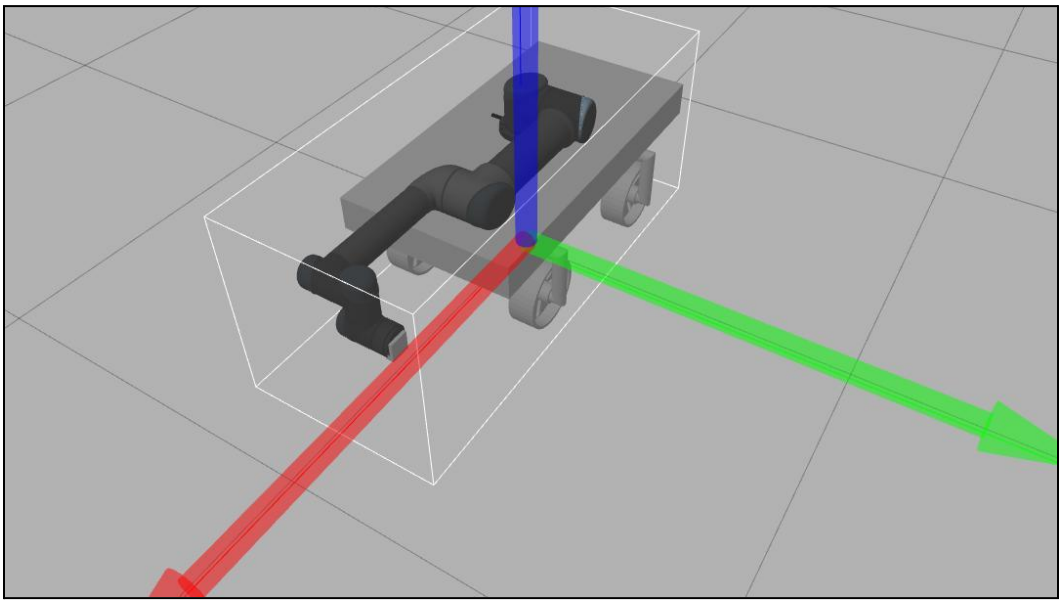
Introduction to Robot Modeling (ENPM662) Project Proposal UR5 Mobile Robot

DATED: 09/12/2022



Team Members:

- **Patrik**
- **Shivam**

Under Guidance:

Prof Reza Monfaredi

Contents

1 Introduction	**3**
2 Application	**3**
3 Robot Type	**4**
4 DOFs and Dimensions	**5**
5 CAD Models	**6**
6 D-H Parameters	**7**
7 Forward Kinematics	**7**
8 Inverse Kinematics	**9**
9 Forward Kinematics Validation	**13**
10 Inverse Kinematics Validation	**14**
11 Workspace Study	**15**
12 Assumptions	**18**
13 Control Method	**18**
14 Gazebo & RViz Visualization	**22**
15 Problems Faced	**23**
16 Lessons Learned	**23**
17 Conclusions	**24**
18 Future Work	**25**
20 References	**25**

Introduction

This document contains our project 2 report of subject Introduction to Robot Modelling (ENPM662). The document is divided into a total of 20 different sections and each section will address a particular area of the project, these sections can be grouped into three phases basically. The first part, containing sections 1-8, revolves around the preparation. We had to plan our steps and set the goals to achieve, moreover the used robots had to be selected and studied, the corresponding virtual models had to be collected and kinematic assumptions were made.

The second step, containing sections 9-14 was to put the theoretical knowledge into practice including validating our forward and inverse kinematic assumptions, planning and building the workspace and building a simulation.

The last step, containing sections 15-20, is an overview of the project, the difficulties, successes and future development proposals will be discussed, as a result an overall conclusion can be drawn.

Application

Having worked in a manufacturing sector where mechanical assembly of components is a major work that is achieved using manual labor. It involves the process of putting together components on an assembly line. Let us consider a case where a part has to be assembled in 4 stages, in every stage a certain assembly is done and the part is transferred to the next stage.

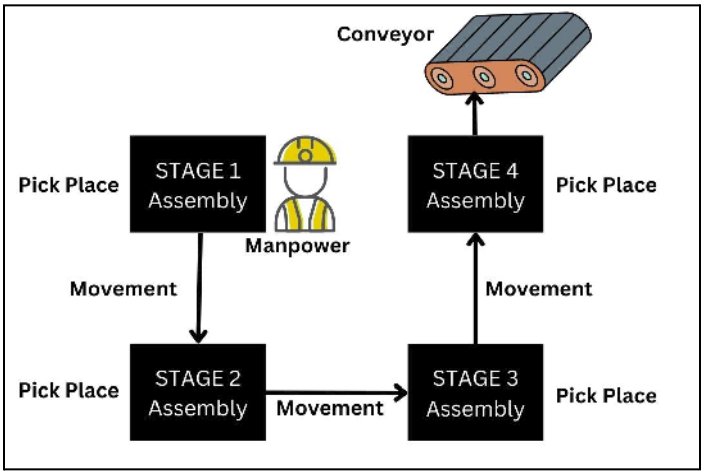


Fig1 Assembly

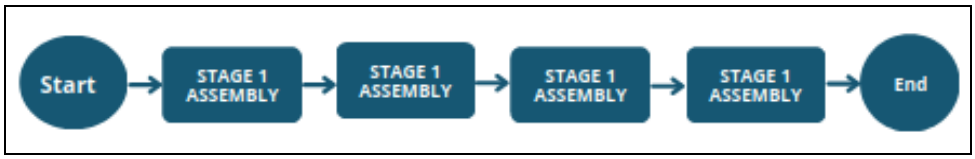


Fig2 Process Flow of Assembly

Our intent is to build a mobile robot that can complete all the tasks described above. **The world is moving forward towards an era where humans would not have to do such menial tasks. Therefore, we want to contribute towards this goal.**

Robot Type

Universal Robots is a Danish robotics company founded in 2005. It provides UR3, UR5 and UR10 series of industrial robots to meet worldwide industrial automation requirements in different levels. The end effector is a vacuum gripper that by means of eliminating the air between the gripper and the object keeps the object stable.
The automated guided vehicle in our case was the car that we created in project 1. On the other hand in real life applications that model is not appropriate enough, to work safely in a hazardous environment, many sensors and safety equipment would need to be attached. Since it was not the main scope of our project we just stuck to our simplified model.

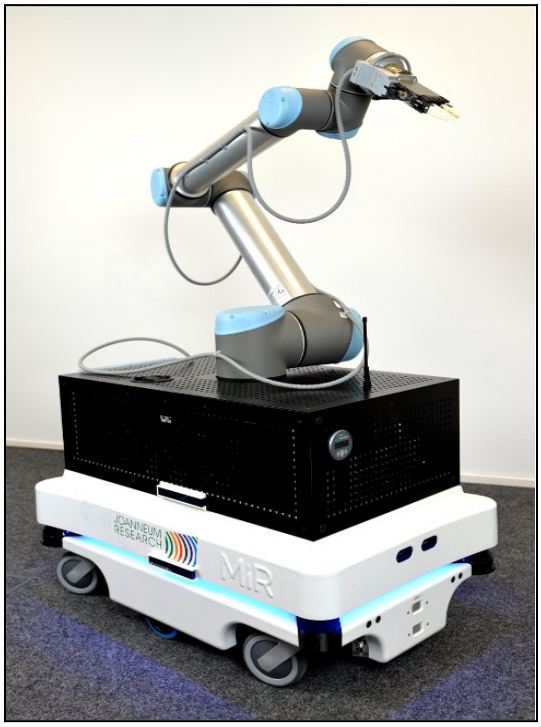


Fig3 UR5 Mobil Robot

DOFs and Dimensions

UR5 is a typical 6-axis industrial robot. The maximum payload of UR5 is 5 kg and its work radius is 850 mm. (Universal Robots A/S, 2009).
The UR5 robot has 6 revolute joints which gives it 6 Degrees of freedom but for our project we are modeling a UR5 on a mobile robot which has 2 degrees of freedom, therefore our robot will have 8 degrees of freedom.

Technical Specifications

System Parameter	UR5
Degrees of Freedom	6 rotating joints
Payload	5 kg / 11 lbs
Repeatability	±0.1 mm / ±0.0039 in (4 mils)
Weight with cable	18.4 kg / 40.6 lbs
Reach	850 mm / 33.5 in
Motion Range	Base: ± 360° Shoulder: ± 360° Elbow: ± 360° Wrist 1: ± 360° Wrist 2: ± 360° Wrist 3: ± 360°
Maximum Speed	Base: ± 180°/Sec. Shoulder: ± 180°/Sec. Elbow: ± 180°/Sec. Wrist 1: ± 180°/Sec. Wrist 2: ± 180°/Sec. Wrist 3: ± 180°/Sec.
Power Consumption	Min 90W, Typical 150W, Max 325W
Robot Mounting	Any
Ambient Temperature Range	0-50°*

Fig4 UR5 specifications

The dimensions of our mobile robot were pretty much the same as it was described in project one, on the other hand we increased the mass a bit to achieve stability when the robot arm moves.

CAD Models

There are different resources to gather the UR5 model, it can be downloaded from Grab CAD, our more official formats can be found on the Universal Robotics web page or already ROS implemented versions onGitHub:
https://www.universal-robots.com/download/mechanical-cb-series/ur5/robot-ur5-step-file-cb-series/
https://github.com/ros-industrial/universal_robot

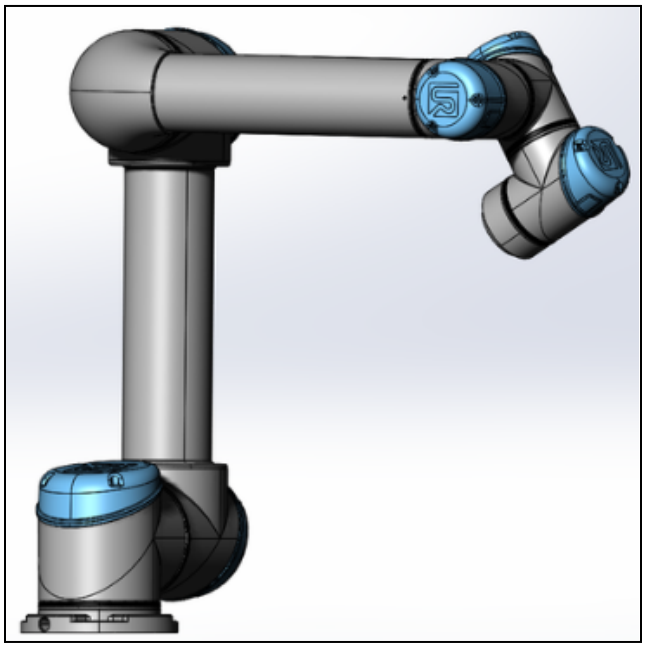


Fig4 UR5 CAD model

The mobile robot is the same that was used in project 1:

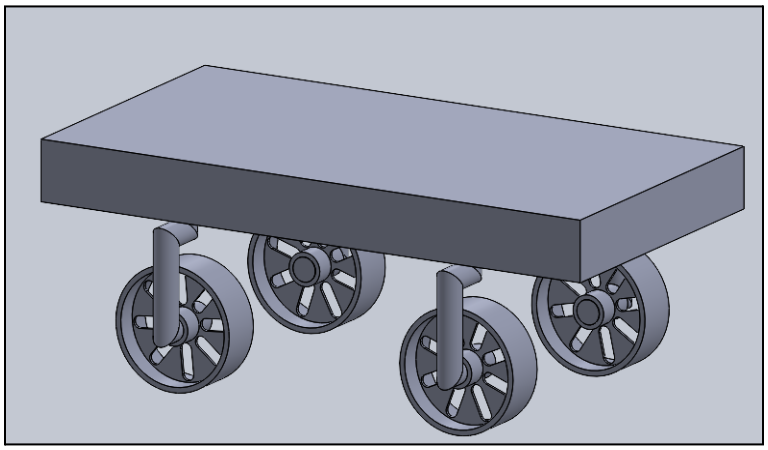
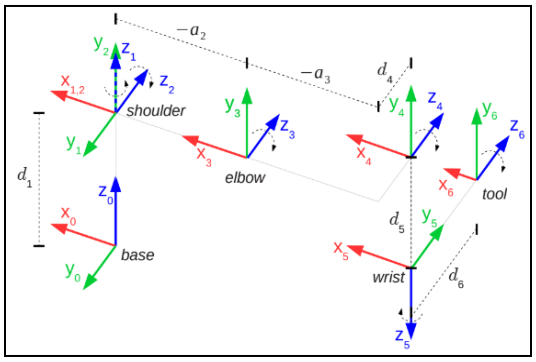
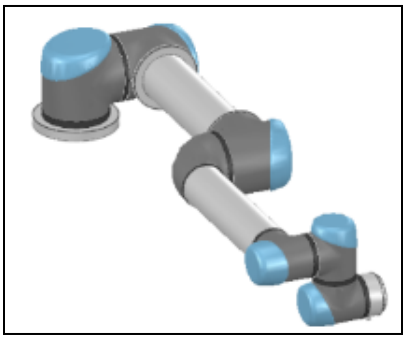


Fig5 Mobil Robot CAD model

D-H Parameters

Since the UR5 is a well researched robot arm there are many good resources about its' kinematics. Since the company is European we stuck to the DH conventions by Craig.





	α_{i-1}	a_i	d_i	ϑ_i
1	0	0	d_1	ϑ_1
2	$\alpha_1 = \frac{\pi}{2}$	0	0	ϑ_2
3	0	a_2	0	ϑ_3
4	0	a_3	d_4	ϑ_4
5	$\alpha_4 = \frac{\pi}{2}$	0	d_5	ϑ_5
6	$\alpha_5 = -\frac{\pi}{2}$	0	d_6	ϑ_6

a_i = distance from Z_i to Z_{i+1} measured along X_i
α_i = angle from Z_i to Z_{i+1} measured about X_i
d_i = distance from X_{i-1} to X_i measured along Z_i
θ_i = angle from X_{i-1} to X_i measured about Z_i

Fig6 UR5 frames in home position and the corresponding DH-table

Forward Kinematics

Using the DH-table and the matrix expression below the individual transformation matrices were calculated.

$$ {}^{i-1}_{i}T = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i \cos(\alpha_{i-1}) & \cos\theta_i \cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -\sin(\alpha_{i-1}) d_i \\ \sin\theta_i \sin(\alpha_{i-1}) & \cos\theta_i \sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & \cos(\alpha_{i-1}) d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Using their multiplication the transformation matrix between the base frame and the tool frame was calculated.

$$^{0}_{6}T(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) = {}^{0}_{1}T(\theta_1)\,{}^{1}_{2}T(\theta_2)\,{}^{2}_{3}T(\theta_3)\,{}^{3}_{4}T(\theta_4)\,{}^{4}_{5}T(\theta_5)\,{}^{5}_{6}T(\theta_6)$$

$$^{0}_{6}T(\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6) = \begin{bmatrix} {}^{0}_{6}R & {}^{0}P_6 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} {}^{0}\hat{X}_{6x} & {}^{0}\hat{Y}_{6x} & {}^{0}\hat{Z}_{6x} & {}^{0}P_{6x} \\ {}^{0}\hat{X}_{6y} & {}^{0}\hat{Y}_{6y} & {}^{0}\hat{Z}_{6y} & {}^{0}P_{6y} \\ {}^{0}\hat{X}_{6z} & {}^{0}\hat{Y}_{6z} & {}^{0}\hat{Z}_{6z} & {}^{0}P_{6z} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The final transformation matrix is huge, its' individual columns can be se bellow:

$$\begin{bmatrix} (\sin(\theta_1)\sin(\theta_5)+\cos(\theta_1)\cos(\theta_5)\cos(\theta_2+\theta_3+\theta_4))\cos(\theta_6)-\sin(\theta_6)\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_1) \\ (\sin(\theta_1)\cos(\theta_5)\cos(\theta_2+\theta_3+\theta_4)-\sin(\theta_5)\cos(\theta_1))\cos(\theta_6)-\sin(\theta_1)\sin(\theta_6)\sin(\theta_2+\theta_3+\theta_4) \\ \sin(\theta_6)\cos(\theta_2+\theta_3+\theta_4)+\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_5)\cos(\theta_6) \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} -(\sin(\theta_1)\sin(\theta_5)+\cos(\theta_1)\cos(\theta_5)\cos(\theta_2+\theta_3+\theta_4))\sin(\theta_6)-\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_1)\cos(\theta_6) \\ (-\sin(\theta_1)\cos(\theta_5)\cos(\theta_2+\theta_3+\theta_4)+\sin(\theta_5)\cos(\theta_1))\sin(\theta_6)-\sin(\theta_1)\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_6) \\ -\sin(\theta_6)\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_5)+\cos(\theta_6)\cos(\theta_2+\theta_3+\theta_4) \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \sin(\theta_1)\cos(\theta_5)-\sin(\theta_5)\cos(\theta_1)\cos(\theta_2+\theta_3+\theta_4) \\ -\sin(\theta_1)\sin(\theta_5)\cos(\theta_2+\theta_3+\theta_4)-\cos(\theta_1)\cos(\theta_5) \\ -\sin(\theta_5)\sin(\theta_2+\theta_3+\theta_4) \\ 0 \end{bmatrix}$$

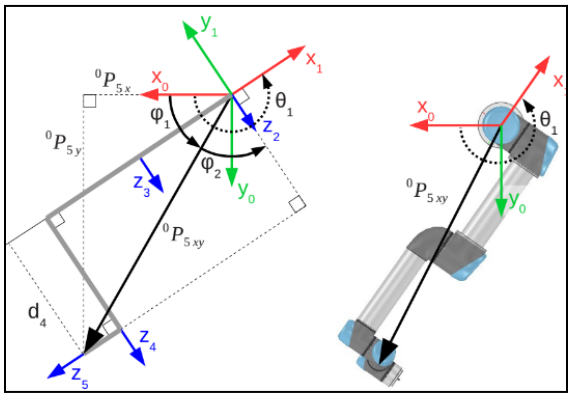
$$\begin{bmatrix} 0.0823\sin(\theta_1)\cos(\theta_5)+0.10915\sin(\theta_1)-0.0823\sin(\theta_5)\cos(\theta_1)\cos(\theta_2+\theta_3+\theta_4)+0.09465\sin(\theta_2+\theta_3+\theta_4)\cos(\theta_1)+0.425\cos(\theta_1)\cos(\theta_2)+0.39225\cos(\theta_1)\cos(\theta_2+\theta_3) \\ -0.0823\sin(\theta_1)\sin(\theta_5)\cos(\theta_2+\theta_3+\theta_4)+0.09465\sin(\theta_1)\sin(\theta_2+\theta_3+\theta_4)+0.425\sin(\theta_1)\cos(\theta_2)+0.39225\sin(\theta_1)\cos(\theta_2+\theta_3)-0.0823\cos(\theta_1)\cos(\theta_5)-0.10915\cos(\theta_1) \\ 0.425\sin(\theta_2)-0.0823\sin(\theta_5)\sin(\theta_2+\theta_3+\theta_4)+0.39225\sin(\theta_2+\theta_3)-0.09465\cos(\theta_2+\theta_3+\theta_4)+0.089159 \\ 1 \end{bmatrix}$$

When all of the joint variables are set to zero the following matrix is referred as the home position:

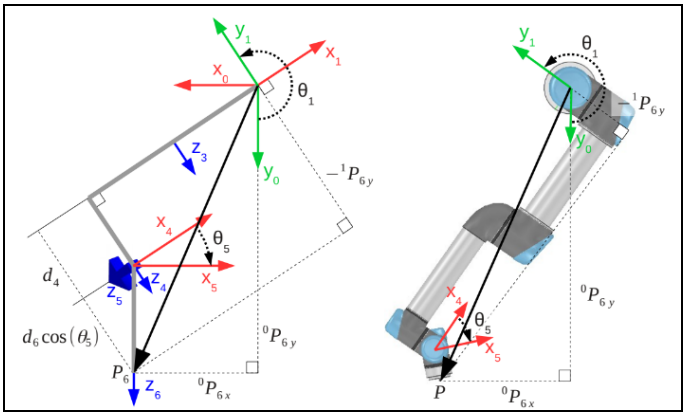
$$\begin{bmatrix} 1.0 & 0 & 0 & 0.81725 \\ 0 & 0 & -1.0 & -0.19145 \\ 0 & 1.0 & 0 & -0.005491 \\ 0 & 0 & 0 & 1.0 \end{bmatrix}$$

Inverse Kinematics

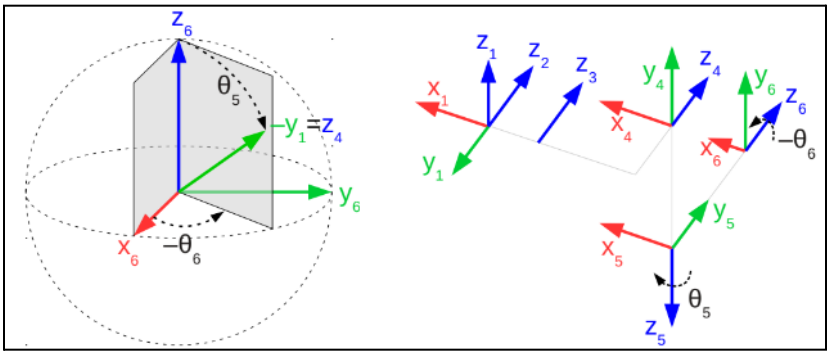
In most of the cases when we are talking about position kinematics the simplest inverse kinematics methode is the geometric, where the joint angles are determined by geometrical constraints. Since we were interested in certain positions rather than a complex movement with controlled velocity we stuck to the geometric method.
Drawing up the pose of the joints the joint angles can be written up as trigonometric functions.
Below you can see the used geometric connections, for further details please refer to the [1].



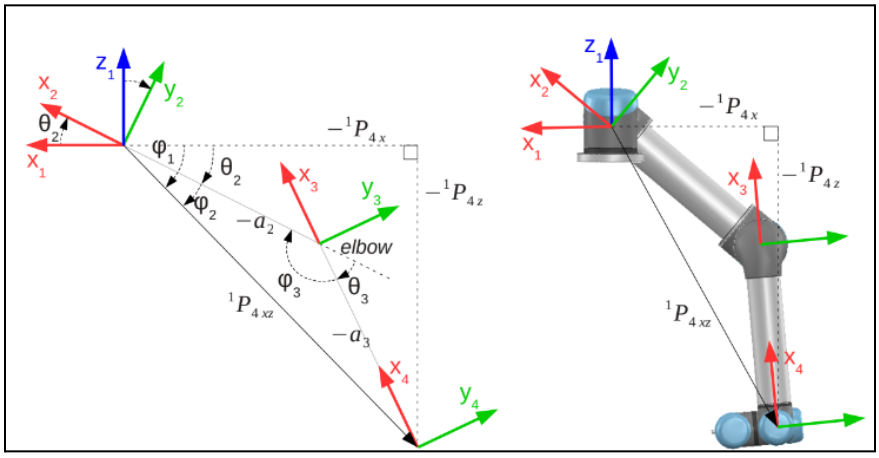
$$\theta_1 = \operatorname{atan2}\left({}^0P_{5y}, {}^0P_{5x}\right) \pm \operatorname{acos}\left(\frac{d_4}{\sqrt{{}^0P_{5x}{}^2 + {}^0P_{5y}{}^2}}\right) + \frac{\pi}{2}$$



$$\theta_5 = \pm \text{acos}\left(\frac{{}^0P_{6x}\sin\theta_1 - {}^0P_{6y}\cos\theta_1 - d_4}{d_6}\right)$$



$$\theta_6 = \text{atan2}\left(\frac{-{}^6\hat{X}_{0y}\cdot\sin\theta_1 + {}^6\hat{Y}_{0y}\cdot\cos\theta_1}{\sin\theta_5}, \frac{{}^6\hat{X}_{0x}\cdot\sin\theta_1 - {}^6\hat{Y}_{0x}\cdot\cos\theta_1}{\sin\theta_5}\right)$$



$$\theta_3 = \pm \text{acos}\left(\frac{|{}^1P_{4xz}|^2 - {a_2}^2 - {a_3}^2}{2a_2a_3}\right)$$

$$\theta_2 = \phi_1 - \phi_2 = \text{atan2}(-{}^1P_{4z}, -{}^1P_{4x}) - \text{asin}\left(\frac{-a_3\sin\theta_3}{|{}^1P_{4xz}|}\right)$$

$$\theta_4 = \text{atan2}({}^3\hat{X}_{4y}, {}^3\hat{X}_{4x})$$

Fig7 UR5 Inverse Kinematics geometric method

There are certain assumptions taken at each step that might not be that appropriate in each situation, so this model can have troubles in more complex movements, on the other hand we felt that it is good enough for our current application, the Jacobian using velocity kinematics is presented below.

CODE

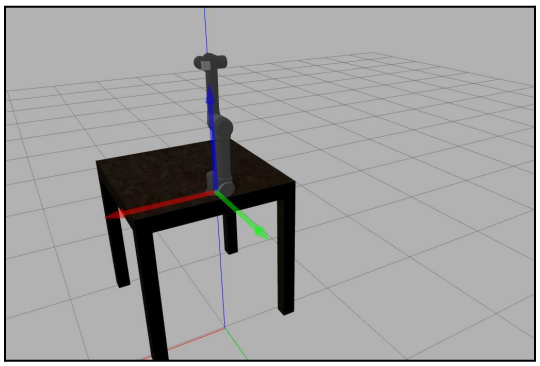
```
#sympy library used to express matrices in terms of symbols
# matplot lib used to plot the graph
import numpy as np
from sympy import *
from IPython.display import Image,display,HTML
import matplotlib.pyplot as plt
from matplotlib import cm
import numpy as np
%matplotlib qt
import math
from mpl_toolkits.mplot3d.axes3d import get_test_data
init_printing()

thetadot,t=symbols('thetadot,t')
theta,d,a,alpha=symbols('theta,d,a,alpha')
d_1,d_3,d_5,d_7,a_3=symbols('d_1,d_3,d_5,d_7,a_3')
theta_1,theta_2,theta_3,theta_4,theta_5,theta_6,theta_7=symbols('theta_1,theta_2,theta_3,theta_4,theta_5,theta_6,thet
Rot_z_theta=Matrix([[cos(theta),-sin(theta),0,0],
                    [sin(theta),cos(theta),0,0],
                    [0,0,1,0],
                    [0,0,0,1]])
Trans_z_d=Matrix([[1,0,0,0],
                  [0,1,0,0],
                  [0,0,1,d],
                  [0,0,0,1]])
Trans_x_a=Matrix([[1,0,0,a],
                  [0,1,0,0],
                  [0,0,1,0],
                  [0,0,0,1]])
Rot_x_alpha=Matrix([[1,0,0,0],
                    [0,cos(alpha),-sin(alpha),0],
                    [0,sin(alpha),cos(alpha),0],
                    [0,0,0,1]])
# Transfromation matrix for each frame
A=Rot_z_theta*Trans_z_d*Trans_x_a*Rot_x_alpha
```

```
# Substituting values form DH table
A_1=A.subs({theta:theta_1,d:0.089159,a:0,alpha:pi/2})
A_2=A.subs({theta:theta_2,d:0,a:-0.425,alpha:0})
A_3=A.subs({theta:theta_3,d:0,a:-0.39225 ,alpha:0})
A_4=A.subs({theta:theta_4,d:0.10915,a:0,alpha:pi/2})
A_5=A.subs({theta:theta_5,d:0.09465,a:0,alpha:-pi/2})
A_6=A.subs({theta:theta_6,d:0.0823,a:0,alpha:0})


Transformation=A_1*A_2*A_3*A_4*A_5*A_6

T_1=A_1
T_2=A_1*A_2
T_3=A_1*A_2*A_3
T_4=A_1*A_2*A_3*A_4
T_5=A_1*A_2*A_3*A_4*A_5
T_6=A_1*A_2*A_3*A_4*A_5*A_6

# End effector position
Xp=Transformation.col(3)
Xp.row_del(3)

# Taking individual x,y,z for the end effector position
x=Xp[0]
y=Xp[1]
z=Xp[2]

# Setting up the Jacobian Matrix

J=Matrix([[diff(x,theta_1),diff(x,theta_2),diff(x,theta_3),diff(x,theta_4),diff(x,theta_5),diff(x,theta_6)],
        [diff(y,theta_1),diff(y,theta_2),diff(y,theta_3),diff(y,theta_4),diff(y,theta_5),diff(y,theta_6)],
        [diff(z,theta_1),diff(z,theta_2),diff(z,theta_3),diff(z,theta_4),diff(z,theta_5),diff(z,theta_6)],
        [T_1[0,2],T_2[0,2],T_3[0,2],T_4[0,2],T_5[0,2],T_6[0,2]],
        [T_1[1,2],T_2[1,2],T_3[1,2],T_4[1,2],T_5[1,2],T_6[1,2]],
        [T_1[2,2],T_2[2,2],T_3[2,2],T_4[2,2],T_5[2,2],T_6[2,2]]])

q_current=Matrix([[0.0], [-math.pi/2], [-math.pi/2], [0.00001], [0.00001], [0.00001]])
J_initial=J.subs({theta_1:q_current[0],
    theta_2:q_current[1],
    theta_3:q_current[2],
    theta_4:q_current[3],
    theta_5:q_current[4],
    theta_6:q_current[5]}).evalf()
```

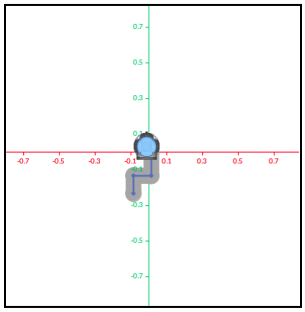
0.1914499999995885	−0.5196500000003497	−0.0946500000034975	−0.0946500000034975	0.082299999999177	0
0.3922498765	0	0	0	$8.22999999986283 \cdot 10^{-7}$	0
0	0.3922498765	0.3922498765	$-1.23500000050683 \cdot 10^{-7}$	$8.22999999955212 \cdot 10^{-7}$	0
0	0	0	$-9.99999999999558 \cdot 10^{-6}$	$9.99999999933333 \cdot 10^{-6}$	$9.99999999933333 \cdot 10^{-6}$
−1.0	−1.0	−1.0	0	−0.99999999995	−0.99999999995
0	0	0	0.99999999995	$9.99999999978913 \cdot 10^{-11}$	$9.99999999978913 \cdot 10^{-11}$

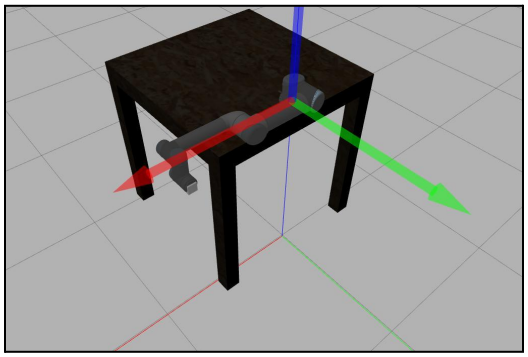
Fig8 Initial Jacobian Matrix

Forward Kinematics Validation

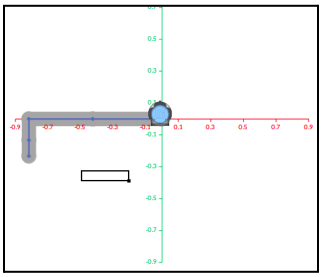


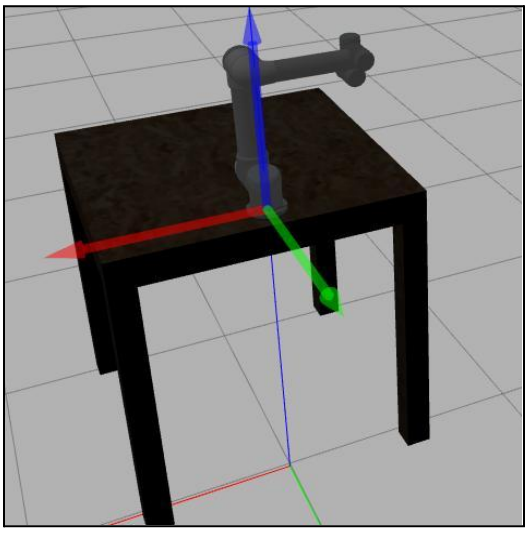
-0.08542	
-0.23290	
0.98132	
1.00000	



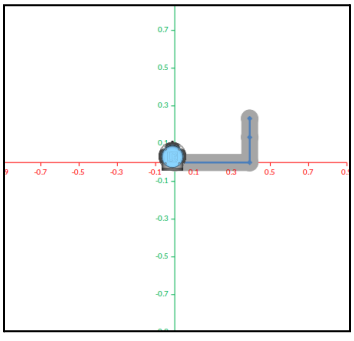


-0.81720	
-0.23290	
0.06280	
1.00000	





0.39220	
0.23290	
-0.36220	
1.00000	



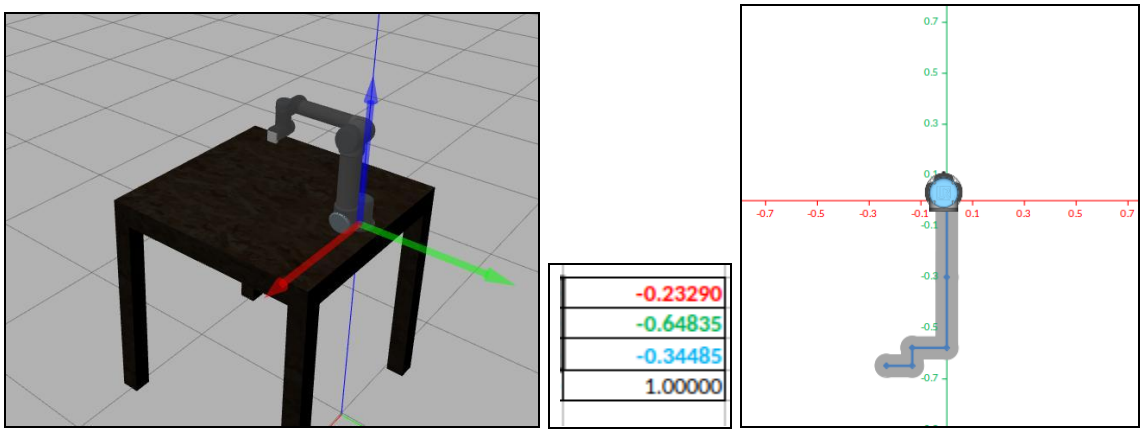


Fig9 Forward Kinematics validation

Inverse Kinematics Validation

We tried to use the geometric inverse kinematics to find the theta values for our simulations but we got 8 values of theta for each joint and had to do trial and error to get the values which were good for us, therefore we had to use some other method to find the joint angle hence we found the Newton Raphson Method to do so.

The Newton Raphson Method is referred to as one of the most commonly used techniques for finding the roots of given equations. It can be efficiently generalised to find solutions to a system of equations. Moreover, we can show that when we approach the root, the method is quadratically convergent. In this article, you will learn how to use the Newton Raphson method to find the roots or solutions of a given equation, and the geometric interpretation of this method.

The geometric meaning of Newton's Raphson method is that a tangent is drawn at the point $[x_0, f(x_0)]$ to the curve $y = f(x)$.

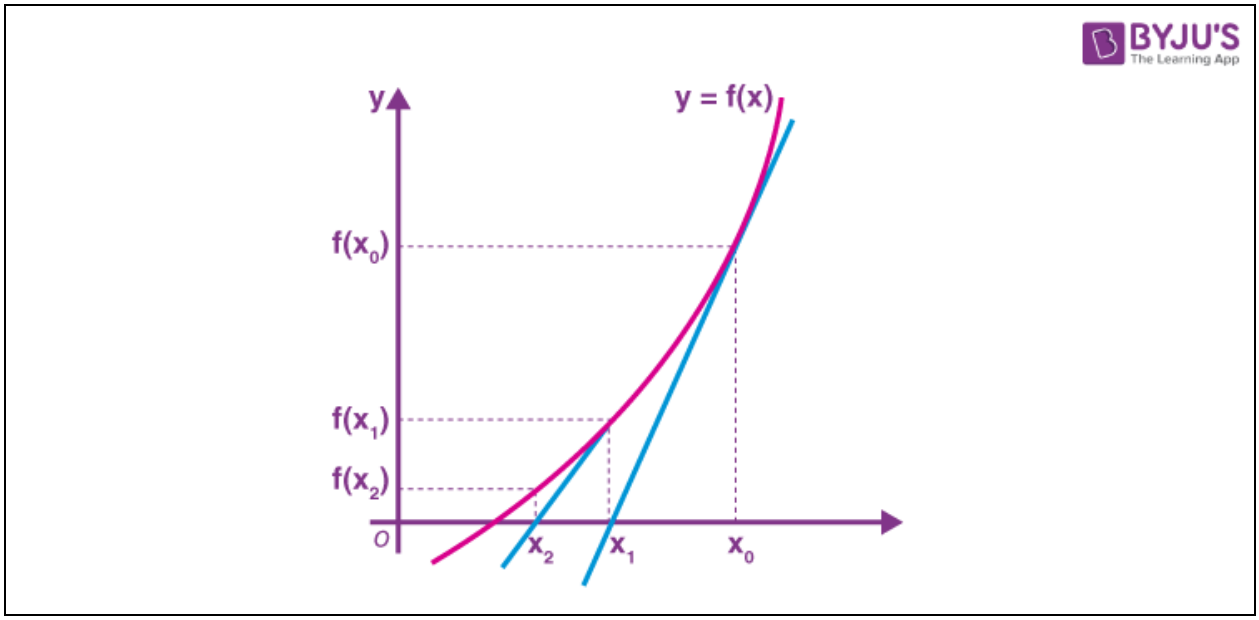


Fig10 Newton's Raphson method

It cuts the x-axis at x_1, which will be a better approximation of the root. Now, drawing another tangent at $[x_1, f(x_1)]$, which cuts the x-axis at x_2, which is a still better approximation and the process can be continued till the desired accuracy is achieved.

So we used the above method as our IK solver .

Its validation can be seen from our simulation in Gazebo.

Workspace Study

Our robot operates on a shop floor with 4 workstations, these stations are tables with parts on them basically, our robot is supposed to move between them, currently there are no obstacles on the pathway, so our mobile robot has to just avoid bumping into the tables.

The robot arm attached to the middle of the mobile robot, the official workspace provided by the manufacturer can be seen below.
The point cloud of the workspace was generated using a nested for loop where the joint variables were iterated using the following code.

```
x=[]
y=[]
z=[]
i=0
j=0
k=0
l=0
m=0
n=0
H=[0,0.5,1,1.5,2,2.5,3]
B=[2,0]
x_=T_6[0,3]
y_=T[1,3]
z_=T[2,3]
for i in H:
    for j in H:
        for k in H:
            for l in B:
                for m in B:
                    for n in B:
                        x.append(x_.subs({theta_1:i,
    theta_2:j,
    theta_3:k,
    theta_4:l,
    theta_5:m,
    theta_6:n}).evalf())
                        y.append(y_.subs({theta_1:i,
    theta_2:j,
    theta_3:k,
    theta_4:l,
    theta_5:m,
    theta_6:n}).evalf())
                        z.append(z_.subs({theta_1:i,
    theta_2:j,
    theta_3:k,
    theta_4:l,
    theta_5:m,
    theta_6:n}).evalf())
```

The corresponding sphere point cloud can be seen below that overlaps with the one provided by the manufacturer.

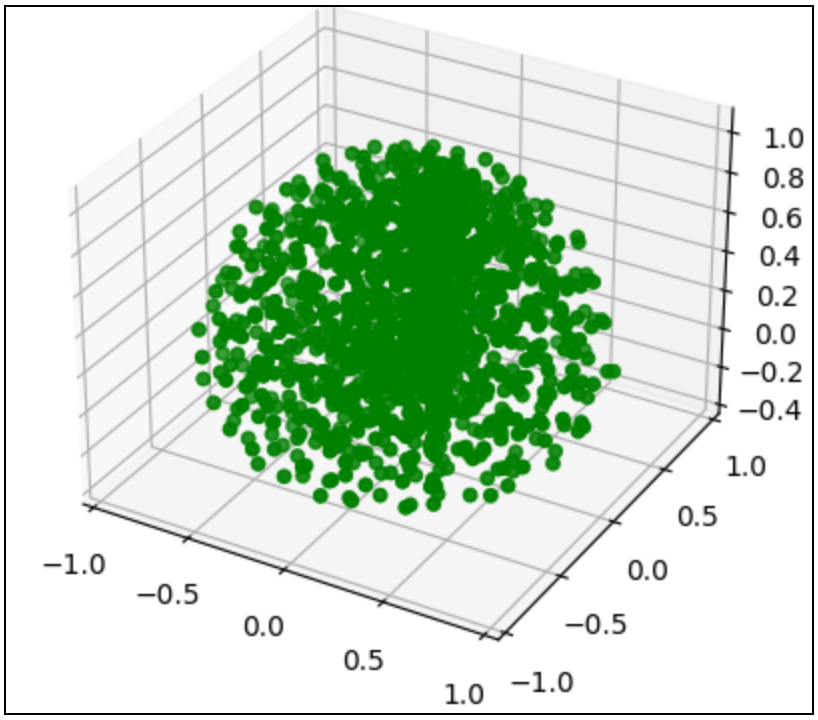


Fig11 Point Cloud of the workspace

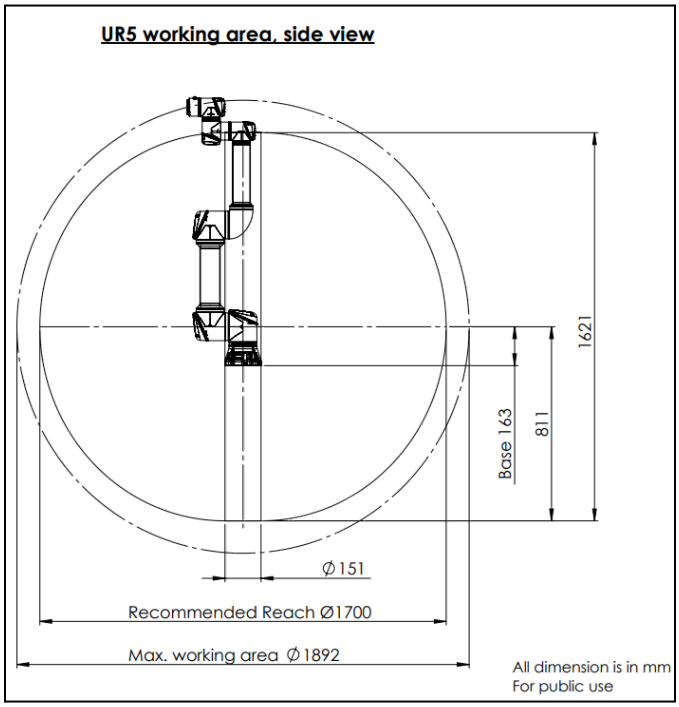


Fig12 Workspace by manufacturer

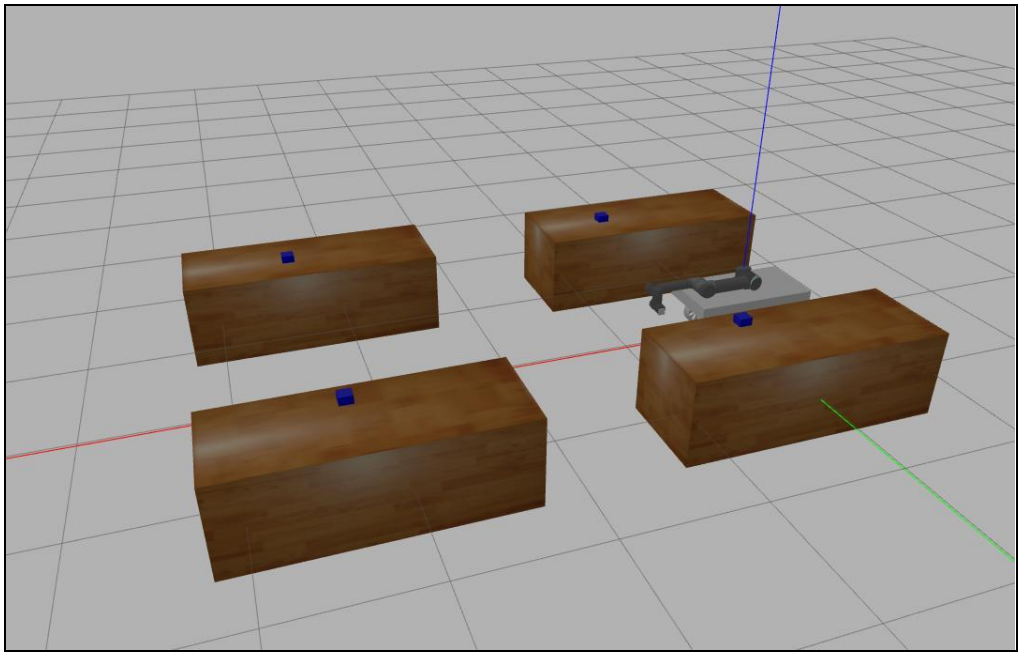


Fig13 Factory setup

Assumptions

- The following assumptions were considered in developing this project:
- Control of robot is not autonomous. It will be using TELEOP.
- Masses of the links are not taken into consideration.
- The item to be picked is a rigid body.
- The floor is flat and level. There are no inclines/obstacles.
- Inventory is completely known to the robot including the component locations.
- Maximum torques are provided to the joints to move freely without any
- constraints.
- No friction is considered between the joints.
- The path taken by the robot arm is just one solution among all other solutions, it may or may not be the optimum solution.

Control Method

A manipulator is a device used to manipulate materials without direct contact that mostly aims at replacing the human arm. The control of such manipulators requires

position tracking by a certain control strategy. The main aim of the control is to ensure that the robot will maintain a desired dynamic behavior.

In our robot, we have used a PID controller for all of the individual joints and the two steerable wheels of the mobile base. Each control loop corresponds to a drive unit, which actuates one axis of motion of the manipulator.

The robot manipulator control using PID controller has great importance in industrial automation. PID control scheme is one of the most general control types and probably the most used. It provides quick response, good control of system stability, and low steady-state error.

In the PID control scheme, the signal driving the actuator is made equal to a weighted sum of the error, the time integral of error, and time derivative error. PID controller has better performance while compared with the conventional PD controller.

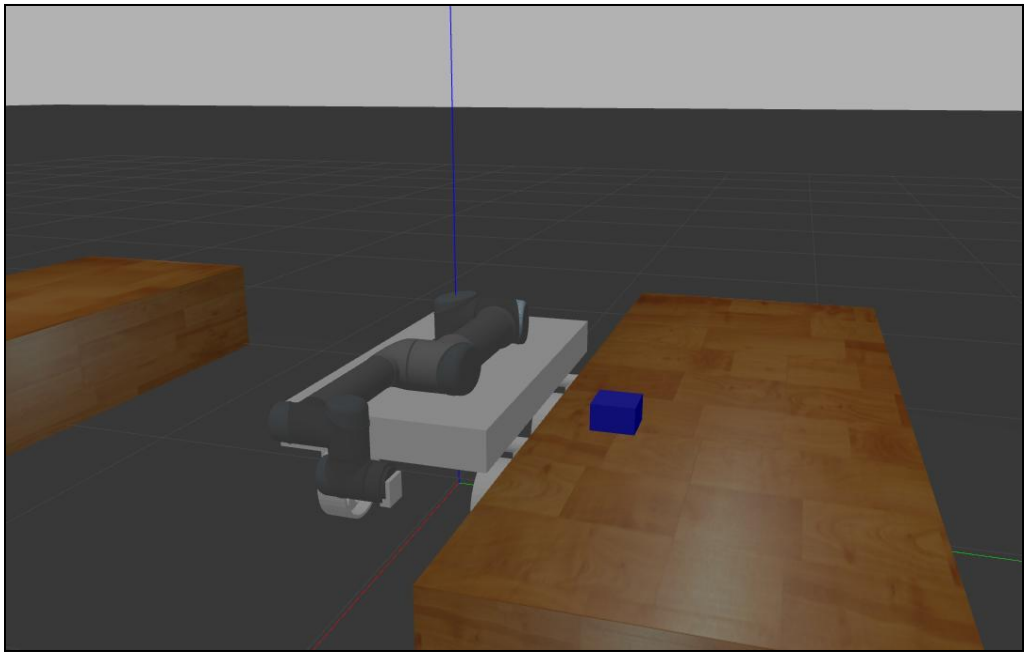
The PID control has a simple structure compared to other control algorithms and clear physical meaning for its three gains. The control performances are acceptable in most industrial processes. Most robot manipulators that are employed in industrial operations are controlled by PID algorithms that operate independently at each joint. The PID controller tuning is easier for robots. It can effectively be able to drive links with considerably large masses. The PID controller tuning is easier for robots. Because of all these advantages and for its simplicity the PID control method is used for ALIRO.

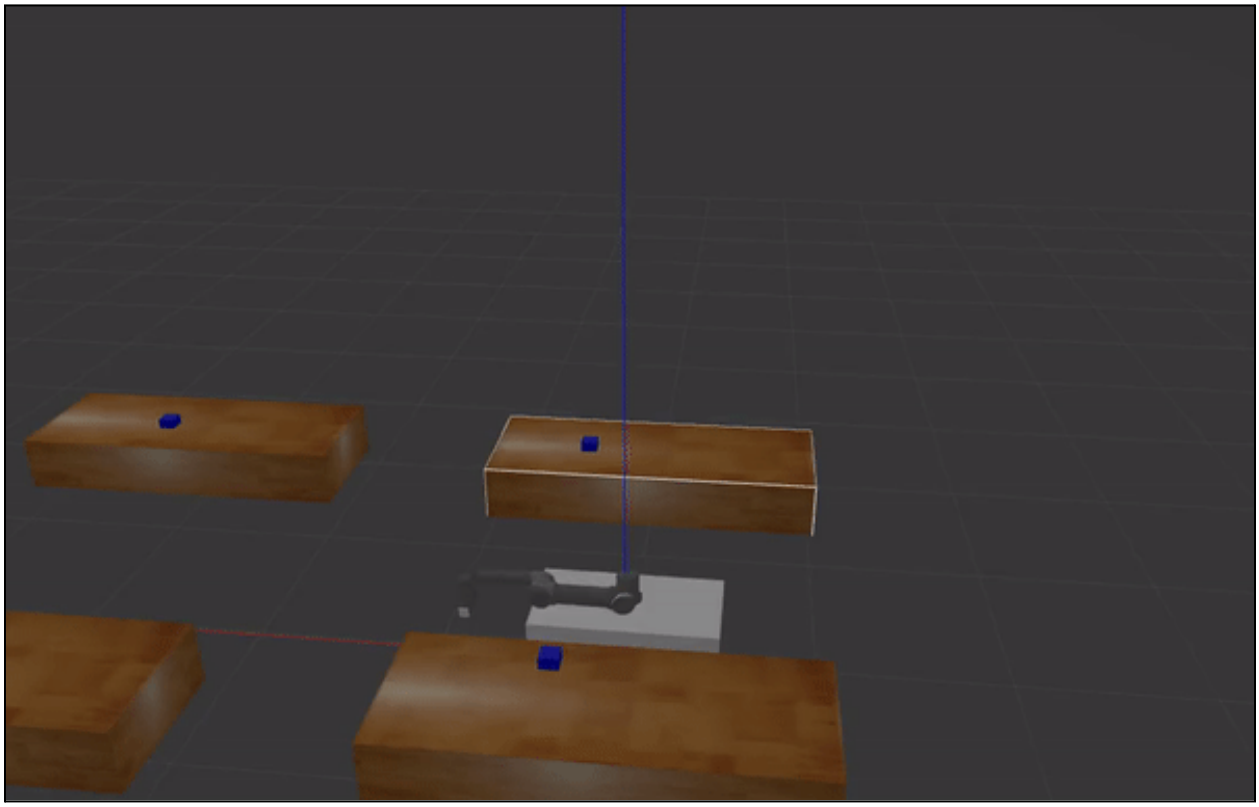
```
 # Publish all joint states ----------------------------------
joint_state_controller:
  type: joint_state_controller/JointStateController
  publish_rate: 50

# Position Controllers ---------------------------------------
shoulder_sweep_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_sweep_joint
  pid: {p: 1000.0, i: .001, d: 60.0, i_clamp_min: -100.0, i_clamp_max: 100.0}

shoulder_lift_position_controller:
  type: effort_controllers/JointPositionController
  joint: shoulder_lift_joint
  pid: {p: 3000.0, i: 100.0, d: 175.0, i_clamp_min: -400.0, i_clamp_max: 400.0}
```

```yaml
elbow_position_controller:
  type: effort_controllers/JointPositionController
  joint: elbow_joint
  pid: {p: 2500.0, i: 100.0, d: 100.0, i_clamp_min: -100.0, i_clamp_max: 100.0}

wrist_pitch_position_controller:
  type: effort_controllers/JointPositionController
  joint: wrist_pitch_joint
  pid: {p: 900.0, i: 10.0, d: 40.0, i_clamp_min: -400.0, i_clamp_max: 400.0}

wrist_roll_position_controller:
  type: effort_controllers/JointPositionController
  joint: wrist_roll_joint
  pid: {p: 900.0, i: 0.1, d: 10.0, i_clamp_min: -100.0, i_clamp_max: 100.0}

wrist_yaw_position_controller:
  type: effort_controllers/JointPositionController
  joint: wrist_yaw_joint
  pid: {p: 100.0, i: 0.1, d: 10.0, i_clamp_min: -100.0, i_clamp_max: 100.0}



#car--------


# gazebo_ros_control:
#   pid_gains:
#     left_rear_wheel_joint:
#       p: 100.0
#       i: 0.001
#       d: 1.0
#     right_rear_wheel_joint:
#       p: 1.0
#       i: 0.001
#       d: 1.0
#     left_front_wheel_joint:
#       p: 0.0
#       i: 0.001
#       d: 1.0
```

```
#     right_front_wheel_joint:
#       p: 0.0
#       i: 0.001
#       d: 1.0

#Controller 1
rear_motor_left:
  type: velocity_controllers/JointVelocityController
  joint: left_rear_wheel_joint
  pid : {p: 1000.0 , i: 0.0 , d: 0.0}
 rear_motor_right:
  type: velocity_controllers/JointVelocityController
  joint: right_rear_wheel_joint
  pid : {p: 1000 , i: 0.0 , d: 0.0}
 front_motor_left:
  type: effort_controllers/JointPositionController
  joint: left_front_wheel_joint
  pid : {p: 1000.0 , i: 0.0 , d: 0.0}

front_motor_right:
  type: effort_controllers/JointPositionController
  joint: right_front_wheel_joint
  pid : {p: 1000.0 , i: 0.0 , d: 0.0}
```

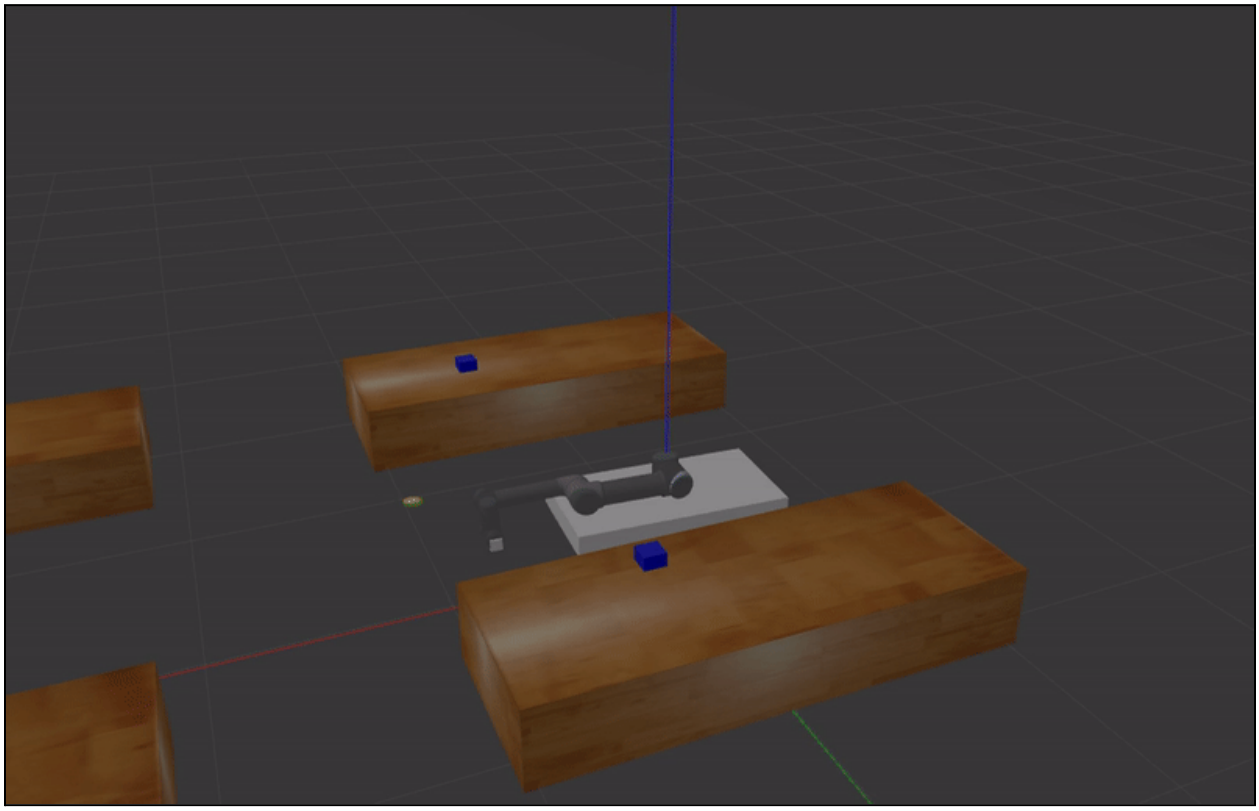
Gazebo & RViz Visualization





Problems Faced

- Due to low inertia of the mobile robot it moved when the UR5 arm moved.
- We planned to use a flat faced gripper but it was not working in gazebo so we used a vacuum gripper.
- All joint angles we got from the geometric method were not correct for the particular case of simulation.
- We had to use other inverse kinematics solver for our simulation.
- Controlling the sudden movement in the robots(jerks)



Lessons Learned

There is no doubt whatsoever that we learned a lot from this project and got familiar with different related concepts, but our overall feeling is a bit mixed. We did not calculate that even though we have related knowledge and enthusiasm about the topic, getting familiar with ROS takes a longer time than what we expected. During the period working on the project we spent a lot of time trying out different packages and ideas, but

most of them could not make it until the final version since we did not have time to implement/ connect the parts fully. The areas that we explored are listed below.

- To understand and calculate Forward and Inverse kinematics for a real-world robot application
- To learn simulated environments and model the real world in ROS - Gazebo & Rviz
- To understand link/joint parent-child relationships in CAD/Solidworks and how to model them.
- To understand ROS and its functionalities like “Move_it”.
- To pick and place objects avoiding obstacles in free space.
- To validate and verify calculated Forward and Inverse kinematics solutions using a developed simulated environment.
- To understand the integration of sensors, controllers in a simulated environment.

As it was mentioned the project did not go the way it was planned, however we still gained a lot of experience and knowledge that can be useful in further projects.

Conclusions

Our scope is highly relevant, nowadays most of the companies facing the same issues and challenges, introducing robotics to task such as transportation and pick and place to replace the human workforce, however the long term goal is to involve robotics in complex assembly structures as well, that’s the area that we would like to explore.This study or project made us understand the basic modeling of robotic system in the real world. We also have a brief understanding of the Kinematic and Dynamic principles.

We took this specific robot to understand rigid body transformation, Inverse Kinematics and Forward Kinematics, contact modeling, and grasping. Modeling this robot made us learn the links and joints mechanism in actual modeling scenarios. Out of other things, this project made us learn the grasping mechanism, trajectory, and path planning. Also, this project gave us the opportunity to learn more in ROS-Gazebo environments.

To be fair, many ambitious goals were set in the beginning, however we could not achieve them mostly due to time constraints. We did not clearly understand the difficulties of each milestone that was set. Even though we were able to complete some of them, connecting them to a real working simulation is another challenge that we could not meet. We created our workspace with the proposed workstations, then we were able to achieve our fall back goal, doing a pick and place but we could not make our mobile robot to move autonomously between the workspaces and we also did not have time to work on more complex assembly structures.

In conclusion, we still put a lot of effort into this project and we created a part of the proposed ideas, which took more time than expected. We also gained a lot of knowledge about different subparts that did not end up in the final version. The take home message is that we should have planned more ahead and a bit less with a more clear vision and cornerstones on the way. However, we are still glad about the chosen topic and satisfied with our end result, since we were able to implement the theory into practice.

Future Work

The future work would be achieving our ambitious goals, creating an autonomously working factory with assemblies and transportation. To achieve it a more systematic approach is needed, it would be useful to try to build everything together instead of integrating separated subparts.

Also, ROS is another area to deepen more into, in this project we mostly relied on online resources since we have limited knowledge. To gain confidence and explore the capabilities, an extended amount of different projects are needed, which is time consuming, however after we were able to build a knowledge base and a programming confidence a complex project should be more manageable than it was this time.

References

1. https://store.clearpathrobotics.com/products/universal-robots-ur5

2. https://www.universal-robots.com/articles/ur/application-installation/what-is-a-singularity/

3. Mathematical Modelling and Simulation of Human-Robot Collaboration-2020 International Russian Automation Conference (RusAutoCon)
4. http://rasmusan.blog.aau.dk/files/ur5_kinematics.pdf
5. https://byjus.com/maths/newton-raphson-method/