Manipulation

Objectives

Modified from CMSC 20600

Your goal in this project is to learn how to manipulate the arm through code, and then use the gripper to grab the can on the floor. If you find yourself stuck on something, email jason27146913@gmail.com or join the TA office hour.

Learning Goals

- Learn how to control the robot's arm
- Create a few poses for your turtlebot to guide imaginary traffic
- Learn how to use your gripper to grab the can

Writeup

Please add the `README.md` file as your write-up for this project. Please add pictures, Youtube videos, and/or embedded animated gifs to showcase and describe your work. Your writeup should answer to question below:

- How does each joint impact the arm's position?
- Can you anticipate how changing a joint's value will affect the XYZ value?

Gif/MP4

At the beginning or end of your writeup please include two gif/mp4 files, the first one is for TrafficBot, and the second is for grabbing the can.

Note that, unfortunately, there are no recording options directly on rviz. If you are running on Windows/MacOS with NoMachine, we recommend that you record your NoMachine's rviz window directly from your base OS since CSIL machines don't come with screen recorders.

If you are on MacOS, QuickTimePlayer would be a great out-of-the-box option for screen recording. On Windows, a few options work great such as OBS Studio, or the Windows 10 builtin Game bar. On Ubuntu, one option would be Simple Screen Recorder

Problem#1 Basics of Robot Arm Control

In this lab problem, you will

1) learn the basics of Turtlebot's arms through GUIs,

2) learn how to manipulate the arm through code.

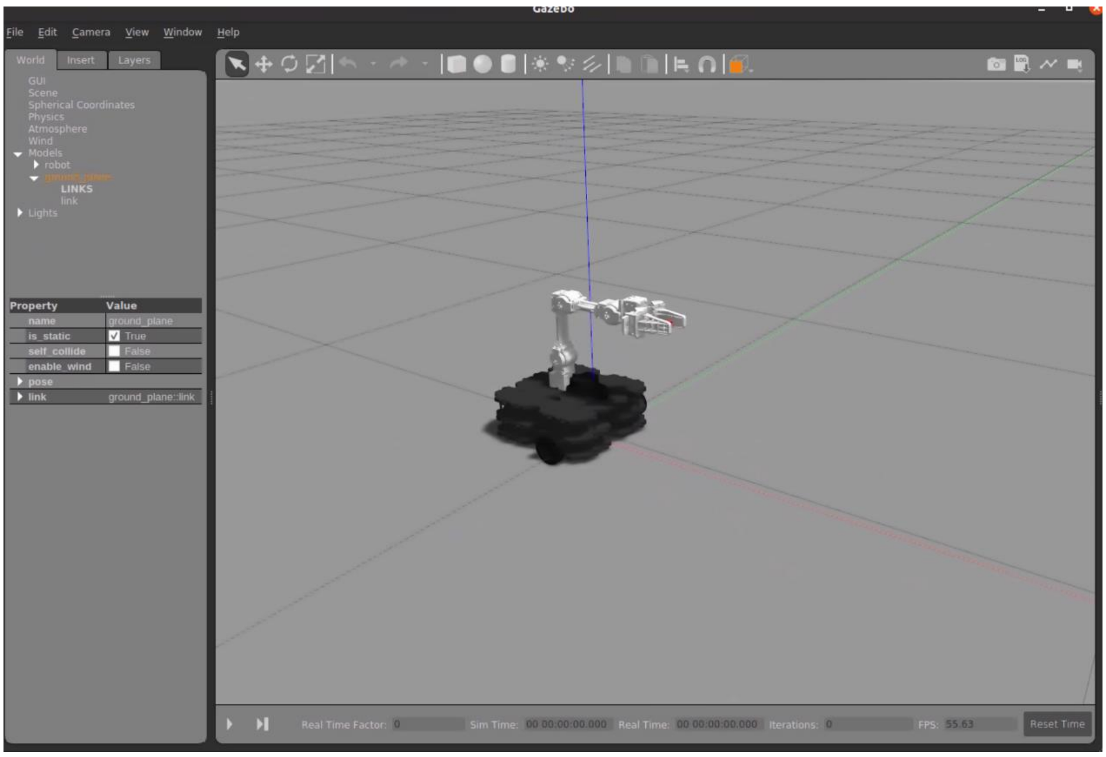
Arm Control Through GUI

To start out today's lab, we'll want to install a few ROS packages:

```
$ sudo apt install ros-noetic-ros-control* ros-noetic-control* ros-noetic-moveit* ros-noetic-gazebo-ros-control
```

Let's go over the Turtlebot3 tools that allow you to control the robot arm with a GUI. First run the following command, which starts a gazebo session a Turtlebot3 with an OpenMANIPULATOR arm.

```
$ roscore

$ roslaunch turtlebot3_manipulation_gazebo turtlebot3_manipulation_gazebo.launch
```



Property	Value
name	ground_plane
is_static	✓ True
self_collide	False
enable_wind	False
pose	
link	ground_plane::link

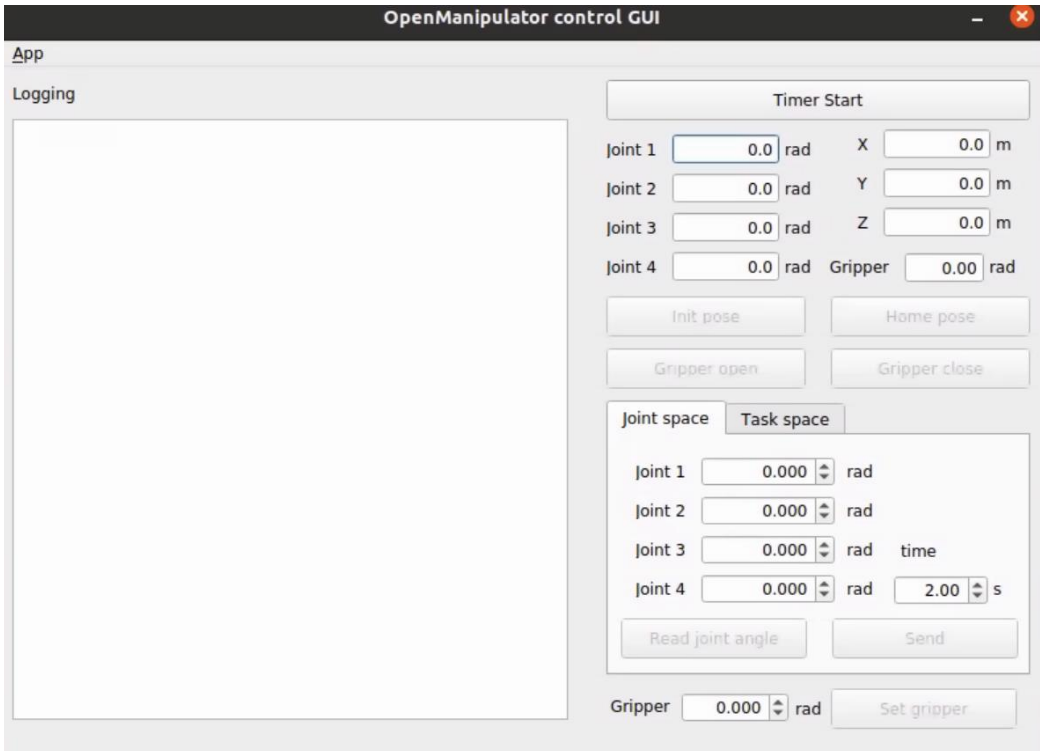
Real Time Factor: 0 Sim Time: 00 00:00:00.000 Real Time: 00 00:00:00.000 Iterations: 0 FPS: 55.63

Next, you need MoveIt to manipulate the arm. Launch the following in a new terminal. Once this node is up and running, start your gazebo simulator by pressing the ▶ button in the bottom left. You should see a message that says, **You can start planning now!** in the terminal running the `move_group`.

```
$ roslaunch turtlebot3_manipulation_moveit_config move_group.la
unch
```

Finally, you need some GUI to control the robot arm. The ROBOTIS GUI provides you with a simple window that allows control of the robot arm within gazebo. Use the following command to launch the GUI. Make sure your gazebo simulation is running and not stopped; otherwise the GUI window will not pop up.

```
$ roslaunch turtlebot3_manipulation_gui turtlebot3_manipulation
_gui.launch
```



Joint 1 0.0 rad   X 0.0 m
Joint 2 0.0 rad   Y 0.0 m
Joint 3 0.0 rad   Z 0.0 m
Joint 4 0.0 rad   Gripper 0.00 rad

Joint 1 0.000 rad
Joint 2 0.000 rad
Joint 3 0.000 rad   time
Joint 4 0.000 rad   2.00 s

Gripper 0.000 rad

The GUI provides the following information,

- the current position of each of the joints,
- the XYZ position of the red square between the robot's gripper in the simulation,
- the position of the gripper.

Let's make sure your simulation robot can move its arm. Press *Time Start* and then click on *Home Pose*. You should see your robot arm move to a new position in the simulation. Pay close attention to how the new pose changes the joint and XYZ values.

Manual movement of the arm is possible through the options in the bottom right features of the GUI. There are two options available for this purpose,

- `Task Space Control` : Sets the position of the red square between the robot's grippers to a certain XYZ.
- `Joint Space Control` : Sets the position of each single join.

The window with an *s* next to it is for selecting how fast you want the the transformation to occur. Once you are done with your manual selection, *send* your desired positions to the robot and watch the arm move.

Arm Control Through Code with the MoveIt Package

The MoveIt ROS package provides a simple, yet powerful interface to control the OpenMANIPULATOR arm on our Turtlebot3. This section outlines the general interface. There is no need to implement anything in the section. Just make sure that you understand the basics of the interface.

First, we setup the interface for controlling the gripper and the arm in the object initialization.

```
#!/usr/bin/env python3
import rospy
# import the moveit_commander, which allows us to control the arms
import moveit_commander
class Robot(object):
  def __init__(self):
      # initialize this node
      rospy.init_node('turtlebot3_dance')
      # the interface to the group of joints making up the turtlebot3
      # openmanipulator arm
      self.move_group_arm = moveit_commander.MoveGroupCommander("arm")
      # the interface to the group of joints making up the turtlebot3
      # openmanipulator gripper
      self.move_group_gripper = moveit_commander.MoveGroupCommander("gripper")
```

With this starter code, moving the arm or the gripper using joint angles is fairly simple. Let's try moving the arm,

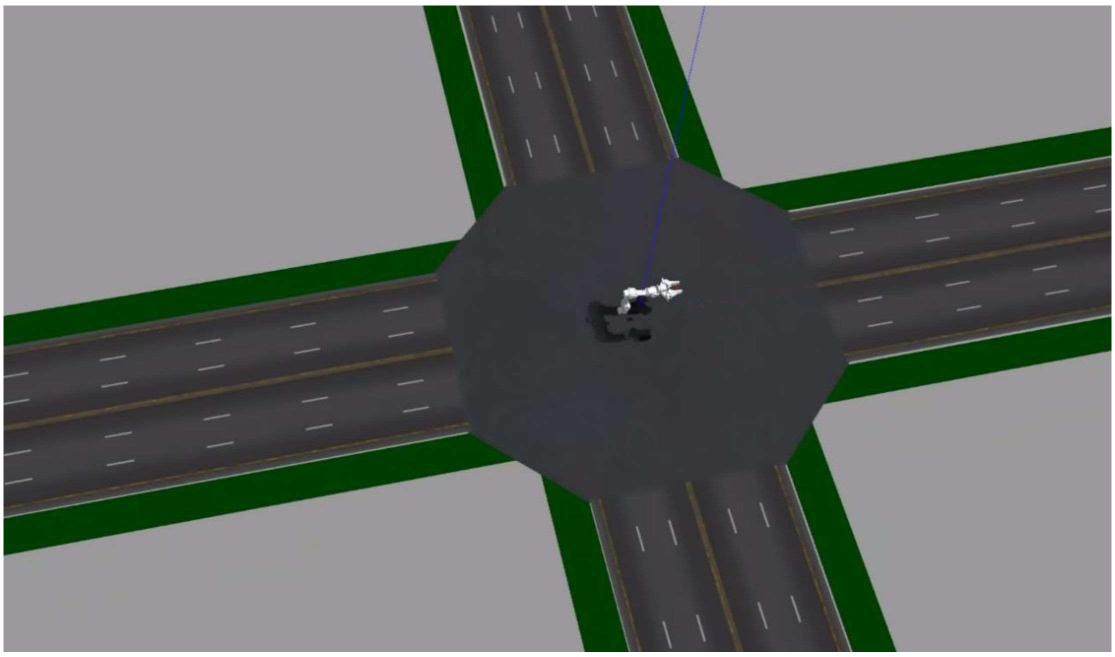
```python
  def run(self):
      ...
      ...
      # We can use the following function to move the arm
      # self.move_group_arm.go(arm_joint_goal, wait=True)
      # arm_joint_goal is a list of 4 radian values, 1 for each joint
      # wait=True ensures that the movement is synchronous
      # Let's move the arm based on what we have learned

      # First determine at what angle each joint should be.
      # You can use the GUI to find appropriate values based on your need.
      arm_joint_goal = [0.0,
                    math.radians(5.0),
                    math.radians(10.0),
                    math.radians(-20.0)]

      # Move the arm
      self.move_group_arm.go(arm_joint_goal, wait=True)

      # The above should finish once the arm has fully moved.
      # However, to prevent any residual movement,we call the following as well.

      self.move_group_arm.stop()
```

Moving the gripper is very similar.

```
        # We can use the following function to move the gripper
        # self.move_group_gripper.go(gripper_joint_goal, wait=True)


        # gripper_joint_goal is a list of 2 values in meters, 1 for the left gripper and one for the right
        # wait=True ensures that the movement is synchronous


        # Let's move the gripper based on what we have learned


        # First determine what how far the grippers should be from the base position.
        # You can use the GUI to find appropriate values based on your need.
        gripper_joint_goal = [0.009,0.0009]


        # Move the gripper
        self.move_group_gripper.go(gripper_joint_goal, wait=True)


        # The above should finish once the arm has fully moved.
        # However, to prevent any residual movement,we call the following as well.
        self.move_group_gripper.stop()
```

Problem#2 TrafficBot

In this problem, we will use the arm of the Turtlebot to guide imaginary traffic. You should not use the robot's wheels for this exercise.



Your Goal

Your goal in this problem will be to have the robot arm move in a distinct manner based on the direction it receives via the `traffic_status` topic. Here is a DEMO GIF.

Running Your Code

First terminal: run `roscore`.

```
$ roscore
```

Second terminal: run your Gazebo simulator with the intersection environment provided in the attached code.

```
$ roslaunch lab3_kinematics turtlebot3_intersection.launch
```

Third terminal: Launch the MoveIt `move_group.`

```
$ roslaunch turtlebot3_manipulation_moveit_config move_group.la
unch
```

Forth terminal: Run the python node that moves your robot arm.

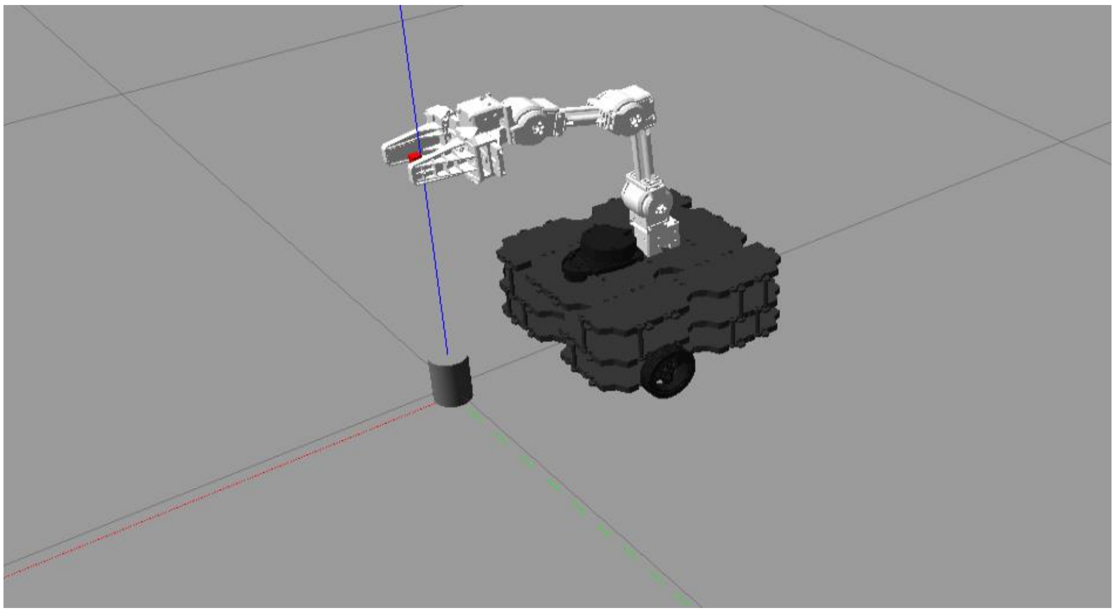
```
$ rosrun lab3_kinematics move.py
```

Problem#3 Can Grabbing

In this problem, we will use the gripper of the Turtlebot to grab the can on the floor. You should not use the robot's wheels for this problem.

The position of the can is **(0, 0, 0.3)**, height and radius are **(0.6, 0.2)**

The position of turtleBot is **(-0.2, 0, 0)**, initial values of joint and gripper are set to 0.



Your Goal

Your goal in this problem will be to have the robot arm and gripper moving, and then pick up the can on the floor. Here is a DEMO GIF.

Running Your Code

First terminal: run `roscore`.

```
$ roscore
```

Second terminal: run your Gazebo simulator with the intersection environment provided in the attached code

```
$ roslaunch lab3_kinematics manipulation.launch
```

Third terminal: Launch the MoveIt `move_group.`

```
$ roslaunch turtlebot3_manipulation_moveit_config move_group.la
unch
```

Forth terminal: Run the python node that moves your gripper to grab th can.

```
$ rosrun lab3_kinematics manipulation.py
```

Grading

The Manipulation Project will be graded as follows:

- 20% Writeup
  - 10% First Question
  - 10% Second Question
- 80% MP4/GIF
  - 40% TrafficBot
  - 40% Can Grabbing

Submission Deadline and Method

Language : python3

Packages : If you use any package, please mention it in README.md

Deadline : Thu 27 Apr 2023 13:00 (UTC+8)

Delay policy : One Day -> points * 0.7
Two Days -> points * 0.5
After Two Days -> 0 point

Everyone has a chance for a one-day late submission this semester!!!

Submission Method:

Upload a zip file to CGU Teams with the format :

```
m1234567_lab3.zip
    + m1234567_lab3
        - scripts
            + move.py
            + traffic.py
            + manipulation.py
        - msg
            + Traffic.msg
        + README.md
        + traffic_bot.mp4/gif
        + can_grabbing.mp4/gif
        + other files in the lab3_kinematics folder
```

**Do not include other files other than those mentioned above*

All file names should be **only zip files (No rar and 7zip)*

*****Incompatible format will not be graded.***

Notice : Zero points for plagiarism!!!
(either from the internet or copy from classmate)