HW3 - Robot Autonomy Plus

Professor: Oliver Kroemer, TA's: Steven Lee, Ami Sawhney

Due: 1/19/21 at 11:59pm Eastern Daylight Time (GMT-4))

1 Dynamic Movement Primitives (DMP)

For this portion of the homework, you will be implementing DMPs to generate smooth trajectories using a joint space trajectory that we have provided you with. You may find the information presented in Lecture 4 (Trajectory Generation) helpful in your implementation.

DMP.py contains a **DMP_trajectory_generator** class, where we provided code in the function **learn_weights_from_raw_trajectory()**, which you will use to learn the weights, i.e. shape parameters, to fit a DMP trajectory to the raw trajectory provided. Then you will implement the function **generate_dmp_trajectory()**, where you will generate a new DMP trajectory using the learned weights. The comments in the code provide more details on the implementation.

First, call the function to learn weights for the provided trajectory via linear regression, and then use your implemented function to generate a DMP trajectory using the learned weights. Feel free to experiment with the number of basis functions used to generate the trajectory in addition to the goal joint positions.

What to submit: Please submit your code, a plot of your generated DMP trajectory, and provide a 2-3 sentence summary of your key learnings from this exercise and how you think DMP trajectories might be useful for robot tasks.

2 Rapidly Exploring Random Trees (RRT)

2.1 Implementing RRT

In this section you will be implementing the **Rapidly Exploring Random Trees (RRT)** algorithm for the LoCoBot. Similar to the Probabilistic Roadmap from homework 2, your planner will need to provide a feasible collision free path from an initial joint configuration to a target configuration. The motion planner should plan in joint space for joints 1 through 5. We are not concerned with moving the gripper fingers in this assignment, so PRM planning for joint 6 and joint 7 is not necessary.

Table 1: Environment Bounding Box Information

Name	Origin (m): (x, y, z)	Orientation (rad): (r, p, y)	Dimensions (m): (d_x, d_y, d_z)
Cuboid0	(0.275, -0.15, 0)	(0,0,0)	(0.1,0.1,1.05)
Cuboid1	(-0.1, 0.0, 0.675)	(0,0,0)	(0.45,0.15,0.1)
Cuboid2	(-0.275, 0.0, 0.0)	(0,0,0)	(0.1,1.0,1.25)
TargetCuboid	(0.5, 0, 0.05)	(0,0,0)	(0.04,0.04,0.1)
LocobotMount	(-0.03768, 0, 0.36142)	(0,0,0)	(0.12,0.26,0.5)
LocobotBase	(0.0, 0, 0.05996)	(0,0,0)	(0.35,0.3521,0.12276)

The task is to pick up a target block, where you know the position and shape of the block. The initial joint configuration is given to you in **RRTLocobot.py** as **qInit**. However, for this task **you are not given a goal configuration.** You will need to sample a suitable grasp point along the target block's surface, run RRT from the initial configuration to this sampled grasp point, and then grasp the block. It may be beneficial to sample multiple grasp points instead of just one.

You will also perform collision checking, so you can use your cuboid collision detection implementation from homework 2. The collision detection should be implemented in the same way as homework 2, so for more details please refer to that assignment. The only difference will be that the environment is different and any variables that had *"prm"* in them will now have *"rrt"*. Additionally, there are not separate file for "map generation" and "query". You will perform the generation of the tree and provide the path to the goal in **RRTLocobot.py**.

The bounding box information of the VREP scene file is provided to you in the **RRTLocobot.py** file, as well as in Table 1. As in homework 2, your planner should produce a smooth trajectory when you visualize it, so you will probably need to interpolate between points in the path.

Similar to homework 2, once you've generated a joint space path, you can visualize it using the PlotCollisionBlockPoints() method of the Locobot class.

Next, you will visualize your planned path using the Locobot in VREP. The section below describes this further.

Visualizing the Output of Your Planner in VREP using Pyrobot:

RRTLocobot.py has a command line argument of –use_pyrobot, that determines whether the visualization of your path will be in Pyrobot (using VREP) or simply plotting with Python's Matplotlib. The default argument is False.

To test your planner in VREP with the environment obstacles we provided to you, start a roscore and then run **RRTLocobot.py** with the command line argument –use_pyrobot = True (don't forget to source the pyrobot virtual

environment first). This will start the VREP simulator and load in the Locobot. Note that because of the way Pyrobot initially loads the Locobot into the simulator, the Locobot arm will initially be intersecting one of the blocks in these scene. You can ignore this during the initial loading, because once the robot arm is commanded to go to the starting joint configuration (with the command robot.arm.set_joint_positions()) it will start in the correct location. You should use the path outputted by your RRT planner and use the robot.arm.set_joint_positions() command in the collision_detection.py script to visualize your planned joint space path in VREP. You can use the command robot.gripper.close() and robot.gripper.open() to close and open the gripper respectively.

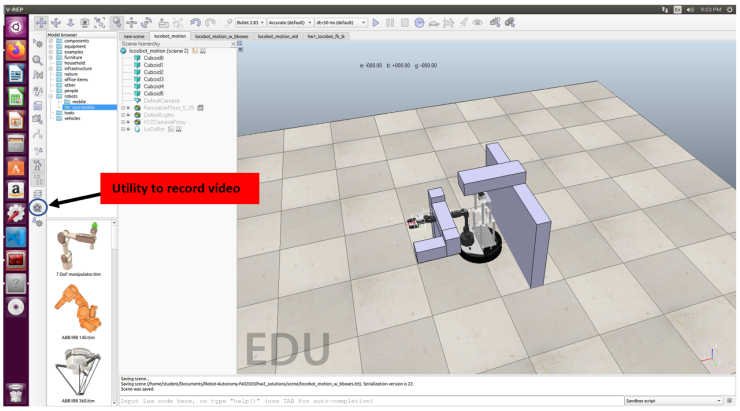
2.2 RRT Path Shortening

For this portion of the assignment, you will be implementing path shortening and applying it to the path your RRT outputs from section 2.1. You should perform the path shortening in **RRTLocobot.py** after your RRT has already given you a path. So run your RRT algorithm as you did in Section 2.1 and then perform path shortening on that output. Once you obtain the shortened path, visualize it in the same way you did in Section 2.1. Please refer to the lecture slides/recordings for details on the path shortening implementation. **Note:** Remember to perform collision detection while you are implementing this portion of the assignment.

Deliverables:

- A small, concise paragraph (2-4 sentences) describing your RRT planner.
- A plot showing joint positions for joints 1 through 5 that shows each point along the path produced by your RRT **before and after path shortening**. Your plots must include labels and units on each axis.
- A video of the LoCoBot executing the plan outputed by the PRM. **You only need to provide a video of the shortened path.** You can use a screen capturing program or the video recording capabilities within V-REP. If your video is large, please provide a link to it through Google Drive, Dropbox, or some other cloud hosting service so that we may access it later.

  Note: Shown below is the icon you can use to record a video in VREP.


Utility to record video
EDU