Lab 3: Motion Planning with a 6-DOF Manipulator

(72 points total)

All video files uploaded to github and can be accessed from this link as well : `https://drive.google.com/drive/folders/1nXUeh-inCNm0RchVSEeGO3Id-bDYjjfT?usp=sharing`

1. **(40 points)** Use an off-shelf screen capture software (e.g., `https://itsfoss.com/kazam-screen-recorder/`) to **record a video** of the trajectory. Include the video in the root of your GitHub repo as a file named `question-3.mp4`.
   Uploaded to github

2. **(10 points)** The RRT trajectory is typically jerky. Typical planners use shortcutting algorithms to make the plath smoother. **Replace the function** `ada.compute_joint_space_path` with `ada.compute_smooth_joint_space_path`. Capture the new trajectory with two videos – one showing the default isometric view, and another showing the top view. Include the videos in the root of your GitHub repo as files named `question-4-default.mp4` and `question-4-top.mp4`.
   Uploaded to github

3. **(10 points)** The goal precision ϵ of 1.0 in the previous question is too large. In order to avoid collisions, we need to improve the precision. However, this dramatically increases the time to compute a solution. To improve computation, **add a method** `_get_random_sample_near_goal` that generates a sample around the goal within a distance of 0.05 along each axis of the search space. Then, change the `build` method so that it calls `_get_random_sample_near_goal` with probability 0.2 and `_get_random_sample` with probability 0.8. Reduce ϵ to 0.2.

   Write down your observations in the PDF file. Also capture the new trajectory with two videos – one showing the default isometric view, and another showing the top view. Include the videos in the root of your GitHub repo as files named `question-5-default.mp4` and `question-5-top.mp4`.

Precision improved: Second-to-last waypoint is 0.09 vs 0.52 units from goal More waypoints: 11 vs 8 (more refined path) Minimal time increase: Only 28Goal-biased sampling effective: Kept computation reasonable

The paths look "similar" because both succeed, but Question 5's path is significantly more precise near the goal!

4. **(10 points)** Explain why it is not a good idea to call `_get_random_sample_near_goal` with probability 1.0. Also present an example where this could be problematic. Write your answer in the PDF.

Why Goal Sampling Probability 1.0 is Problematic

Main Issues

Setting goal sampling probability to 1.0 eliminates RRT's exploration capability, causing:

- **No exploration**: The tree never samples random configurations, losing RRT's ability to discover alternative paths
- **Obstacle entrapment**: Without exploration, the algorithm cannot navigate around obstacles blocking the direct path to the goal
- **Completeness loss**: Even if a solution exists, the algorithm may never find it

Example: U-Shaped Obstacle

Consider a scenario where the goal is behind a U-shaped barrier:

```
Start (S)
                 Wall

         Wall

                 Wall
          Goal (G)
```

With probability 1.0:

- Every iteration samples near the goal region
- Tree attempts to extend toward goal but is blocked by the wall

- Tree cannot grow and remains stuck near start
- **Result**: No solution found

With proper probability (0.05-0.20):

- Random sampling allows the tree to explore upward and around the obstacle
- Occasional goal-biased samples help connect to goal once a feasible path exists
- **Result**: Path found successfully

Conclusion

RRT's effectiveness relies on balancing exploration (random sampling) and exploitation (goal-biased sampling). Probability 1.0 removes exploration entirely, preventing the algorithm from discovering paths around obstacles. The recommended range is **0.05-0.20** (5-20%).

In-person lab: Once you are confident in your simulation results, you are ready to run it on the real robot. This can be done on the lab workstations with -

```
$ python adarrt.py --real
```

Refer to Piazza for more instructions on scheduling time in the lab.

1 Additional Questions

As a very rough guideline, we anticipate that for most teams, the answers to each question below will be approximately one paragraph long (or about 4-5 bullet points). However, your answers may be shorter or longer if you believe it is necessary.

1.1 Resources Consulted

Question: **(1 points)** Please describe which resources you used while working on the assignment. You do not need to cite anything directly part of the class (e.g., a lecture, the CSCI 545 course staff, or the readings from a particular lecture). Some examples of things that could be applicable to cite here are: (1) did you get help from a classmate *not* part of your lab team; (2) did you use resources like Wikipedia, StackExchange, or Google Bard in any capacity; (3) did you use someone's code (again, for someone *not* part of your lab team)? When you write your answers, explain not only the resources you used but HOW you used them. If you believe your team did not use anything worth citing, *you must still state that in your answer* to get full credit.
1) No help from classmates outside of lab team 2) Yes, used google and claude to verify our code and understand the outputs of the paths. 3) No, all code was developed by our lab team

1.2 Team Contributions

Question: **(1 points)** Please describe below the contributions for each team member to the overall lab. *Furthermore, state a (rough) percentage contribution for each member.* For example, in a team of 4, did each team member contribute roughly 25% to the overall effort for the project?
Katherine -20% , Aryan - 20%, Harshawn - 20%, Pranav - 20%, Yutao - 20%