CSCI 445 Labs 11, 12 — Final Project

Prof. Culbertson, University of Southern California Fall 2021

In this project, you will combine the work you did in several labs to control operations to transport an object through a maze, like a robot might do in a factory setting. From its initial location, your Create2 will carry an object through a maze to a robotic arm, which will pick up the object and place it on a shelf. You will write code for the Create2 and the robotic arm.

This project will have two parts. The Create2 must localize itself in the maze, plan a path to the goal, and follow that path. The robotic arm must reach the object, grasp it, and move it to the shelf.

Note: The simulation file and sample code provided include the buttons that you used to control the robot in Lab 8. However, you should not use predefined actions or manually control your robot in your final implementation. We have included them so that you can use them during testing and integration if you'd like.

1 Challenges

There are several challenges that need to be solved.

1. Your robot's onboard odometry tends to drift over time, making it impossible to follow long paths (as we have seen in previous labs). In open space, this doesn't matter very much. However, here you will be working in a cluttered environment, like you might find in a factory or a warehouse, so you will need to do additional localization using measurements of that environment to avoid hitting any walls.

2. You will not know your start and goal locations until the final demonstration of your project. You should be able to recompute paths and joint angles as needed (i.e., no paths should be hard-coded).

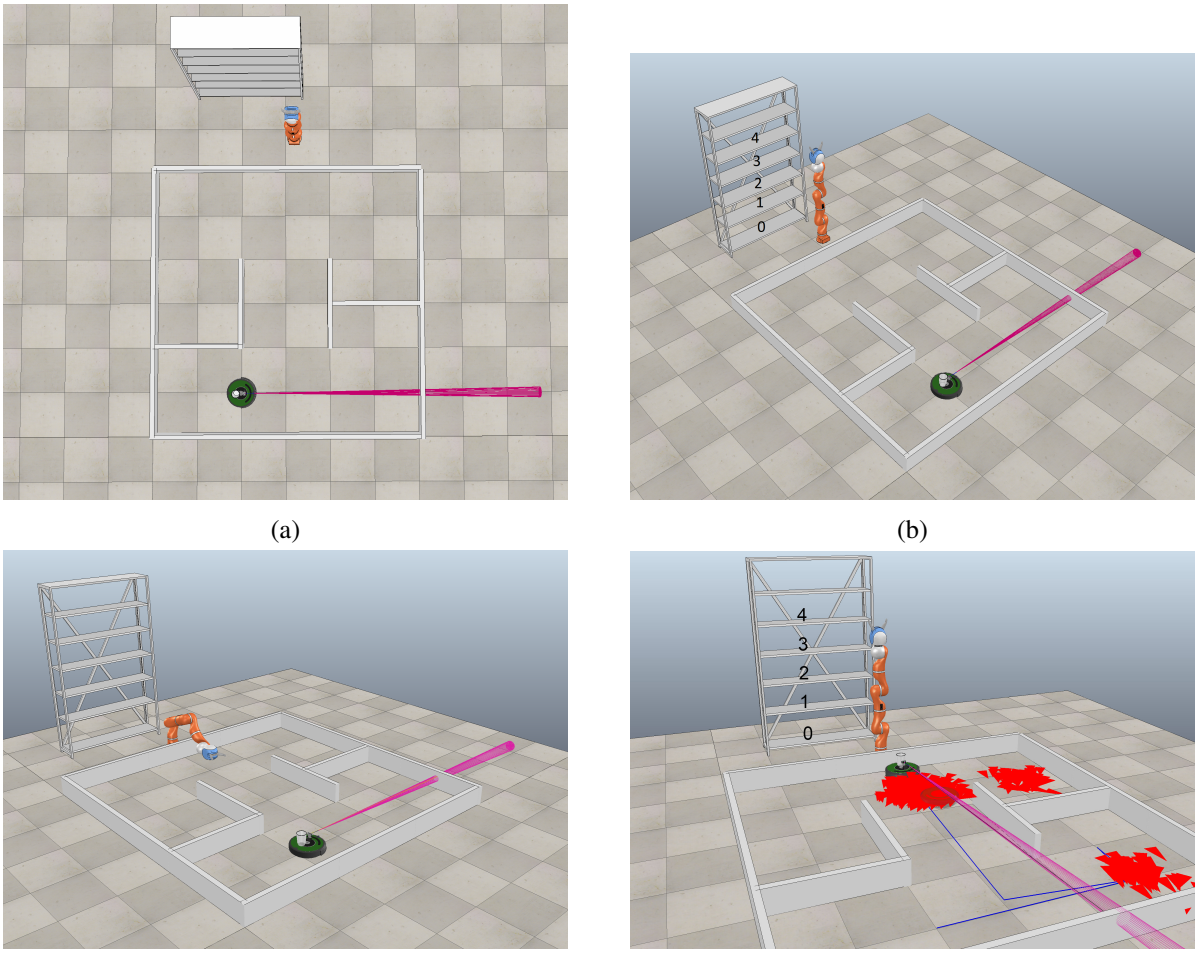
3. You should aim to complete the task as quickly and accurately as possible. You will have a limited amount of time, and you will be penalized if you do not complete any part of the task within the time limit or if you hit any walls.

2 Example

In the example setup shown, the robot begins on the near side of the maze carrying a glass. It needs to move to the far side of the maze, where the robot arm is located. Then the robot arm must reach the position of the glass, grip it, and place it on shelf 2. This is the example that you should work with to test your implementation, but the final demonstration will have different positions for the robot and the arm.

3 Hints

- Reuse as much code as possible from previous labs (PID controller, odometry, particle filter, RRT).
- You can make changes to the simulation. For example, you might have the robot start near the arm so you can work on using the arm without needing to navigate the robot to it every time.
- You can query the simulator to obtain the ground-truth pose via `self.create.sim_get_position()` so that you can implement the motion planning module independent from the localization of the particle filter.



(a) (b)

(c) (d)

Figure 1: Example setup for robot and arm from different perspectives. The robot begins on the near side of the maze, as shown in (a) and (b). It must localize itself and plan and follow a path to a position near the arm (d). Meanwhile, the arm must reach down to the robot's position and grasp the glass, as in (c), and place it on shelf 2. Your robot should be able to reach the arm even if it is in a different location, and your arm should be able to place the glass on any reachable shelf.

- You can assume that you know the robot's starting location, and you do not need to localize the robot in the beginning. The starting location should not be hard-coded, but can be determined using `self.create.sim_get_position()`.
- There may be some shelves that the robot arm cannot reach, so you will not be able to place the cup on those shelves. Your IK calculations will return an error if you try to reach a location that is outside of your arm's workspace.
- In the configuration space image, the origin [0; 0] is in the upper left corner. However, for the robot, the origin is in the lower left corner.
- Try to split up the work and have team members work on different tasks in parallel.

4 Grading

You will need to write a report describing your algorithm, how you tackled the challenges, and including some example pictures/screenshots of your results. Your solution will be scored as follows:

- **45%: Algorithm design.** How did you integrate the modules we built throughout the semester? Did you use error checking for errors you have previously seen on the robot, such as inaccurate sonar measurements? Did you hard-code everything, or could your implementation be adapted to changing goals?
- **30%: Simulation performance.** How much of the task did you complete? Did you hit any walls? How accurately did you reach your goal location and place the object on the shelf? If you showed multiple demonstration runs, how consistently did you achieve good performance?
- **25%: Writeup quality.** Did you explain clearly what you did and why? Did you include examples of the output of your algorithms and your performance? Could someone reconstruct your solution based on your writeup?