Assignment 6
Stress Buster
Nov 21

In this assignment, you will implement a sequence of point-to-point motion to knock over a set of objects. You can choose any controllers and planners from the previous assignments. You are recommanded to try ROS actions.

1 Steps

1.1 Create your own world file

- Open Gazebo and add a minimum of 3 objects. Save it as a world file (e.g., mine is a6.world).
- Please your world file in a ROS package (e.g., mine is under `highlevel_controller/worlds`). Once this is done, make sure the you export the gazebo environment variables.

  `export GAZEBO_RESOURCE_PATH=$GAZEBO_RESOURCE_PATH:[path to highlevel_controller]`
- Pull the most recent `kortex_gazebo`. Create a launch file `highlevel_controller/launch/a6.launch` and launch Gazebo with your world file. After roslaunch, you should see the arm with the objects you created.

  For kinematic control:

  ```
  <include file="$(find kortex_gazebo)/launch/gen3.launch">
      <arg name="world_name" value="worlds/a6.world"/>
  </include>
  ```

  For dynamic control:

  ```
  <include file="$(find kortex_gazebo)/launch/gen3_dynamics.launch">
      <arg name="world_name" value="worlds/a6.world"/>
  </include>
  ```

1.2 Action Client

- Create a YAML file `highlevel_controller/config/a6.yaml` that contains a list of target positions. You can structure your yaml file that way you like.
- Your action client should read the target positions from this yaml file.
- The action client sends the target positions to the server (one at a time)
- You can pull the most recent `hello_action`, copy `hello_action_client`, and modify it to fit your problem.

1.3 Action Server

- Implement an action server that can handle a sequence of requests
- Your action server can be part of your planner. You can add action callback on top of service callback in your existing planner or create a separate planner.
- You can pull the most recent `hello_action`, copy `hello_action_server`, and modify it the way you want.

1.4 Readme

Write a `README.md` in your main repository (under `robotic-coursework-f2022`) with some descriptions and instructions for A6. e.g.,

- What is your choice of planner and controller?
- How do you handle this problem? Did you use ROS actions?
- Where is the world file?
- How to run A6?
- Where and how to modify your target positions?
- Any limitations?

2 Evaluation

2.1 Test

Push your change to `robotic-coursework-f2022`. We will test your implementations as follow:

1. Open a terminal and run
   `> roslaunch highlevel_controller a6.launch`
   Once Gazebo starts running, your robot should move toward the end-effector position specified in `a6.yaml`.
2. We will change the values in this yaml file and restart.

2.2 Marking Scheme (10 points total)

(3 points) Correctly generate a world file and visible after launch
(2 points) Planner is able to generate a sequence of motion
(2 points) All objects fall
(3 points) Planner works after modifying the values in the yaml file