Github code repository

Internship Report

Project report for internship at Healthcare technology innovation centre (HTIC), IIT-Madras

Submitted by
Adwait P. Naik
(*Intern, Spine robotics team*)

Supervised by
Mr. Manojkumar Lakshmanan
(*Project lead, Spine robotics team*)
&
Mr. Shyam A
(*Spine robotics team*)

Github code repository

Contents

Github code repository

Robotic Operating System (ROS) based tasks

Objective - The tasks were performed to explore different features, capabilities and applications of ROS that mainly involved -

1) In-depth study of ROS and preparing presentation [presentation1] [presentation2]

2) Exploring different packages supported by ROS.

3) Assignments based on the UR5 robot [code / output]
   (a) Simulating UR5e robot into rviz. [code/ output]
   (b) Trajectory generation and inverse kinematics with UR5e. [code/ output]
   (c) Code to rotate UR5e in circular motion. [code/ output]
   (d) Creating Objects in rviz. [code/ output]
   (e) Object manipulation in rviz. [code/ output]

4) some try-outs [code/ output]

Analysis & Observations -

1. Simulating UR5e robot into rviz.

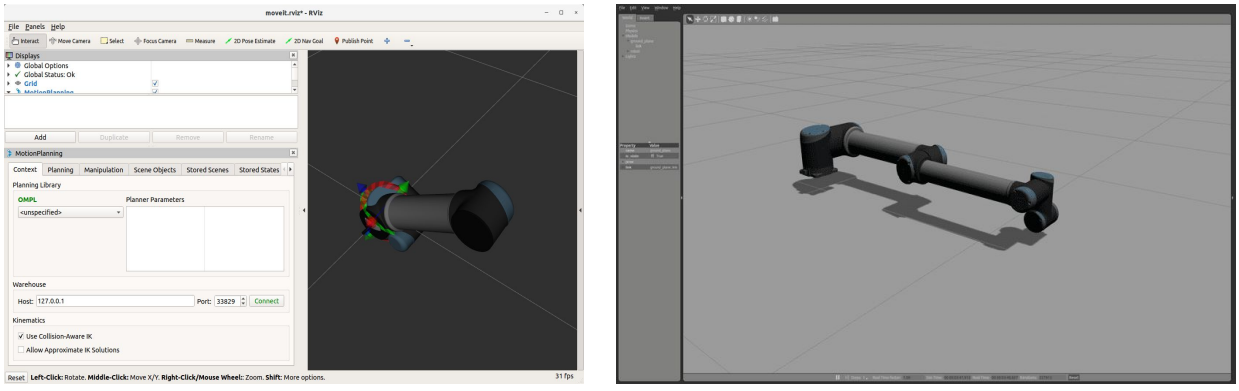


Fig. 1 UR5_e simulation in rviz

Steps

1) Install ROS, moveit package and UR modern driver package from github.

Github code repository

2) create a catkin workspace to generate the .launch files.
3) paste this in terminal **roslaunch ur5_moveit_config moveit_rviz.launch**

2. Trajectory generation & inverse kinematics

In this task the robot is made to come to its default (*home**) position by providing the point coordinates. Generally, the robot is made to return to the home position by following different set of way-points each time. [code] [video]

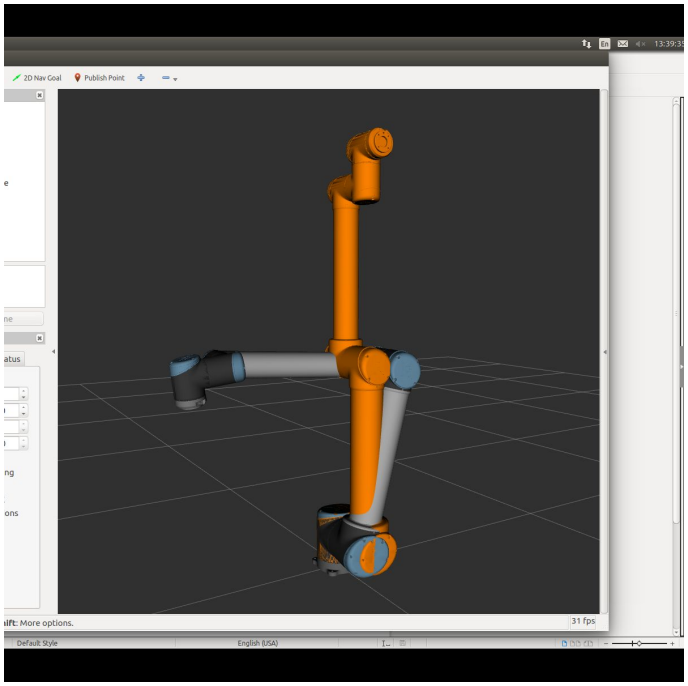


Fig. 2 UR5 at home* position

*The home position can be set manually by giving the joint coordinates.

3. UR5e in circular motion

The robot is programmed to form a circle. [code]

[Github code repository]

4. Creating Objects in rviz [code]

This task involves creating solids, meshes in rviz by giving different orientation and positions.

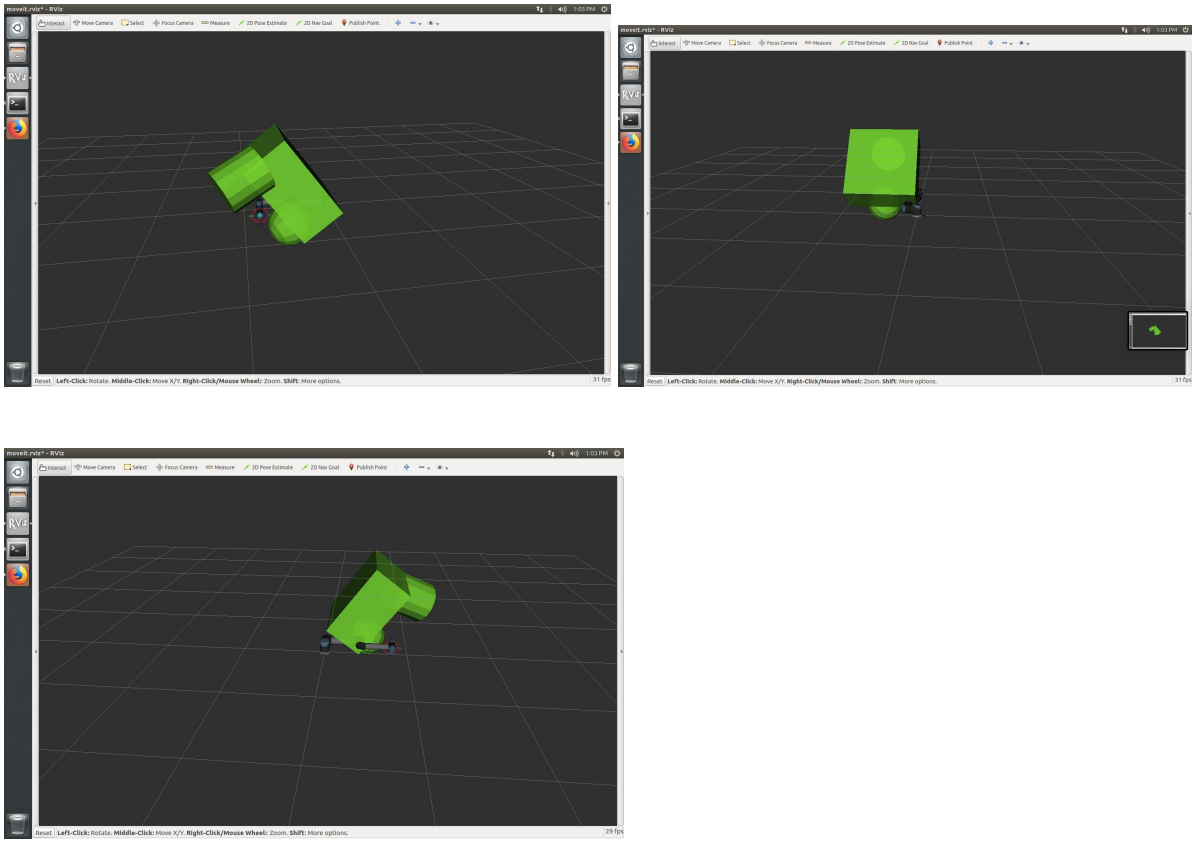


Fig. 3 Solids in rviz

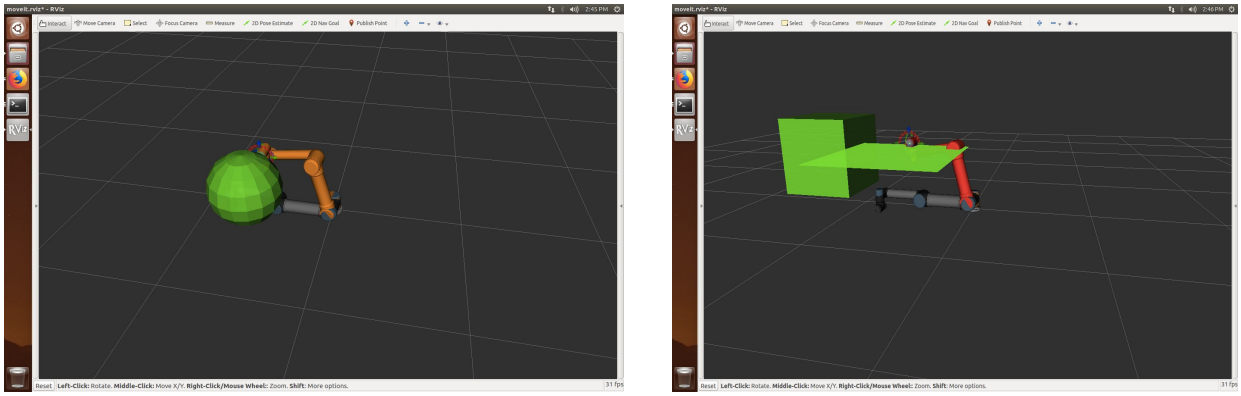


Fig. 4 Solids in-collision with the UR5 robot

[Github code repository]

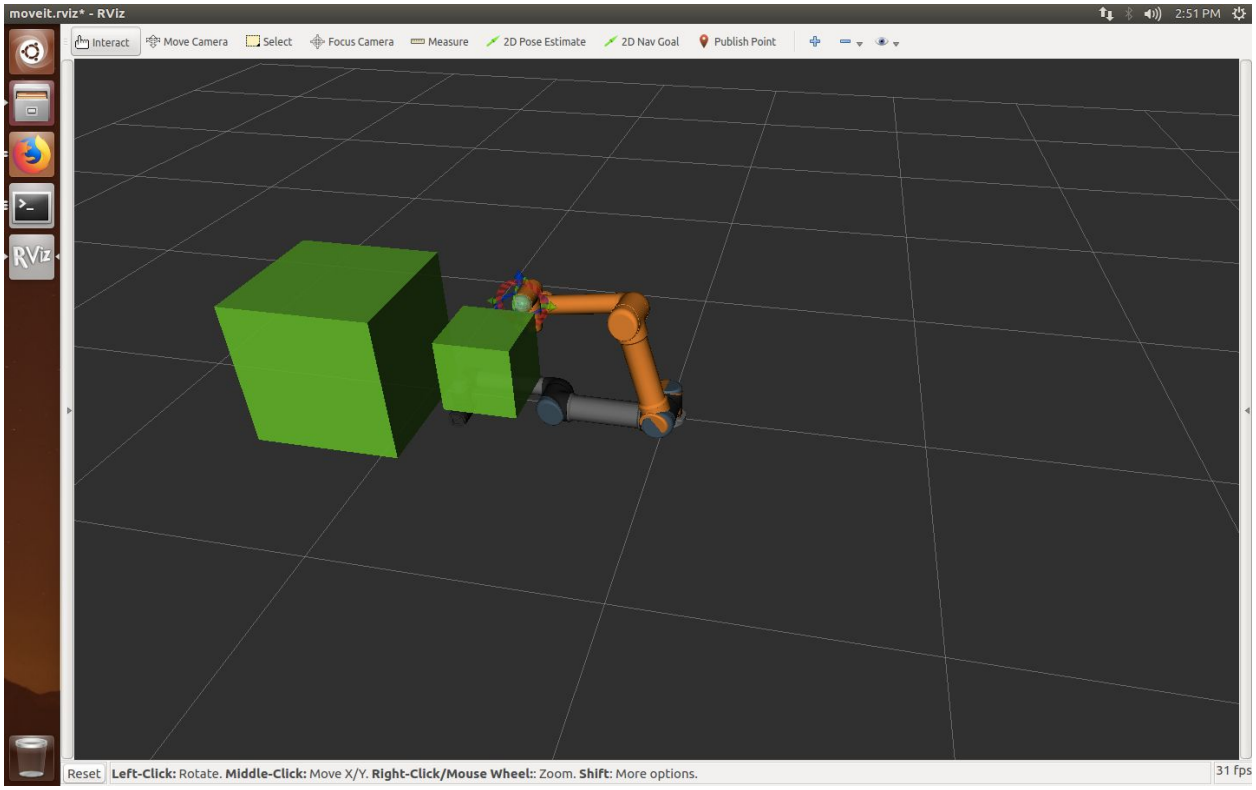


Fig. 5 Solids with the robot

5. Object Manipulation [code]

In this task the object (*cube*) is picked by the robot and moved by the robot from one point to another.

Github code repository

Tasks based on remote data acquisition from the robot (RTDE)

Objective - To acquire the joint velocity, joint angles, temperature, controller status, and controller version.

What is RTDE ?

RTDE or Real-Time Data Exchange is an interface for connecting, controlling and fetching data related to the robot, remotely with a desktop. It can be accessed using specific port numbers 30004 and 30002.

Key features

1) Real-time synchronization: The RTDE generally generates output messages on 125 Hz. However, the real-time loop in the controller has a higher priority than the RTDE interface.

2) Input messages: The updating of variables in the controller can be divided into multiple messages. One can have one message to update everything or a message per variable or any division in between.

3) Runtime environment: An RTDE client may run on the UR Control Box PC or on any external PC. The advantage of running the RTDE client on the Control Box is no network latency.

Github code repository

TCP calibration and Accuracy testing

Objective - To solve both robot accuracy and workpiece position error problems, this is only so if the end frame, and tool frame are accurately known with respect to each other.

What is TCP calibration? Why is it a critical aspect to be implemented?

The TCP is defined by the exact translational and rotational difference between the robot flange frame and the tip of the end effector.

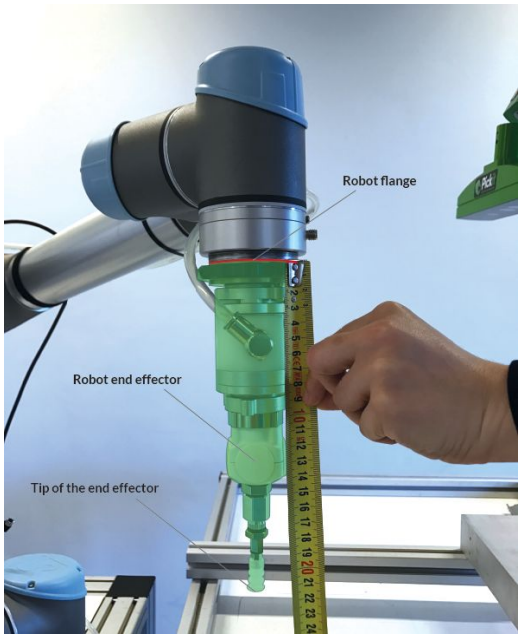


Fig. 6 TCP calculation (*image courtesy - pickit3d*)

TCP is a critical aspect before deploying the robot into the production process or implementing any application using the robot. Roughly TCP is nothing but mapping of End-effectors' reference frame with that of robots' base to obtain the relative difference between tools' movement with respect to the base. The calibration appears to solve both **robot accuracy** and **workpiece position error problems**, this is only so if the sensor frame, end frame, and tool frame are accurately known with respect to each other.

Mainly for calibrating the manipulators like UR5 robots include: touching reference parts, employing distance sensors, and using computer vision based techniques. Most of them are time consuming

Github code repository

and require expensive apparatus, unlike the teach-position method which needs no sensors, cameras or any sophisticated equipment.

Determining the tool center point

- **Initial Setup**

Before carrying out testing, the robot has to be properly installed by a certified technician. Some steps are listed below to be strictly followed to avoid potential damage to the environment and the robot.

a. Ensure the robotic arm has ample space for movement.
b. Ensure the docking is carried out in a proper way by the technician.
c. Ensure the TCP calibration is done.
d. The surface table shouldn't be touched specifically when the testing is going on.
e. Abstain going into robot's vicinity when the testing takes place.

- **Surface alignment (Plane alignment)**

Surface alignment requires the plane of the surface table (*Fig. 1*) to be aligned parallel to robot's (*Fig. 2*) plane or perpendicular to the needle attached to the end-effector.

Fig.1 Surface Table

Fig.2 UR5 Robotic Arm

[Github code repository]

- **TCP calibration (Tool Centre Point)**

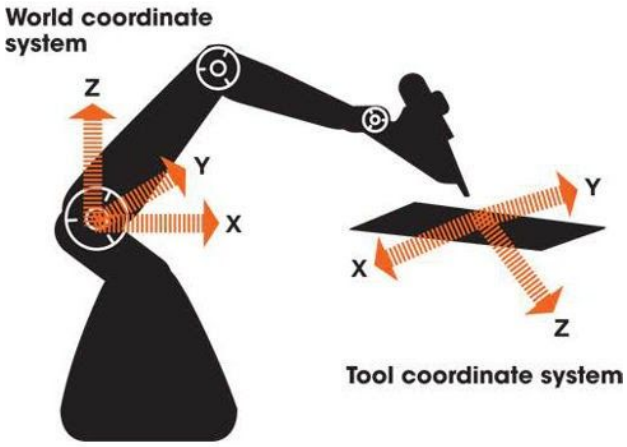
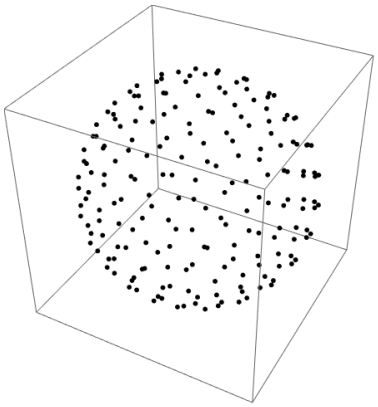


Fig.3 coordinate system mapping

Since the TCP point is shifted from robot to needle which is placed at the end effector, there is a need of calibration step to precisely locate its position and orientation. The protocol for this calibration is done by using UR5 system. 2 stage calibration is done (one for position and one for orientation) to avoid any system error.

- **Algorithm for testing protocol**

The initial step is to sample points from the plane which was made in line with the robot. The point is sampled by positioning the robot with the help of a needle using a graph sheet. Then using the position information a virtual sphere is constructed, which contains different entry points but same target point. This module is to create different angulation for robot position. Angle between entry points and target point is calculated before and after robot position to verify its accuracy. Since the whole system maintains the same target point, which is sampled from graph physical error can also be computed easily.



Github code repository

- **Procedure**

    1. The surface alignment and TCP calibration is done initially in order to nullify any physical error from the system.
    2. A point is sampled from graph sheet using a needle which is attached to the end effector of the robot.
    3. Construction of virtual sphere, generating different target points.
    4. Using constraint IK solutions, position the robot using a pair of points (Entry and Target) generated from sphere.
    5. Measure the physical difference from the graph sheet (Negligible error - 0.5 mm)
    6. Note down the UR5 system error of Entry, Target and Orientation before and after robot position. (Since all points are sampled from robot space, there is no need for registration)

- **Anomalies**

    a. We observed a bend in the needle with an error of 1mm, which created angulation error at times.

- **Calibrated setup**
    a. TCP point was calibrated with less than 0.2-0.3 mm error and same procedure was repeated
- **Results**
    a. All the results were tabulated and shared as a google sheet.

Accuracy Testing for UR5e with REST Protocol

The above setup was maintained for testing the REST protocol in which the data points are randomly sampled from the robot workspace. Those points were passed through a REST api upon which the robot position itself. The position accuracy was estimated between actual entry location and desired entry location. Nearly 110 points were sampled for this testing protocol and all the results were tabulated and shared as a google sheet.