Technical University of Cluj-Napoca

ROS Project

Student name: *Pacuraru Fabian-Virgil*

Course: *Robotics Control Systems* – Professor: *Anastasios Natsakis*
Due date: *January 17th, 2023*

Robotic platform

I chose to use a **panda robot** for my project because I found a step by step tutorial about **moveit** which was using this platform. My choice for the platform turned out not to be good because the tutorial was quite old, the git repositories that it was referencing changed and I struggled a lot to make some things work.

Important resources

Learning materials.

(*1*) ros basics

(*2*) moveit tutorial

(*3*) panda programming guide

Repository links.

(*1*) moveit msgs

(*2*) moveit resources

(*3*) geometric shapes

(*4*) srdfdom

(*5*) moveit

(*6*) rviz visual tools

(*7*) moveit visual tools

(*8*) moveit tutorials

(*9*) panda moveit config

Task description

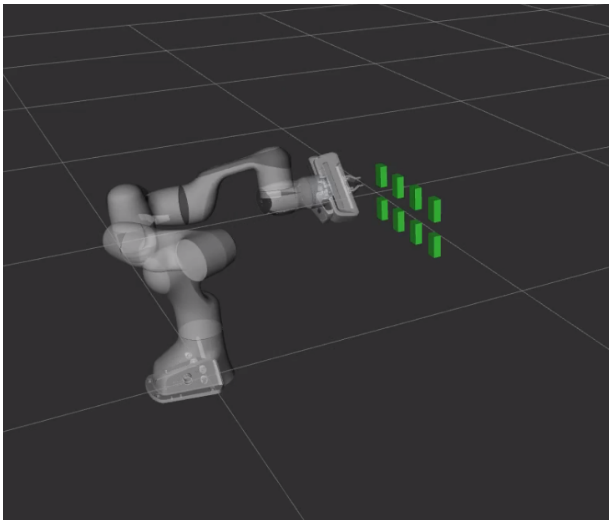
Short description.
I have a panel of 8 switches. The robot must open or close some of the switches. Which switches are opened/closed and the order in which they are switched can be configured by modifying a text file.

Detailed description.
The program will open a file that I can edit beforehand by writing into it predefined commands like this:

1: orders.txt

```
switch 2 up
switch 3 up
switch 1 up
switch 4 up
switch 1 down
switch 8 up
switch 1 up
switch 4 down
```

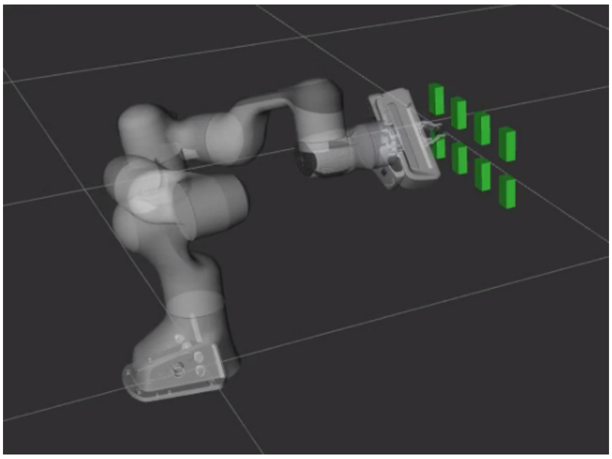
The text file is read line by line. Each command is being processed into a robot command and a set of coordinates for the robot.



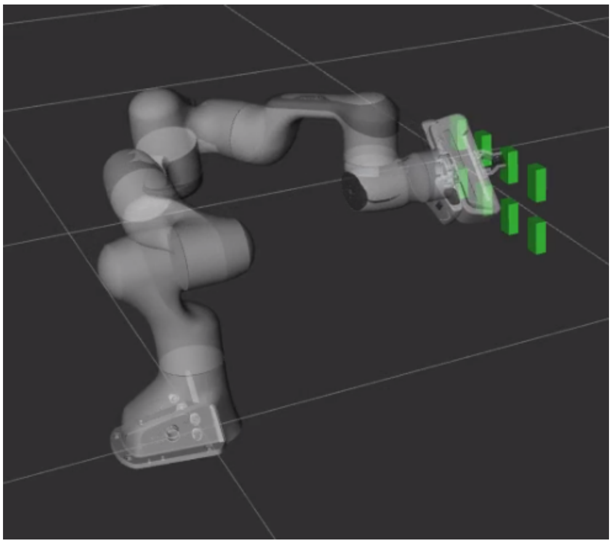
The switches are ordered as:

switch 4	switch 3	switch 2	switch 1
switch 8	switch 7	switch 6	switch 5

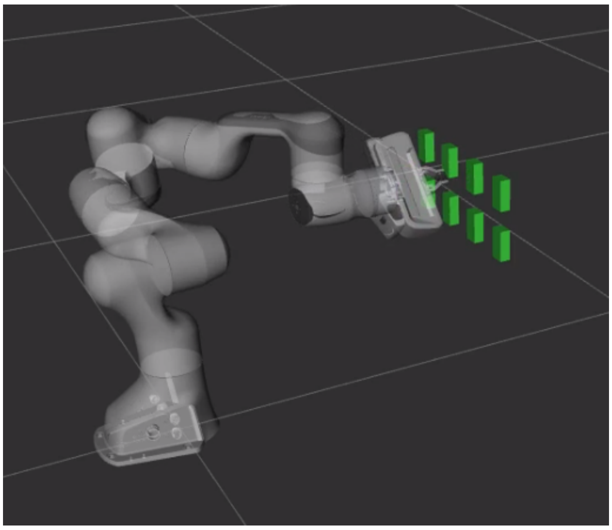
The robot moves in front of the switch with its end-efector perpendicular to the switch. The end-efector is also brought to the upper-half of the switch level if we want to open it or at the lower-half of the switch level if we want to close it. If we want to close a switch that is already closed or open one that is already opened the robot will pass to the next command.



After the robot is positioned in front of the switch it will move forward, touching the switch.



Then it will move back, ready for the next command.



And the cycle repeats for the next switch.

Performance description

It is hard to quantify the performance of the system because the number of commands is configurable.
I only control the position and orientation of the end-efector so the movement of the joints is inefficient and takes a longer time to perform the tasks.
Depending on how close the resting position is to the next switch, a switching can be performed somewhere between 6 and 30 seconds. By testing I approximate the average time it takes the robot to flip a switch as being 15 seconds.
The process can be improved by controlling the movement of the joints between the resting phase and the next switch.
The collisions are not implemented in this simulation so there is a possibility that the robot can collide with itself or the floor.

Video

panda robot simulation of flipping switches

Code

2: robot.py

```
#!/usr/bin/env python
'''
The above line is must for any python script you write.
Infact this line itself is making your script, a python file in linux.
'''
from __future__ import print_function
from six.moves import input

import sys
import copy
import rospy
import moveit_commander
import moveit_msgs.msg
import geometry_msgs.msg

try:
    from math import pi, tau, dist, fabs, cos
except:  # For Python 2 compatibility
    from math import pi, fabs, cos, sqrt

    tau = 2.0 * pi

    def dist(p, q):
        return sqrt(sum((p_i - q_i) ** 2.0 for p_i, q_i in zip(p, q)))


from std_msgs.msg import String
from moveit_commander.conversions import pose_to_list

moveit_commander.roscpp_initialize(sys.argv)
rospy.init_node("move_group_python_interface", anonymous=True)

robot = moveit_commander.RobotCommander()
```

```
scene = moveit_commander.PlanningSceneInterface()
move_group = moveit_commander.MoveGroupCommander("panda_arm")
group = moveit_commander.MoveGroupCommander("panda_hand")

display_trajectory_publisher = rospy.Publisher(
    "/move_group/display_planned_path",
    moveit_msgs.msg.DisplayTrajectory,
    queue_size=20,
)

coordinates = [[0.6,-0.15,0.4],[0.6,-0.05,0.4],[0.6,0.05,0.4],[0.6,0.15,0.4],
               [0.6,-0.15,0.3],[0.6,-0.05,0.3],[0.6,0.05,0.3],[0.6,0.15,0.3]]
activated = [False,False,False,False,False,False,False,False]
robot_movement = 0
end_position = [0,0,0]

switch1_pose = geometry_msgs.msg.PoseStamped()
switch1_pose.header.frame_id = "world"
switch1_pose.pose.position.x = coordinates[0][0]
switch1_pose.pose.position.y = coordinates[0][1]
switch1_pose.pose.position.z = coordinates[0][2]
switch1_name = "switch1"
scene.add_box(switch1_name, switch1_pose, size=(0.02, 0.03, 0.06))

switch2_pose = geometry_msgs.msg.PoseStamped()
switch2_pose.header.frame_id = "world"
switch2_pose.pose.position.x = coordinates[1][0]
switch2_pose.pose.position.y = coordinates[1][1]
switch2_pose.pose.position.z = coordinates[1][2]
switch2_name = "switch2"
scene.add_box(switch2_name, switch2_pose, size=(0.02, 0.03, 0.06))

switch3_pose = geometry_msgs.msg.PoseStamped()
switch3_pose.header.frame_id = "world"
switch3_pose.pose.position.x = coordinates[2][0]
switch3_pose.pose.position.y = coordinates[2][1]
switch3_pose.pose.position.z = coordinates[2][2]
switch3_name = "switch3"
scene.add_box(switch3_name, switch3_pose, size=(0.02, 0.03, 0.06))

switch4_pose = geometry_msgs.msg.PoseStamped()
switch4_pose.header.frame_id = "world"
switch4_pose.pose.position.x = coordinates[3][0]
switch4_pose.pose.position.y = coordinates[3][1]
switch4_pose.pose.position.z = coordinates[3][2]
switch4_name = "switch4"
scene.add_box(switch4_name, switch4_pose, size=(0.02, 0.03, 0.06))

switch5_pose = geometry_msgs.msg.PoseStamped()
switch5_pose.header.frame_id = "world"
switch5_pose.pose.position.x = coordinates[4][0]
switch5_pose.pose.position.y = coordinates[4][1]
switch5_pose.pose.position.z = coordinates[4][2]
switch5_name = "switch5"
scene.add_box(switch5_name, switch5_pose, size=(0.02, 0.03, 0.06))

switch6_pose = geometry_msgs.msg.PoseStamped()
switch6_pose.header.frame_id = "world"
switch6_pose.pose.position.x = coordinates[5][0]
switch6_pose.pose.position.y = coordinates[5][1]
switch6_pose.pose.position.z = coordinates[5][2]
switch6_name = "switch6"
scene.add_box(switch6_name, switch6_pose, size=(0.02, 0.03, 0.06))

switch7_pose = geometry_msgs.msg.PoseStamped()
switch7_pose.header.frame_id = "world"
switch7_pose.pose.position.x = coordinates[6][0]
switch7_pose.pose.position.y = coordinates[6][1]
switch7_pose.pose.position.z = coordinates[6][2]
switch7_name = "switch7"
scene.add_box(switch7_name, switch7_pose, size=(0.02, 0.03, 0.06))

switch8_pose = geometry_msgs.msg.PoseStamped()
```

```
switch8_pose.header.frame_id = "world"
switch8_pose.pose.position.x = coordinates[7][0]
switch8_pose.pose.position.y = coordinates[7][1]
switch8_pose.pose.position.z = coordinates[7][2]
switch8_name = "switch8"
scene.add_box(switch8_name, switch8_pose, size=(0.02, 0.03, 0.06))

with open('src/panda_moveit_config/scripts/orders.txt') as f:
    for line in f:
        commands = line.split()
        if commands[0] == 'switch':
            if commands[1] == '1':
                if (commands[2] == 'up') and (not activated[0]):
                    activated[0] = not activated[0]
                    end_position = [coordinates[0][0],coordinates[0][1],coordinates[0][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[0]):
                    activated[0] = not activated[0]
                    end_position = [coordinates[0][0],coordinates[0][1],coordinates[0][2]-0.01]
                    robot_movement = 1
            elif commands[1] == '2':
                if (commands[2] == 'up') and (not activated[1]):
                    activated[1] = not activated[1]
                    end_position = [coordinates[1][0],coordinates[1][1],coordinates[1][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[1]):
                    activated[1] = not activated[1]
                    end_position = [coordinates[1][0],coordinates[1][1],coordinates[1][2]-0.01]
                    robot_movement = 1
            elif commands[1] == '3':
                if (commands[2] == 'up') and (not activated[2]):
                    activated[2] = not activated[2]
                    end_position = [coordinates[2][0],coordinates[2][1],coordinates[2][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[2]):
                    activated[2] = not activated[2]
                    end_position = [coordinates[2][0],coordinates[2][1],coordinates[2][2]-0.01]
                    robot_movement = 1
            elif commands[1] == '4':
                if (commands[2] == 'up') and (not activated[3]):
                    activated[3] = not activated[3]
                    end_position = [coordinates[3][0],coordinates[3][1],coordinates[3][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[3]):
                    activated[3] = not activated[3]
                    end_position = [coordinates[3][0],coordinates[3][1],coordinates[3][2]-0.01]
                    robot_movement = 1
            elif commands[1] == '5':
                if (commands[2] == 'up') and (not activated[4]):
                    activated[4] = not activated[4]
                    end_position = [coordinates[4][0],coordinates[4][1],coordinates[4][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[4]):
                    activated[4] = not activated[4]
                    end_position = [coordinates[4][0],coordinates[4][1],coordinates[4][2]-0.01]
                    robot_movement = 1
            elif commands[1] == '6':
                if (commands[2] == 'up') and (not activated[5]):
                    activated[5] = not activated[5]
                    end_position = [coordinates[5][0],coordinates[5][1],coordinates[5][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[5]):
                    activated[5] = not activated[5]
                    end_position = [coordinates[5][0],coordinates[5][1],coordinates[5][2]-0.01]
                    robot_movement = 1
            elif commands[1] == '7':
                if (commands[2] == 'up') and (not activated[6]):
                    activated[6] = not activated[6]
                    end_position = [coordinates[6][0],coordinates[6][1],coordinates[6][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[6]):
                    activated[6] = not activated[6]
                    end_position = [coordinates[6][0],coordinates[6][1],coordinates[6][2]-0.01]
                    robot_movement = 1
            elif commands[1] == '8':
                if (commands[2] == 'up') and (not activated[7]):
                    activated[7] = not activated[7]
                    end_position = [coordinates[7][0],coordinates[7][1],coordinates[7][2]+0.01]
                    robot_movement = 1
                elif (commands[2] == 'down') and (activated[7]):
                    activated[7] = not activated[7]
                    end_position = [coordinates[7][0],coordinates[7][1],coordinates[7][2]-0.01]
                    robot_movement = 1
        elif commands[0] == 'socket':
            if commands[1] == 'plug':
                if commands[2] == 'in':
                    rospy.loginfo("%s",commands)
                    robot_movement = 2
                elif commands[2] == 'out':
                    rospy.loginfo("%s",commands)
                    robot_movement = 2
        if (robot_movement):
            if robot_movement == 1:
                joint_goal = group.get_current_joint_values()
                joint_goal[0] = 0.00
                joint_goal[1] = 0.00
                group.go(joint_goal, wait=True)
                group.stop()
                pose_goal = geometry_msgs.msg.Pose()
                pose_goal.position.x = end_position[0]-0.2
                pose_goal.position.y = end_position[1]
                pose_goal.position.z = end_position[2]
                pose_goal.orientation.x = 0.5
                pose_goal.orientation.y = -0.5
                pose_goal.orientation.z = 0.5
                pose_goal.orientation.w = -0.5
                move_group.set_pose_target(pose_goal)
                success = move_group.go(wait=True)
                move_group.stop()
                move_group.clear_pose_targets()
                waypoints = []
                pose_goal = move_group.get_current_pose().pose
                pose_goal.position.x += 0.2
                waypoints.append(copy.deepcopy(pose_goal))
                (plan, fraction) = move_group.compute_cartesian_path(
                    waypoints, 0.001, 0.0  # waypoints to follow  # eef_step
                )
                display_trajectory = moveit_msgs.msg.DisplayTrajectory()
                display_trajectory.trajectory_start = robot.get_current_state()
                display_trajectory.trajectory.append(plan)
                move_group.execute(plan, wait=True)
                move_group.stop()
                move_group.clear_pose_targets()
                waypoints = []
                pose_goal = move_group.get_current_pose().pose
                pose_goal.position.x -= 0.05
                waypoints.append(copy.deepcopy(pose_goal))
                (plan, fraction) = move_group.compute_cartesian_path(
                    waypoints, 0.001, 0.0  # waypoints to follow  # eef_step
                )
                display_trajectory = moveit_msgs.msg.DisplayTrajectory()
                display_trajectory.trajectory_start = robot.get_current_state()
                display_trajectory.trajectory.append(plan)
                move_group.execute(plan, wait=True)
                move_group.stop()
                move_group.clear_pose_targets()
            robot_movement = 0

scene.remove_world_object(switch1_name)
scene.remove_world_object(switch2_name)
scene.remove_world_object(switch3_name)
scene.remove_world_object(switch4_name)
scene.remove_world_object(switch5_name)
scene.remove_world_object(switch6_name)
scene.remove_world_object(switch7_name)
scene.remove_world_object(switch8_name)
```