Pick-and-Place

Assignment 7

Dec 5

In this assignment, you will implement a pick-and-place process to grasp an object and drop it in a bowl. You can choose any controllers and planners from the previous assignments.

1 Starting Code

1.1 Gripper

- Download the plugins
- Add this package to your catkin workspace
- catkin buikld `roboticsgroup_upatras_gazebo_plugins` and re-source your catkin workspace.

1.2 Kortex

Pull the code from `ros_kortex`

- for kinematic control, run `roslaunch kortex_gazebo gen3_kin_with_gripper.launch`
- for dynamic control, run `roslaunch kortex_gazebo gen3_dyn_with_gripper.launch`

Once you run this, you should see gen3 with a gripper attached. You should also see a small object and a bowl on the ground. Your job is to pick up this object and place it in the bowl.

2 Steps

2.1 Gripper

The action server for the gripper is provided by ROS. You only need to write an action client that opens and closes the gripper.

The ROS action is called `control_msgs::GripperCommand`. The action has two fields: `position` and `max_effort`. The range of `position` is between 0.0 (completely open) and 0.8 (completely closed). You need to define an open position and a close position for the gripper that fits your problem. You can set the `max_effort` to 1 to start.

Here is an example of setting your goal state:

```
control_msgs::GripperCommandGoal squeeze ;
squeeze.command.position = 0.0 ;
squeeze.command.max_effort = 1.0 ;
```

2.2 Pick-and-Place

Implement a node that can perform the pick-and-place sequence.

- go to a default position
- reach the pre-grasp position
- grasp the object
- move the object to release position
- release the object

Here are some *hopefully* useful tips:

- It is highly recommanded that you start by making a copy of your A6 and modifying it from there. Your `pick-and-place` can be a node that reads a yaml file and contains various action clients to handle different actions.
- You should have a yaml file that includes the actions and targets of the sequence. During development, you can start by testing individual process (e.g., have 1 action in your yaml file at a time), before putting them together.
- You can also observe that `rostopic echo /gen3/joint_states` gives you 8 elements, where the last element is the gripper state. Although we are not using using this gripper state direction, make sure you subscriber callback functions read the joint feedback correctly

2.3 README

Update your README.md. You can add it on-top of your A6. Your README should explain your pick-and-place process. In particular,

- What is the opening positions and closing positions of your gripper?
- What is the target of your end-effector?
- What is the result? Does it work all the time?

3 Evaluation

3.1 How to submit

Push your change to `robotic-coursework-f2022`. Do not create a new repository.

3.2 Test

- We will test your implementations by running
  ```
  > roslaunch highlevel_controller a7.launch
  ```
  Your program should start the pick-and-place process until it finishes. You should include the planner, the controller, gazebo, and all necessary components in your launch file.
- If your program requires additional instructions, please specify your instructions in README.md in your Gitlab repository.

3.3 Marking Scheme (15 points total)

(2 points) README with instructions and information
(4 points) Implement a pick-and-place process
(3 points) End-effector reaches the pre-grasp position with the gripper open
(3 points) Object is grasped (at least 2 seconds)
(3 points) Object is inside of the bowl