Lab 3: Motion Planning with a 6-DOF Manipulator

Team: Gradient Descent Squad

Nov 20, 2025

1 Main Implementation

3. **(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`.

4. **(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`.

   Answer:

   Our RRT pipeline grows a tree by repeatedly sampling a configuration, finding the nearest tree node, and extending toward the sample by a fixed step if it's collision-free. Once a path is returned, we generate a trajectory with `ada.compute_joint_space_path`. In the next step, we swap this with `ada.compute_smooth_joint_space_path` (toggled via the `--smooth` flag).

   After replacing the function with `ada.compute_smooth_joint_space_path`, the robot arm follows a noticeably smoother trajectory. The results are shown in the two captured videos, included in the repository as `question-4-default.mp4` and `question-4-top.mp4`.

5. **(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`.

Answer:

We implemented the requested modifications as follows:

- Reduced the goal precision from $\epsilon = 1.0$ to $\epsilon = 0.2$.
- Added a goal-biased sampling strategy that samples configurations within ± 0.05 of the goal state with probability 0.2 (and uniform sampling with probability 0.8).

We then compared the performance of the modified planner against the baseline (before applying the changes) using 10 manual simulation runs.

- **Improved final pose accuracy.** With the reduced ϵ, the final step before reaching the goal is noticeably closer to the target configuration. In the baseline, the gripper often stopped far from the target or at an awkward orientation that would make grasping difficult. After the modifications, the final pose is consistently closer to the desired goal and more aligned, increasing the likelihood of a successful grasp.
- **More consistent trajectories.** The modified planner produces trajectories with more consistent numbers of steps and RRT nodes. In particular, the average number of RRT nodes is significantly lower than in the baseline. The standard deviations of both the number of steps and the number of RRT nodes also decreased (from 1.66 to 1.26, and from 69.8 to 8.6, respectively), indicating improved stability and less variability across runs. The detailed statistics are shown in Table 5.

	#Steps		**#RRT nodes**	
	before	**after**	**before**	**after**
Mean	4.1	6.6	87.6	26.3
Standard deviation	1.66	1.26	69.8	8.6

Table 5: Mean and standard deviation before and after the sampling strategy changes

6. **(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.

   Answer:

_get_random_sample_near_goal with probability 1.0 is not a good idea because it eliminates all exploration of the configuration space in the RRT algorithm. With 100% goal biasing, the algorithm repeatedly samples only in a small region around the goal, so it causes the tree to extend in the direction of the goal from existing nodes. This can lead to the tree getting stuck if direct paths to the goal are blocked. As a result, the planner may fail to find feasible paths, take excessively long, or never converge.

2 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.

2.1 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.
Answer:

- Phillip Huang: I asked ChatGPT and Grok to teach me how to set up the docker image, docker environment, and how to show the animation of the trajectory. Asked Grok to double check if my code was correct.
- Zeyuan Wu: I consulted ChatGPT on the docker image and used it for debugging. Also used it to understand the code written by teammates.
- CheWei Hsu: I ask claude for screen recording issues, kazcam only recorded a black image with my mouse. End up using default ubuntu screen recording then convert to .mp4 file using ffmpeg.
- Andrew Chang: I consulted Claude to help implement the extend sample function and to set up the Docker container for running the lab environment. I also asked ChatGPT to convert the Question 4 top-view recording into MP4 format.

- Jui-Hung Chen: I consulted ChatGPT on how to sample around the requested range of 0.05 while still confining the samples to the joint limits, and I adapted the example code to the `_get_random_sample_near_goal` function. In addition, I used ChatGPT to proofread my written answers to Questions 4 and 5.

2.2 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?
Answer:

- Phillip Huang: contribution (20%). I implemented the `_get_random_sample` function and recorded the Q4 default view video. I also answered Q6.
- Zeyuan Wu: contribution (20%). I implemented the build and path-tracing function. Recorded q5 default view video.
- CheWei Hsu: contribution (20%). I implemented the `_get_nearest_neighbor` function and ran the simulation script to record the video for question 3.
- Andrew Chang: contribution (20%). I implemented the extend sample function and recorded the Q4 top-view video.
- Jui-Hung Chen: contribution (20%). I replaced the original planner with the smoothed version and implemented the required changes for both Q4 and Q5. Additionally, I recorded the Q5 top-view video and completed the written answers for Q4 and Q5 in the report.