Hardware based Pick & Place Operation using UR5

ENPM 662 – Project 2 Technical Report

submitted by

Team members:

Tej Kiran, UID: 119197066
Dhinesh Rajasekaran, UID: 119400241
Arshad Shaik, UID: 118438832

Under the guidance of

Dr. Reza Monfaredi

& support from TAs for ENPM-662 course

Pavan Mantripragada
Adarsh Malapaka

DECEMBER 7, 2022

MAGE, UNIVERSITY OF MARYLAND
College Park, MD

Table of Contents

Table of Figures

Tables Index

1. Introduction

The main objective of this project is to implement an application based on basic pick and place operation using the Universal UR5 manipulator in the Robot Realisation Laboratory.

Utilising the core concepts from the 'Introduction to Robot Modelling' course, it is agreed among the team members to explore these concepts in a real-world hardware application. Upon several discussions with Prof.Reza Monfaredi and evaluating different available options at the University of Maryland labs, it is narrowed down to utilise Universal Robot's UR5 manipulator, a flexible robotic arm, to implement 'pick and place' application.

2. Scope Description:

The scope of the project is to implement an application based on pick and place operation using the UR5 robotic arm, utilising the concepts of forward kinematics, inverse kinematics, and the tools such as Solidworks, Gazebo and ROS.

Here the inverse kinematic and forward kinematics of the robot may be studied and explored. Gazebo implementation of pick and place may be performed using SOLIDWORKS model and our own ROS package.

Deploy the parameters derived for IK and FK on the official ROS package on real hardware UR5 in RRL.

3. Application

The project will implement the basic pick and place operation for the given co-ordinates of the end-effector. The ambitious and the fallback goals of the above application are described as below

Ambitious Goal:

a) Check availability of the end effector and move the robot arm to desired waypoints in the 3-D space, to perform an operation such as pick and place, hook etc

b) In the event of its availability of end-effector, basic pick and place of some objects placed in given waypoints may be performed or any other application based on prof recommendation.

c) The object location and its desired end location will be given, UR5 will reach the initial location, grab the object, and place it in the given desired location.

Fall Back Goals:

a) Simulation of UR5 in gazebo world with custom designed world in Gazebo with our own ROS package.

b) Implement basic movement of UR5 hardware in RRL where the robot may reach a given start location from any random arbitrary location and then move to a given end/desired location without any pick and place of objects.

4. Robot Type

The UR5 is a lightweight, adaptable collaborative industrial robot that tackles medium-duty applications with ultimate flexibility. The UR5 is designed for seamless integration into a wide range of applications. UR5 is also offered as an OEM robot system and with a 3-Position teach pendant.

- Main Technical Specifications:
- Payload Capacity: Up to 5 kg
- Weight: 20.56 kg
- I/O Power Supply Voltage: 12/24 V

5. DOFs and Dimensions

The Degrees of Freedom of UR5 is calculated as follows:

DOF = $m(N-1-J)+\sum_{i=1}^{J} fi$

Number of Links N = 6

Number of Joints J = 5

DOF in 3-D space m = 6

DoF = 6(6-1-5) + 5 * 1 = 5

The robot dimensions are as below:

- Max Reach: 850 mm
- Footprint: 149 mm

Movement		
Pose Repeatability per ISO 9283	± 0.03 mm	
Axis movement	Working range	Maximum speed
Base	± 360°	± 180°/s
Shoulder	± 360°	± 180°/s
Elbow	± 360°	± 180°/s
Wrist 1	± 360°	± 180°/s
Wrist 2	± 360°	± 180°/s
Wrist 3	± 360°	± 180°/s
Typical TCP speed	1 m/s (39.4 in/s)	

6. Robot Appropriateness for the task

UR5 is selected for the application, for the following reasons:

- Easy programming: Quickly set up and operate with intuitive, 3D visualization
- Collaborative and Safe: As this is our primitive project on robot modelling, UR5 being compliant to safety standards as per TUV (The German Technical Inspection Association), would be ideal for our project.
- Flexible Deployment: Easy to re-deploy to multiple applications, creating opportunity to further explore the possibilities in the upcoming course of robotic studies.
- Availability: The robot is expected to be available for the implementation and explore tasks associated with the project.
- ROS Package: compatible with ROS and official ROS package is available for UR5 which may be used during the hardware implementation phase. The ROS package that we

Figure 1: Universal UR5 arm

developed will be used for demonstration in Gazebo and while deploying in actual hardware, for safety reasons the official package may be used.

7. CAD Models

UR5 Cad Model is downloaded from the official website of Universal Robots and used for the simulation.

The downloaded models can be seen at the GitHub repository:

https://github.com/stark-2000/UR5_Pick-Place_Hardware

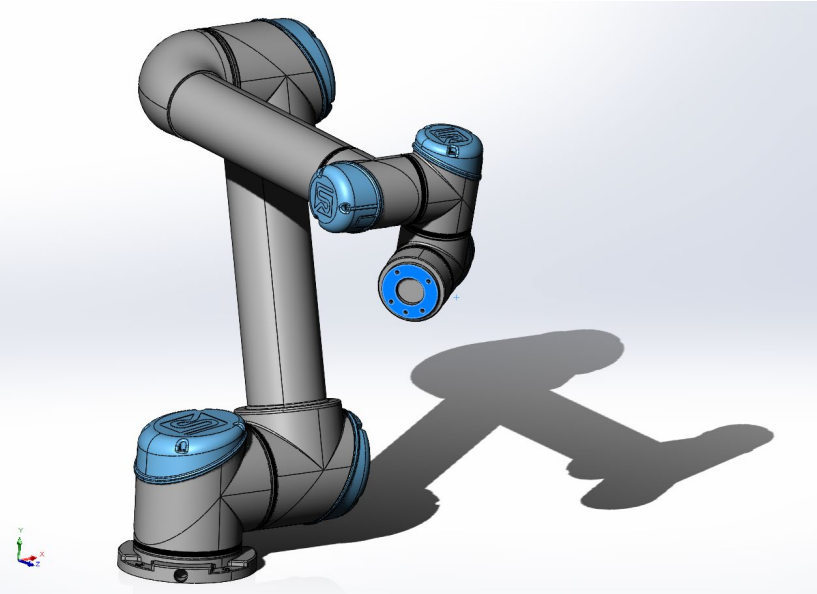

Figure 2: UR5 Solidworks Model

8. D-H Parameters

The D-H parameters are computed as follows:

Table 1: D-H Parameters

Joint	d	alpha	a	theta
J1	0.089159	pi/2	0	Θ1
J2	0	0	-0.425	Θ2
J3	0	0	-0.39225	Θ3
J4	0.10915	pi/2	0	Θ4
J5	0.09465	-pi/2	0	Θ5
J6	0.0823	0	0	θ6

9. Forward Kinematics

End effector configuration for UR5 home location (all joint angles 0) is calculated from the

Table 2 Transformation matric for end effector

```
[[ 1.          0.          0.         -0.81725001]
 [ 0.          0.         -1.         -0.19145   ]
 [ 0.          1.          0.         -0.005491  ]
 [ 0.          0.          0.          1.        ]]
```

python program as:

10. Inverse Kinematics

Inverse Kinematics gives us a method for finding the joint angles given an end effector configuration This can sometimes be done analytically (geometrically), but this is often difficult. Hence, a numerical method - Newton-Raphson method – has been used for inverse kinematics.

Numerical Inverse Kinematics

Inverse kinematics problem can be viewed as finding roots of a nonlinear equation:

$T(\theta) = X$

Many numerical methods exist for finding roots of nonlinear equations

The standard Newton-Raphson method for solving $x = f(\theta)$, where $\theta \in Rn$ and $x \in Rm$.

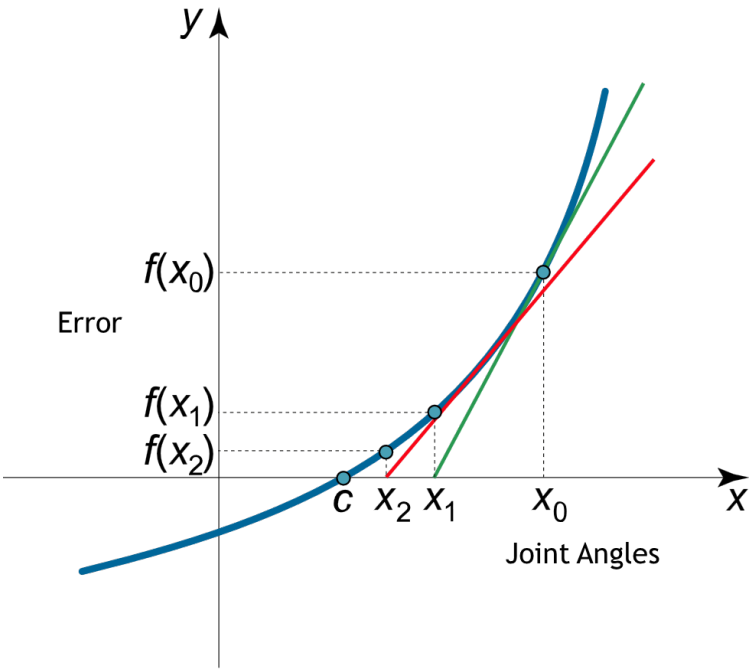

Figure 3: Convergence graph - Error & Joint angles - Newton-Raphson method for IK

Joint update equation is (Newton – Raphson method)

q = q + step*deltaq

deltaq = jacobian * ΔError (as end effector Transformation Matrix Difference)

Jacobian = error gradient between q± Δ (as end effector Transformation Matrix Difference)

Damped Least Squares update is

$\Delta q = J.T * Pinv(J*J.T + \lambda*I) * \Delta Error$

11. Forward Kinematics Validation

The co-ordinates of the end frame calculated, match with the figure shown here, geometrically.

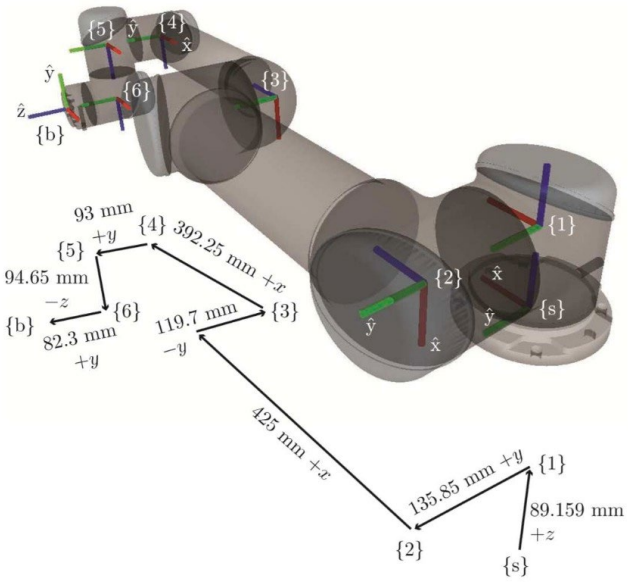

Figure 4: FK Validation

12. Inverse Kinematics Validation

A guess pick location is provided as shown below, with which the IK solver will converge to the actual position in a certain number of iterations. The below diagram shows that it converges in the 16 iterations.

```
Actual Pick location   :  [-1.57079633 -1.04719755  1.57079633 -0.52359878  1.57079633  1.57079633]
Guess Pick location    :  [-2.41660973 -2.0943951   1.57079633 -0.57119866  1.25663706  0.8975979 ]
Running IK for Pick location :
  0%|
Computed Pick location :  [-1.57079325 -1.04722709  1.57084178 -0.52362947  1.57082315  1.57074342]
```

Number of iterations

```
| 16/10000 [00:01<16:30, 10.08it/s]
```

13. Workspace study

The workspace of a UR arm is spherical, and in the working area diagrams it's represented with two concentric circles, a smaller one labelled "Recommended Reach" and a slightly larger one labelled "Max. working area". In the centre of the workspace, directly above and below the base joint there is a column, inside which there are also some restrictions on robot movement. The example below is from the UR5e robot working area diagram that can be found here:

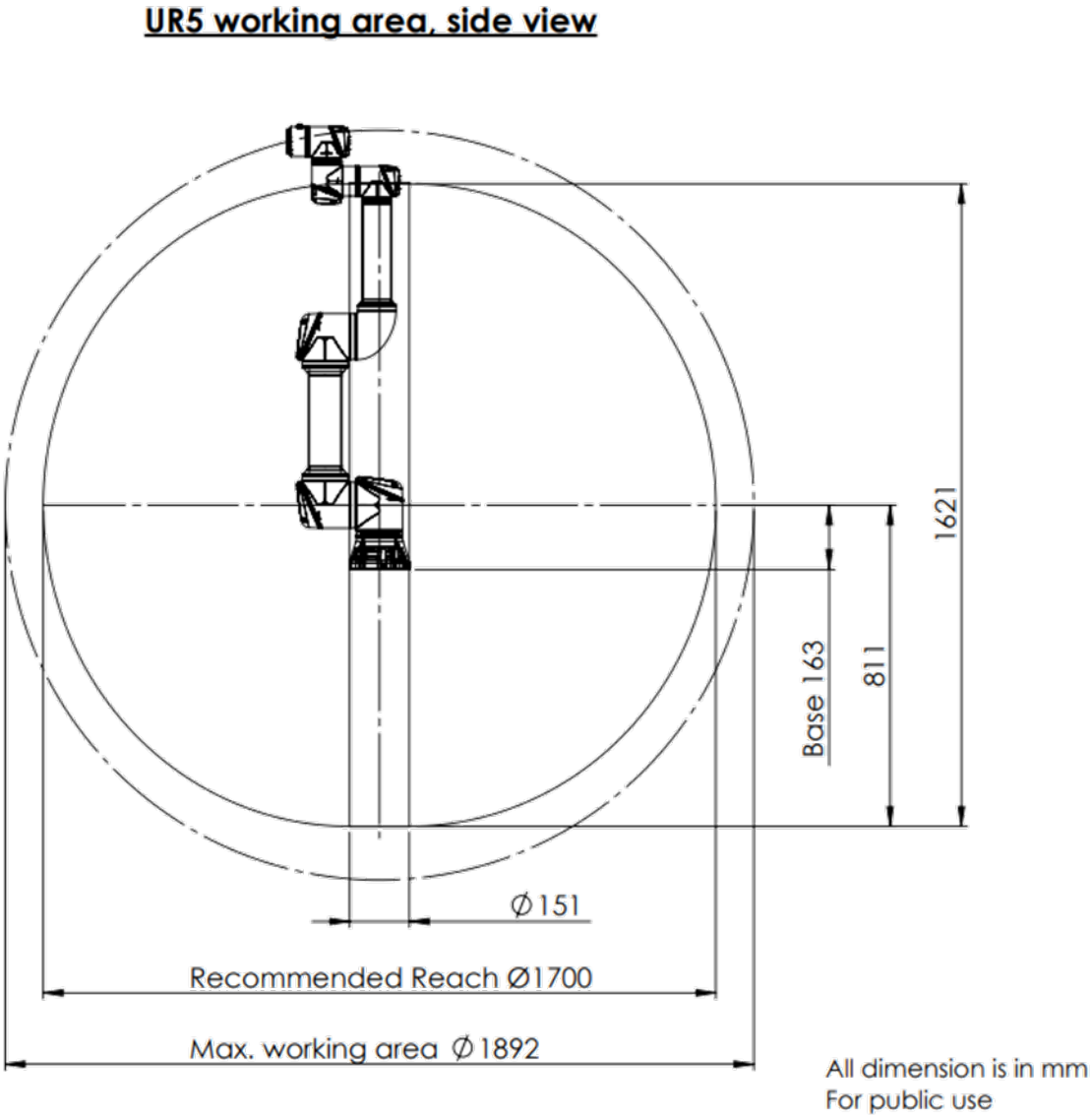

Figure 5: UR5 workspace area

14. Assumptions

The following assumptions are made during the execution of the project.

- Numerical Inverse Kinematics calculation requires an initial guess which can be solved using a planner in real world applications
- Joint constraints are ignored in IK calculation
- Environmental/workspace constraints will be handled by the planner.
- Numerical Inverse Kinematics will converge in real time

15. Control Method

The control method implemented in the project is 'PID Control'. The below is the snippet of the control implementation.

```
#https://classic.gazebosim.org/tutorials?tut=ros_control

 # 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}

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}
```

Figure 6: Control method - PID control

16. Gazebo & RViz Simulation

The final ROS package and other related artifacts are archived at the following location

https://github.com/stark-2000/UR5_Pick-Place_Hardware

Gazebo simulation using own IK solver:

The simulation videos are archived in the folder – 'results' – at the following GitHub repo:

https://github.com/stark-2000/UR5_Pick-Place_Hardware

MoveIt joint control with solution generated by our IK

The simulation videos demonstrating the application of 'MoveIt' ROS package, utilising our own developed IK plugin, are archived in the folder – 'results' – at the following GitHub repo:

https://github.com/stark-2000/UR5_Pick-Place_Hardware

Implementation on real hardware – UR5

The forward and inverse kinematics, developed above, are implemented on real hardware – UR5 and validated against the expected results as shown in the above simulation videos.

The following is the recorded video on the real hardware.

https://github.com/stark-2000/UR5_Pick-Place_Hardware

17. MoveIt – IK Planner Settings

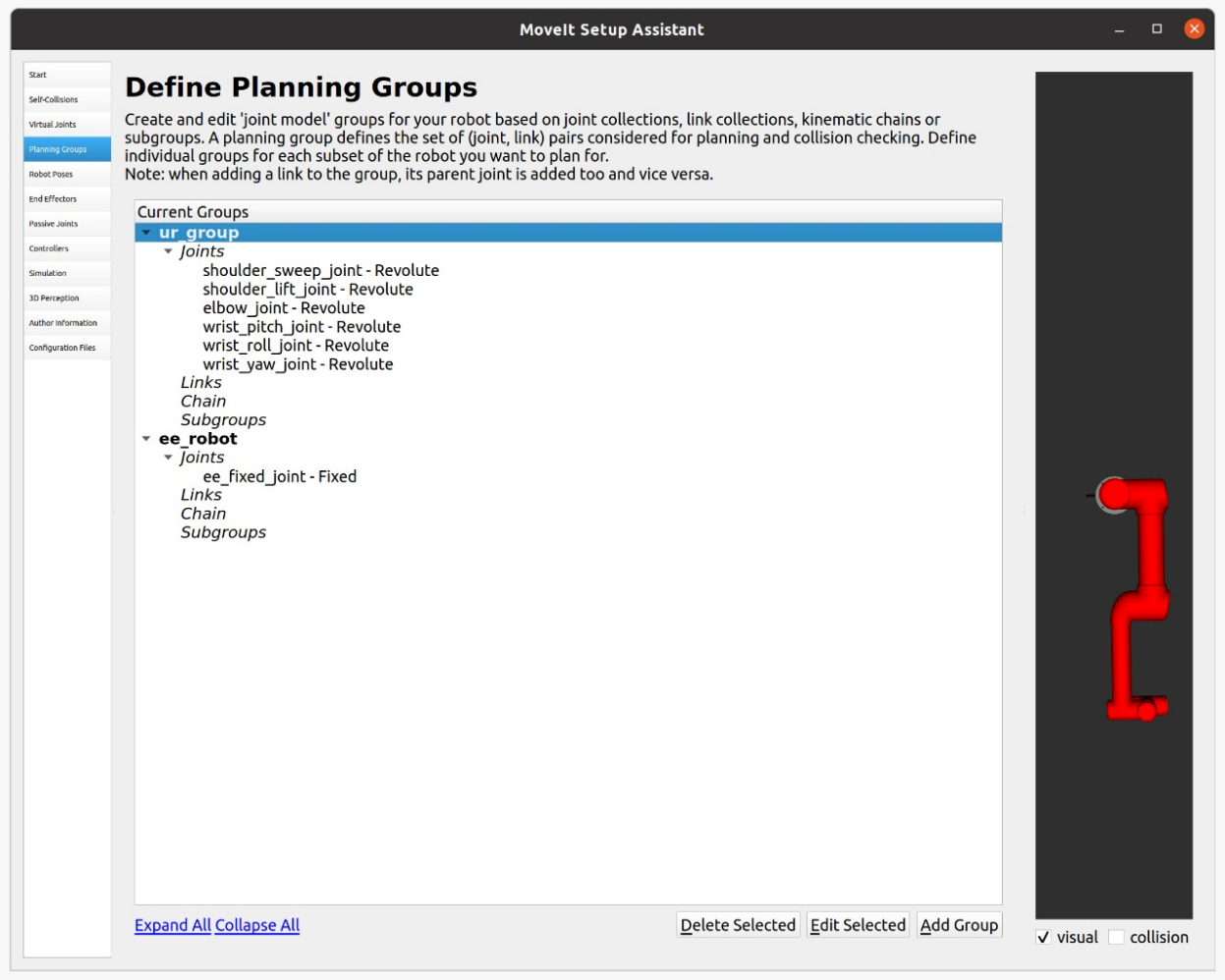

Figure 7: MoveIt - IK Planner setting

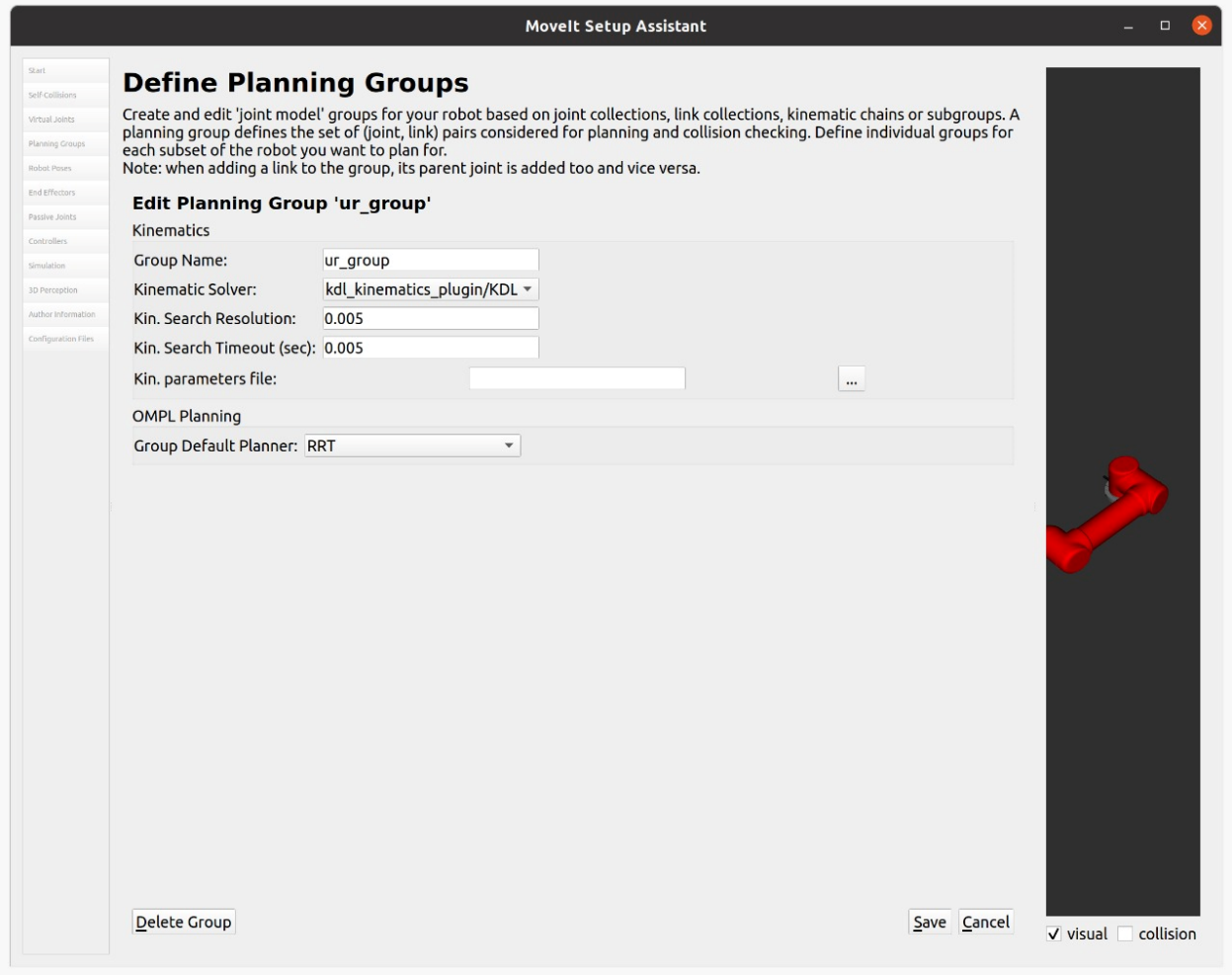

Figure 8: MoveIt - IK Planner setting (2)

18. MoveIt – IK Planner Settings – Own IK plugin:

The below diagram shows that our own IK plugin is selected for MoveIt implementation.

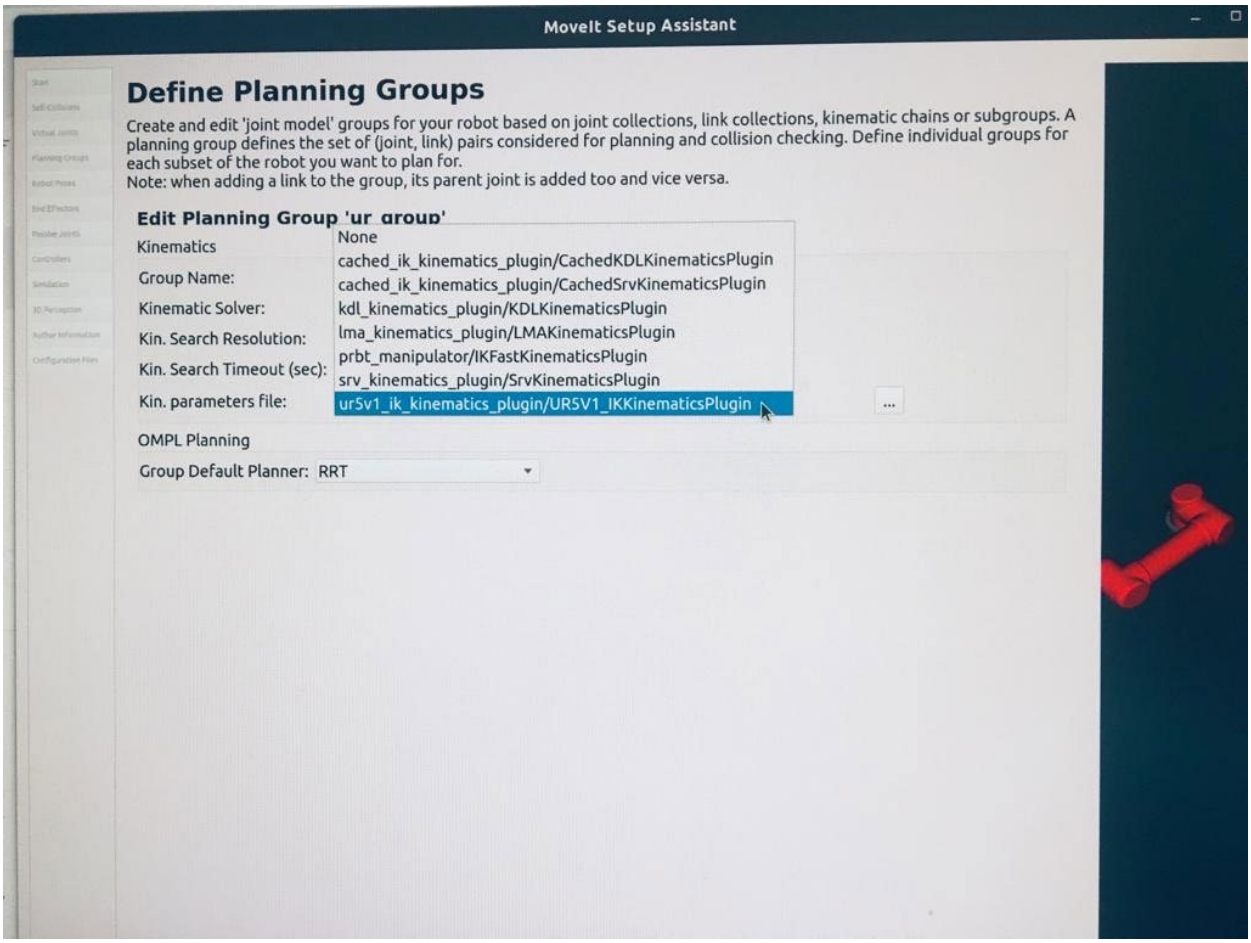

Figure 9: MoveIt - IK Planner setting - Own IK Plugin

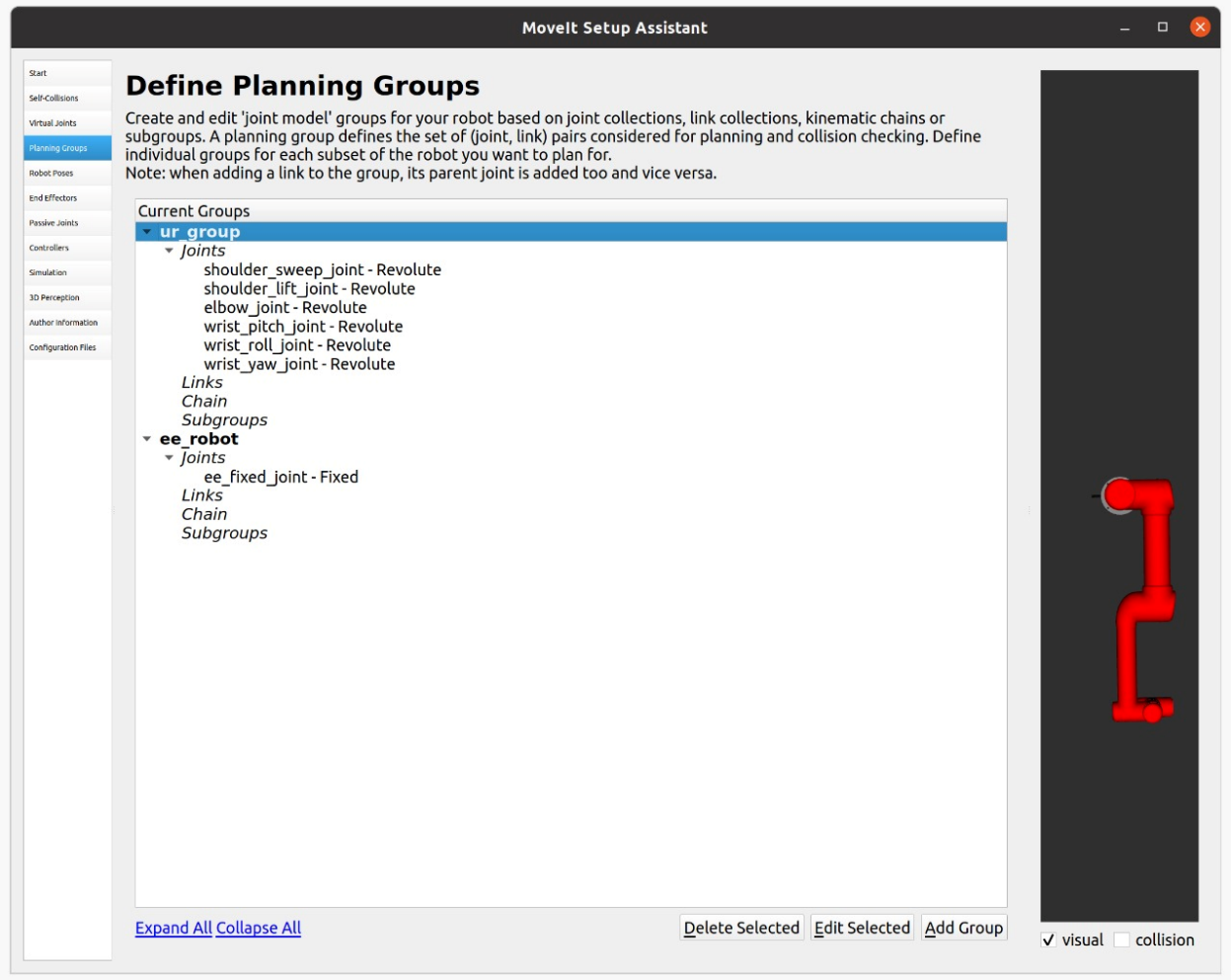

Figure 10: Planning groups

19. Problems Faced & Lessons Learned

- Mass Inertia problems in the URDF generation from Solidworks
- MoveIt planning group creation
- Adding MoveIt transmission controllers to URDF
- Using our own IK solver with MoveIt planner
- IK logic adaptation from python code to C++ ROS plug-in
- Solving IK using numerical methods
- Damped Least square method takes more time to converge than direct Newton-Raphson method

Lessons Learned:

- Make sure to allow MoveIt setup assistant to overwrite URDF with transmission controllers

- Make sure to turn on Gazebo simulation before publishing commands to controller topics
- IK solver needs additional refinements and constraints validations to make it work with MoveIt planners

20. Conclusions

The main objective of the project is to implement an application based on basic pick and place operation using the Universal UR5 manipulator in the Robot Realization Laboratory.

The solidworks model of UR5 is taken and used for pick and place simulation

- Created a ROS package and implemented FK, IK
- The UR5 is implemented with MoveIt package in Rviz

21. Future Work

- The project will be further developed to work on actual hardware.
- Own IK solver plugin to work with MoveIt planner

22. References

1. https://docs.fetchrobotics.com/manipulation.html
2. https://gramaziokohler.github.io/compas_fab/0.21.0/examples/03_backends_ros/08_ros_create_moveit_package_from_custom_urdf.html
3. https://ros-planning.github.io/moveit_tutorials/doc/moveit_cpp/moveitcpp_tutorial.html#the-entire-code
4. https://ros-planning.github.io/moveit_tutorials/doc/pick_place/pick_place_tutorial.html#the-entire-cod
5. https://roboticscasual.com/ros-tutorial-how-to-create-a-moveit-config-for-the-ur5-and-a-gripper/
6. https://roboticscasual.com/ros-tutorial-pick-and-place-task-with-the-moveit-c-interface/
7. https://gramaziokohler.github.io/compas_fab/0.21.0/examples/03_backends_ros/08_ros_create_moveit_package_from_custom_urdf.html#3-8-3-add-virtual-joints

23. Individual Contribution:

Tej Kiran:

- Developed IK solver for UR5
- Added controller
- Added hardware interfaces

- Written our own FK and IK algorithms based on D-H parameters
- Implemented 'Pick & Place' workflow using the above functions and simulated the motion in Gazebo

Dhinesh:

- Created a UR5 MoveIt package using MoveIt setup assistant
- Used MoveIt Pick and Place API to move an abstract object in Rviz simulation environment
- Explore the solidworks model for UR5

Arshad:

- Created launch files, addition of hardware interfaces and controllers to URDF
- Calculated DH parameters Table
- Created joint angle publisher module
- Test and validation of forward kinematics, inverse kinematics algorithms,
- Test and validation of MoveIt ROS package in Rviz
- Project proposal, report, and ppt

24. Acknowledgements:

We would like to thank Dr. Reza Monfaredi & TAs – Pavan & Adarsh, for their valuable guidance in accomplishing this project.