta1)*sin(theta2)*cos(theta3) -
0.08535*sin(theta1)*sin(theta3)*cos(theta2))*sin(theta4) +
0.2132*sin(theta1)*sin(theta2)*cos(theta3) + 0.24355*sin(theta1)*sin(theta2) +
0.2132*sin(theta1)*sin(theta3)*cos(theta2) (-0.0921*(sin(theta1)*sin(theta2)*sin(theta3) -
sin(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - 0.0921*(-
sin(theta1)*sin(theta2)*cos(theta3) -
sin(theta1)*sin(theta3)*cos(theta2))*cos(theta4))*sin(theta5) + (-
0.08535*sin(theta1)*sin(theta2)*sin(theta3) +
0.08535*sin(theta1)*cos(theta2)*cos(theta3))*cos(theta4) + (-
0.08535*sin(theta1)*sin(theta2)*cos(theta3) -
0.08535*sin(theta1)*sin(theta3)*cos(theta2))*sin(theta4) +
0.2132*sin(theta1)*sin(theta2)*cos(theta3) + 0.2132*sin(theta1)*sin(theta3)*cos(theta2)
(0.0921*(-sin(theta1)*sin(theta2)*sin(theta3) +
sin(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - 0.0921*(-
sin(theta1)*sin(theta2)*cos(theta3) -
sin(theta1)*sin(theta3)*cos(theta2))*cos(theta4))*sin(theta5) + (-
0.08535*sin(theta1)*sin(theta2)*sin(theta3) +
0.08535*sin(theta1)*cos(theta2)*cos(theta3))*cos(theta4) -
(0.08535*sin(theta1)*sin(theta2)*cos(theta3) +
0.08535*sin(theta1)*sin(theta3)*cos(theta2))*sin(theta4) (-0.0921*(-
sin(theta1)*sin(theta2)*sin(theta3) + sin(theta1)*cos(theta2)*cos(theta3))*cos(theta4) -
0.0921*(-sin(theta1)*sin(theta2)*cos(theta3) -
sin(theta1)*sin(theta3)*cos(theta2))*sin(theta4))*cos(theta5) +
0.0921*sin(theta5)*cos(theta1)
0]

[
]

[
0
(-0.0921*(-sin(theta2)*sin(theta3) + cos(theta2)*cos(theta3))*cos(theta4) - 0.0921*(-
sin(theta2)*cos(theta3) - sin(theta3)*cos(theta2))*sin(theta4))*sin(theta5) + (-
0.08535*sin(theta2)*sin(theta3) + 0.08535*cos(theta2)*cos(theta3))*sin(theta4) +
(0.08535*sin(theta2)*cos(theta3) + 0.08535*sin(theta3)*cos(theta2))*cos(theta4) +
0.2132*sin(theta2)*sin(theta3) - 0.2132*cos(theta2)*cos(theta3) - 0.24355*cos(theta2)
(-0.0921*(-sin(theta2)*sin(theta3) + cos(theta2)*cos(theta3))*cos(theta4) - 0.0921*(-
sin(theta2)*cos(theta3) - sin(theta3)*cos(theta2))*sin(theta4))*sin(theta5) + (-
0.08535*sin(theta2)*sin(theta3) + 0.08535*cos(theta2)*cos(theta3))*sin(theta4) +
(0.08535*sin(theta2)*cos(theta3) + 0.08535*sin(theta3)*cos(theta2))*cos(theta4) +
0.2132*sin(theta2)*sin(theta3) - 0.2132*cos(theta2)*cos(theta3)
(-0.0921*(-sin(theta2)*sin(theta3) + cos(theta2)*cos(theta3))*cos(theta4) +
0.0921*(sin(theta2)*cos(theta3) + sin(theta3)*cos(theta2))*sin(theta4))*sin(theta5) -
(0.08535*sin(theta2)*sin(theta3) - 0.08535*cos(theta2)*cos(theta3))*sin(theta4) +
(0.08535*sin(theta2)*cos(theta3) + 0.08535*sin(theta3)*cos(theta2))*cos(theta4)
(-0.0921*(-sin(theta2)*sin(theta3) + cos(theta2)*cos(theta3))*sin(theta4) -
0.0921*(sin(theta2)*cos(theta3) +
sin(theta3)*cos(theta2))*cos(theta4))*cos(theta5)
0]

[
]

[
sin(theta1)
sin(theta1)
sin(theta1)
(-sin(theta2)*sin(theta3)*cos(theta1) + cos(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - (-
sin(theta2)*cos(theta1)*cos(theta3) - sin(theta3)*cos(theta1)*cos(theta2))*cos(theta4)
-((-sin(theta2)*sin(theta3)*cos(theta1) + cos(theta1)*cos(theta2)*cos(theta3))*cos(theta4)
+ (-sin(theta2)*cos(theta1)*cos(theta3) -
sin(theta3)*cos(theta1)*cos(theta2))*sin(theta4))*sin(theta5) + sin(theta1)*cos(theta5)
-((-sin(theta2)*sin(theta3)*cos(theta1) + cos(theta1)*cos(theta2)*cos(theta3))*cos(theta4)
+ (-sin(theta2)*cos(theta1)*cos(theta3) -
sin(theta3)*cos(theta1)*cos(theta2))*sin(theta4))*sin(theta5) + sin(theta1)*cos(theta5)]

[
]

[
-
cos(theta1)

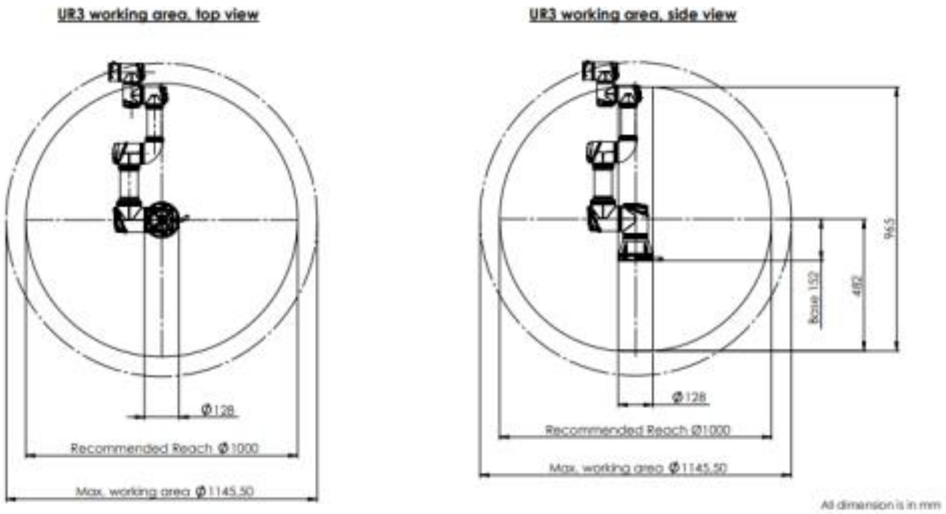
-
cos(theta1)
-
cos(theta1)
(-sin(theta1)*sin(theta2)*sin(theta3) + sin(theta1)*cos(theta2)*cos(theta3))*sin(theta4) - (-sin(theta1)*sin(theta2)*cos(theta3) - sin(theta1)*sin(theta3)*cos(theta2))*cos(theta4)
-((-sin(theta1)*sin(theta2)*sin(theta3) + sin(theta1)*cos(theta2)*cos(theta3))*cos(theta4) + (-sin(theta1)*sin(theta2)*cos(theta3) - sin(theta1)*sin(theta3)*cos(theta2))*sin(theta4))*sin(theta5) - cos(theta1)*cos(theta5)
-((-sin(theta1)*sin(theta2)*sin(theta3) + sin(theta1)*cos(theta2)*cos(theta3))*cos(theta4) + (-sin(theta1)*sin(theta2)*cos(theta3) - sin(theta1)*sin(theta3)*cos(theta2))*sin(theta4))*sin(theta5) - cos(theta1)*cos(theta5)]

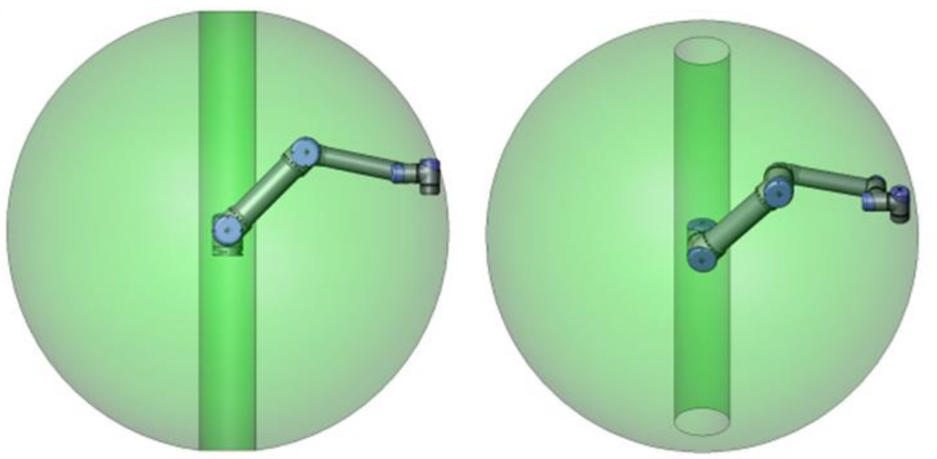
[
]

[
0
0
0
-(-sin(theta2)*sin(theta3) + cos(theta2)*cos(theta3))*cos(theta4) + (sin(theta2)*cos(theta3) + sin(theta3)*cos(theta2))*sin(theta4)
-((-sin(theta2)*sin(theta3) + cos(theta2)*cos(theta3))*sin(theta4) + (sin(theta2)*cos(theta3) + sin(theta3)*cos(theta2))*cos(theta4))*sin(theta5)
-((-sin(theta2)*sin(theta3) + cos(theta2)*cos(theta3))*sin(theta4) + (sin(theta2)*cos(theta3) + sin(theta3)*cos(theta2))*cos(theta4))*sin(theta5)]

WORKSPACE DESCRIPTION:

The workspace of UR3 robot is spherical and the inner circle represents recommended reach, and the outer circle represents max working area. Any position in the circle can be achieved but with limitations to robot orientation. The robot cannot reach the position outside the sphere. The robot cannot reach the positions in the column directly above and below of the robot.





Front Tilted

The attachment of the mobile platform allows the robot manipulator to reach any point in the X and Y plane. The front castor wheels can be steered to achieve this, and the back wheels are used for forward and backward movement.

MODEL ASSUMPTIONS:

UR3 manipulator will have a 6-axis robot arm with a camera at its end-effector. Usually, the high-speed camera will be placed on a rail with only horizontal movement to move the robot a specific distance. We added wheels to the robot base so that the robot can move any distance in the room in both horizontal and vertical directions.

Design Assumptions -

1. Considering the surfaces & wheels are smooth (frictionless).
2. Objects and links are considered to be rigid and inextensible.
3. Negligible air and wind resistance.

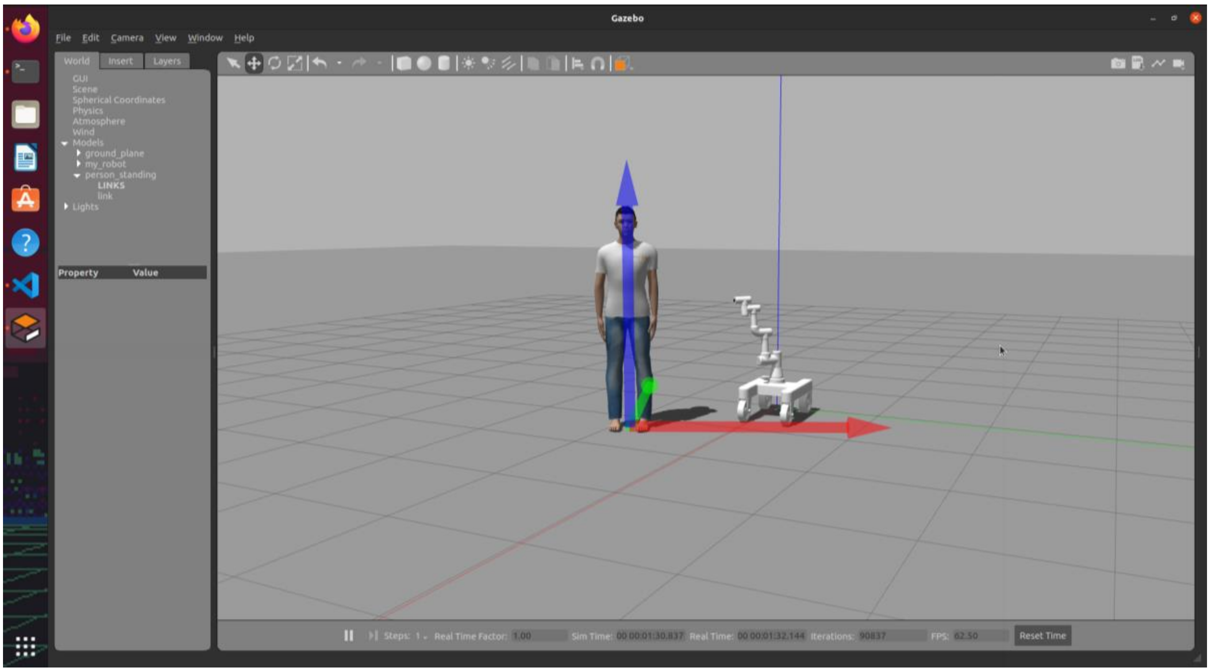
CONTROL METHOD :

The UR3 robot has 6 revolute joints, and the mobile platform has two front wheels which can steer, and another two rear wheels that can be used for forward and backward movement. To get precise movement, we used Position controllers for all six joints of UR3 robot. For the rear wheels of the platform, velocity controllers were used and for the front wheels, effort controllers were used which will allow them to steer the platform. All the controllers internally use PID gains for error free movement and a closed loop is not needed. We included PID gains for joints and wheels to reduce the error in the movement of joints and wheels and follow the specific trajectory with precision. The position controller accepts the position values in radians or meters as input.

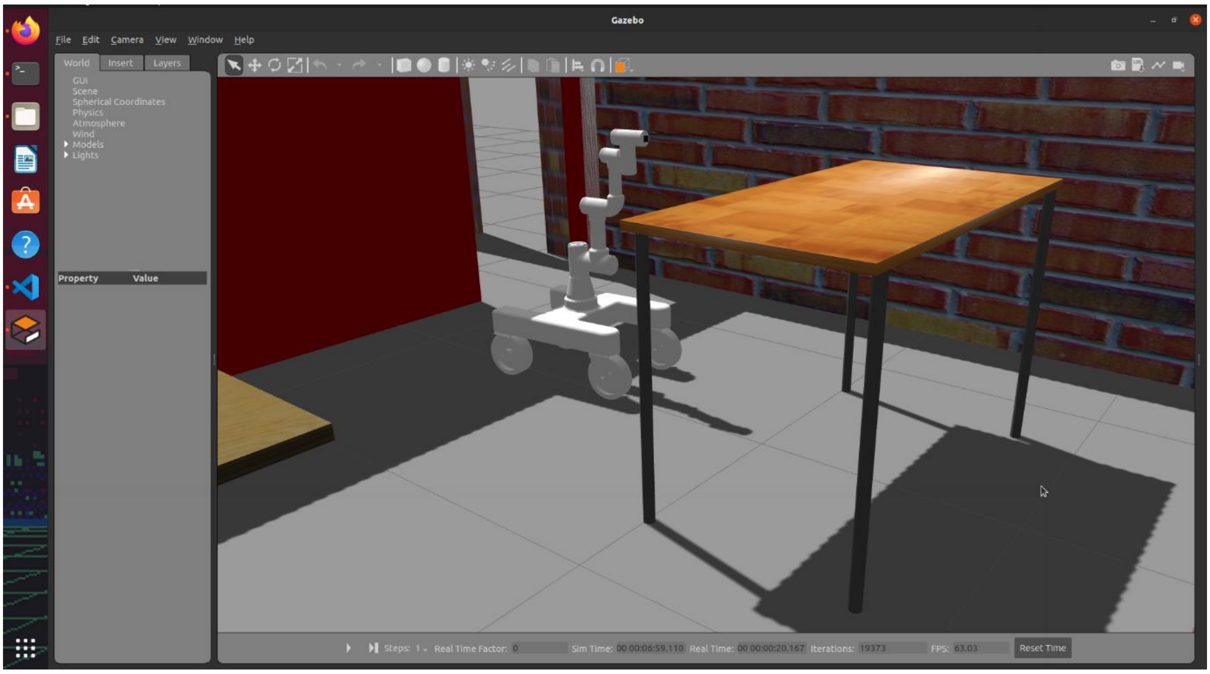
GAZEBO AND RVIZ VISUALIZATION:

The robot arm is moved by directly publishing the messages to the controllers using a python script.

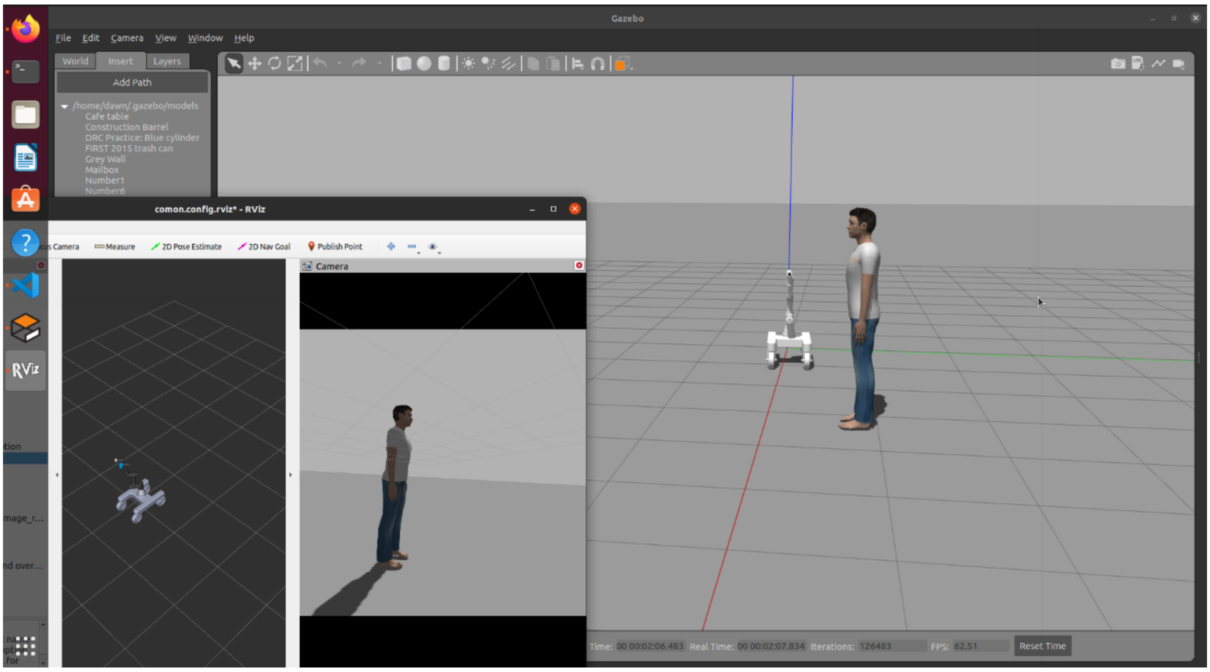
Simulation video link - https://drive.google.com/file/d/18MGY7aEzOq0-LxsUfm40aQw0cfJXwQ_W/view?usp=drivesdk



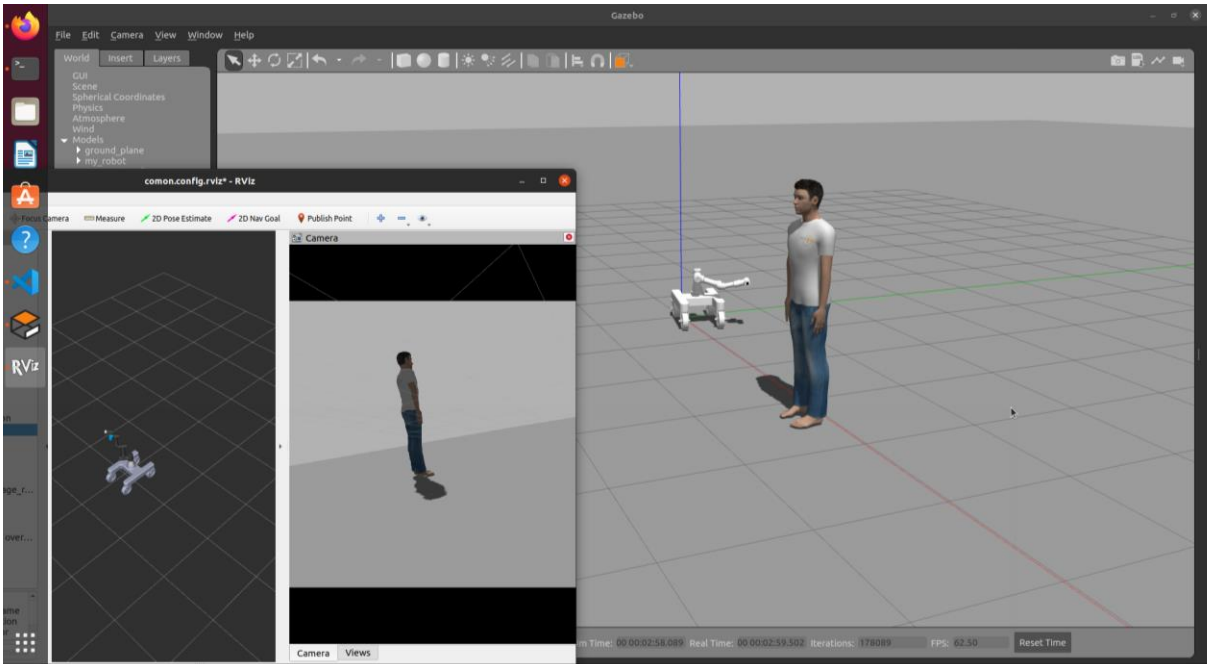
Mobile Manipulator next to a person



Mobile Manipulator next to a table



RViz Visualization 1



RViz Visualization 2

PROBLEMS FACED:

Design -

1. The first challenge is choosing an optimal design for the platform of the mobile manipulator so that the robot arm stays in the center of the platform and the platform does not topple due to the arm's weight. We designed a platform in the shape of H and placed the robot arm in the center of the platform and increased the mass of the platform to carry the arm without toppling.
2. The other major challenge faced is exporting the mobile manipulator to URDF. The CAD model had the wrong joint axes definition and origin. We changed the origin and joint axes to rotate the joints correctly.

ROS & Visualization -

1. The challenge we faced during the gazebo simulation is the toppling of the robot arm due to the misalignment of the center of mass of the robot arm. We adjusted the mass of the links in the URDF file to avoid the robot arm toppling in the gazebo.
2. The next challenge we faced is choosing the appropriate controllers for the smooth movement of arm joints. Firstly, we chose effort controllers which made the joints behave disorderly. Then, we chose position controllers which resolved the issue.
3. The major challenge is finding the correct position and orientation of the camera for the perfect view at the end effector. After multiple iterations and gazebo launches, we found the ideal position and orientation of the camera.

LESSONS LEARNED:

1. Designing a robot platform to avoid robot toppling and defining the joint axes correctly in URDF for a robot manipulator.
2. Using peter Corke toolbox for visualizing the DH table of a robot manipulator.
3. Using Position controllers for the revolute joints for precise movement.
4. Integrating the camera to the end effector and visualizing in RViz.
5. Calculating inverse position kinematics using rotation matrices and geometrical approach.

CONCLUSION:

In this project, a mobile platform was designed and attached to the robot manipulator with the right amount of weight to carry the manipulator. By adding the platform, we can now transport the robotic arm with ease. The forward kinematics and inverse kinematics were derived for the UR3 manipulator and validated. Position controllers help to move the robot joints with angular precision. Gazebo simulation and RViz visualization are done using by moving the robot by executing a python ROS script.

In conclusion, mobile robotic cameras can reduce the workload of humans by easily programming the movements and the speed to achieve desired complex camera movements. A High-speed camera with a mobile manipulator is the need of the hour for many cinematographers and directors.

FUTURE WORK:

1) Design a robot arm with more reachability and height. For example, the arm should reach two meters and three meters in height, so that it can be used for real-world applications.
2) Execute and simulate the trajectory of the robot manipulator using inverse velocity kinematics.
3) Use MoveIt for path planning and avoiding collisions with the mobile platform.
4) Increase the mass of the manipulator to handle heavy cameras at the end effector and increase the stability of the manipulator.
5) Calculate the optimal path to reach a desired position in a quick time.

References :

1) UR3 CAD Model - https://www.traceparts.com/en/product/universal-robots-as-ur3-robot?Product=10-06032017-106400
2) UR3 Specifications - https://www.usna.edu/Users/weaprcon/kutzer/_files/documents/User%20Manual,%20UR3.pdf
3) Inverse kinematics - https://airccse.com/jares/papers/1419jares01.pdf
4) UR3 Workspace - https://devicebase.net/en/universal-robots-ur3/questions/what-are-the-dimensions-of-a-robot-and-its-parts/3ay
5) DH parameters explanation - https://www.researchgate.net/profile/Mohamed_Mourad_Lafifi/post/How_to_avoid_singular_configurations/attachment/59d6361b79197b807799389a/AS%3A386996594855942%401469278586939/download/Spong+-+Robot+modeling+and+Control.pdf

APPENDIX

Authentic code for simulation and validation have been given below:

Appendix A –

1. "forward_kinematics.py"

```
# Program to find the final transformation matrix (Forward Kinematics), position and orientation of the end effector
# of UR3 mobile manipulator using DH Parameters

from sympy import *

init_printing(use_unicode=False, wrap_line=False)

""" Defining symbols for DH parameters """
al, th, d, a = symbols('al th d a')

""" Defining joint angles as symbols """
th1, th2, th3, th4, th5, th6 = symbols('\u03B8\u2081 \u03B8\u2082 \u03B8\u2083 \u03B8\u2084 '
                            '\u03B8\u2085 \u03B8\u2086')

""" Defining link offsets and link lengths as symbols """
d1, d4, d5, d6, a2, a3 = symbols('d1 d4 d5 d6 a2 a3')

""" Setting up Transformation matrices for multiplication """
Rz = Matrix([[cos(th), -sin(th), 0, 0],
        [sin(th),  cos(th), 0, 0],
        [     0,        0, 1, 0],
        [     0,        0, 0, 1]])

Tz = Matrix([[  1,  0,  0,  0],
        [ 0,  1,  0,  0],
        [ 0,  0,  1,  d],
        [ 0,  0,  0,  1]])

Tx = Matrix([[  1,  0,  0,  a],
        [ 0,  1,  0,  0],
        [ 0,  0,  1,  0],
        [ 0,  0,  0,  1]])

Rx = Matrix([[  1,        0,        0,  0],
        [ 0,  cos(al), -sin(al),  0],
        [ 0,  sin(al),  cos(al),  0],
        [ 0,        0,        0,  1]])
```

```
""" Defining symbolic expressions of transformations for each row of DH-parameter """
result1 = Rz*Tz*Tx*Rx
result2 = Rz*Tz*Tx*Rx
result3 = Rz*Tz*Tx*Rx
result4 = Rz*Tz*Tx*Rx
result5 = Rz*Tz*Tx*Rx
result6 = Rz*Tz*Tx*Rx

print("A :")
pprint(result1)
print('\n')

""" Defining DH parameters """
alpha = [pi/2, 0, 0, pi/2, -pi/2, 0]
theta = [th1, th2, th3, th4, th5, th6]
di = [ d1, 0, 0, d4, d5, d6]
ai = [ 0, a2, a3, 0, 0, 0]

""" Substituting each row of DH parameters in their respective expression """
T1 = result1.subs(al, alpha[0]).subs(th, theta[0]).subs(d, di[0]).subs(a, ai[0])
T2 = result2.subs(al, alpha[1]).subs(th, theta[1]).subs(d, di[1]).subs(a, ai[1])
T3 = result3.subs(al, alpha[2]).subs(th, theta[2]).subs(d, di[2]).subs(a, ai[2])
T4 = result4.subs(al, alpha[3]).subs(th, theta[3]).subs(d, di[3]).subs(a, ai[3])
T5 = result5.subs(al, alpha[4]).subs(th, theta[4]).subs(d, di[4]).subs(a, ai[4])
T6 = result6.subs(al, alpha[5]).subs(th, theta[5]).subs(d, di[5]).subs(a, ai[5])

""" Setting up homogeneous transformation matrices """
H1 = T1
H2 = H1 * T2
H3 = H2 * T3
H4 = H3 * T4
H5 = H4 * T5
H6 = H5 * T6

print("Final T matrix:")
pprint(H6)
print('\n')

""" Defining joint angles with offset """
q1 = 0
q2 = 0
q3 = 0
q4 = 0
q5 = 0
q6 = 0
```

```
""" Substituting values """
H6_1 = H6.subs({th1: q1, th2: q2, th3: q3, th4: q4, th5: q5, th6: q6, d1: 0.15185, d4: 0.13105,
d5: 0.08535, d6: 0.0921,
            a2: -0.244, a3: -0.213})

""" Extracting end effector position and orientation """
Xp = Matrix([[H6_1[3]], [H6_1[7]], [H6_1[11]]])
R = atan2(H6_1[8], H6_1[9])
P = atan2(-H6_1[8], sqrt(H6_1[9]**2 + H6_1[10]))
Y = atan2(H6_1[0], H6_1[4])

print("End effector position:")
pprint(Xp)
print('\n')
print("Roll(R):", R)
print("Pitch(P):", P)
print("Yaw(Y):", Y)
```

2. Forward kinematics validation – Matlab script

```
clear all
clf

d1=0.15185; d4= 0.13105; d5= 0.08535; d6=0.0921;
a2=-0.24355; a3 = -0.2132;

L(1)=Link([0, d1,  0,  pi/2, 0]);
L(2)=Link([0,  0,  a2, 0, 0]);
L(3)=Link([0, 0, a3, 0, 0]);
L(4)=Link([0,  d4, 0,  pi/2, 0]);
L(5)=Link([0, d5,   0,  -pi/2, 0]);
L(6)=Link([0,  d6, 0, 0, 0]);

robot = SerialLink (L);
robot.name = 'UR 3';
robot.teach
```

3. Inverse kinematics.py

```
from sympy import *
from sympy.matrices import Matrix
from sympy import pprint
import numpy as np
import matplotlib.pyplot as plt

#Function to create a Transformation matrix using DH parameters - α, θ, d, a
def Transformation(theta, d, a, alpha):
    d = float(d)
    a = float(a)
```

```
    T = Matrix([[cos(theta), -sin(theta)*cos(alpha), sin(theta)*sin(alpha),
a*cos(theta)],
    [sin(theta), cos(theta)*cos(alpha), -cos(theta)*sin(alpha),
a*sin(theta)],
    [0, sin(alpha), cos(alpha), d],
    [0, 0, 0, 1]])
    return T


theta = np.linspace(0, 360, num=50)
dt = 5/50   #dt = T/N where T is 5 secs and No.of intervals is 50
x = []
y = []
z = []

d1=0.15185
d4=0.13105
d5= 0.08535
d6=0.0921
a2=-0.24355
a3 = -0.2132

q1_1 = Symbol('theta1')
q2_1 = Symbol('theta2')
q3_1 = Symbol('theta3')
q4_1 = Symbol('theta4')
q5_1 = Symbol('theta5')
q6_1 = Symbol('theta6')

T1 =Transformation(q1_1, d1,  0,  pi/2)
T2 =Transformation(q2_1,  0,  a2, 0)
T3 =Transformation(q3_1, 0, a3, 0)
T4 =Transformation(q4_1,  d4, 0,  pi/2)
T5 =Transformation(q5_1, d5,   0,  -pi/2)
T6 =Transformation(q6_1,  d6, 0, 0)

H1 = T1
H2 = H1*T2
H3 = H2*T3
H4 = H3*T4
H5 = H4*T5
H6 = H5*T6

Z0 = Matrix([[0], [0], [1]])                # Z component of base frame
Z1 = Matrix([[H1[2]], [H1[6]], [H1[10]]])   #Taking Z component of H1
Z2 = Matrix([[H2[2]], [H2[6]], [H2[10]]])   #Taking Z component of H2
Z3 = Matrix([[H3[2]], [H3[6]], [H3[10]]])   #Taking Z component of H3
Z4 = Matrix([[H4[2]], [H4[6]], [H4[10]]])   #Taking Z component of H4
Z5 = Matrix([[H5[2]], [H5[6]], [H5[10]]])   #Taking Z component of H5
Z6 = Matrix([[H6[2]], [H6[6]], [H6[10]]])   #Taking Z component of H6

Xp = Matrix([[H6[3]], [H6[7]], [H6[11]]])   #Translation component of final
transformation matrix - H6

C1 = diff(Xp, q1_1) #Differentitaing Xp w.r.t theta 1
C2 = diff(Xp, q2_1) #Differentitaing Xp w.r.t theta 2
C3 = diff(Xp, q3_1) #Differentitaing Xp w.r.t theta 4
C4 = diff(Xp, q4_1) #Differentitaing Xp w.r.t theta 5
C5 = diff(Xp, q5_1) #Differentitaing Xp w.r.t theta 6
C6 = diff(Xp, q6_1) #Differentitaing Xp w.r.t theta 7

J1 = Matrix([[C1], [Z1]]) #Computing Jacobian 1st column
J2 = Matrix([[C2], [Z2]]) #Computing Jacobian 2nd column
```

```
J3 = Matrix([[C3], [Z3]]) #Computing Jacobian 3rd column
J4 = Matrix([[C4], [Z4]]) #Computing Jacobian 4th column
J5 = Matrix([[C5], [Z5]]) #Computing Jacobian 5th column
J6 = Matrix([[C6], [Z6]]) #Computing Jacobian 6th column

J = Matrix([[J1, J2, J3, J4, J5, J6]])
pprint(J)

#Initial Joint angles
q1 = 0     #Theta 1
q2 = 0     #Theta 2
q3 = 0     #Theta 3
q4 = 0     #Theta 4
q5 = 0     #Theta 5
q6 = 0     #Theta 6

fig = plt.figure(figsize=(4,4))
ax = fig.add_subplot(111, projection='3d')

for t in theta:
    Vx = 0                             # X-component of velocity trajectory
    Vy = -0.025*2*(pi/5)*sin(t*(pi/180))  # Y-component of velocity
trajectory
    Vz = 0.025*2*(pi/5)*cos(t*(pi/180))   # Z-component of velocity
trajectory
    Wx = 0                             # Angular velocities are zero
    Wy = 0
    Wz = 0
    X_dot = Matrix([[Vx], [Vy], [Vz], [Wx], [Wy], [Wz]]) # Cartesian velocity
matrix
    J_1 = J.evalf(subs ={q1_1 : q1, q2_1 : q2, q3_1 : q3, q4_1 : q4, q5_1 :
q5, q6_1 : q6}) # Substituting theta values in Jacobian Matrix
    H6_1 = H6.subs({q1_1 : q1, q2_1 : q2, q3_1 : q3, q4_1 : q4, q5_1 : q5,
q6_1 : q6})      # Substituting theta values in Final Trasnformation Matrix
H6
    x.append(H6_1[3])   # X-component of end effector position
    y.append(H6_1[7])   # Y-component of end effector position
    z.append(H6_1[11])  # Z-component of end effector position
    ax.scatter(H6_1[3], H6_1[7],H6_1[11])
    plt.xlim(-2, 0)
    plt.ylim(-0.35, -0.2)
    ax.set_xlabel('X-axis', fontweight ='bold')
    ax.set_ylabel('Y-axis', fontweight ='bold')
    ax.set_zlabel('Z-axis', fontweight ='bold')
    ax.set_title('Trajectory of end effector', fontsize = 18, fontweight
='bold')
    plt.pause(0.5)
    q_dot = J_1.pinv()*X_dot    # Obtaining the joint angular velocities
Matrix
    q1_dot = q_dot[0].evalf()
    q2_dot = q_dot[1].evalf()
    q3_dot = q_dot[2].evalf()
    q4_dot = q_dot[3].evalf()
    q5_dot = q_dot[4].evalf()
    q6_dot = q_dot[5].evalf()
    # Performing numerical integration on the joint angular velocities to
obtain the new joint angles
    q1 = q1 + (q1_dot*dt)
    q2 = q2 + (q2_dot*dt)
    q3 = q3 + (q3_dot*dt)
    q4 = q4 + (q4_dot*dt)
    q5 = q5 + (q5_dot*dt)
    q6 = q6 + (q6_dot*dt)
```

```
    q1 = q1.evalf()
    q3 = q3.evalf()
    q2 = q2.evalf()
    q4 = q4.evalf()
    q5 = q5.evalf()
    q6 = q6.evalf()


plt.show()
```