LAB 4: PATH PLANNING

Due: Tuesday, March 7th 5:30pm

The objective of this lab is to implement and test path planning capabilities for Cozmo. The lab consists of the following parts:

A* path planning [40 points]: Implement A* path planning and validate your implementation in the provided simulated map environment. You are provided the following files:

`planning.py` – File in which you will implement A* and your Cozmo code (described in next section). Executing this file starts your Cozmo code and a visualizer.

`grid.py` – Defines the CozGrid class, which represents an 8-connected discretized grid and holds information about start, goal, and obstacle locations as well as visited locations and the current path. Also specifies the size of a grid square, in mm.

`visualizer.py` – A visualizer to help debug your algorithms. Displays start location (green), goals (blue), obstacles (dark gray), visited locations (light gray), and the current path (red arrows). The other files are set up to launch this automatically.

`autograder.py` – This autograder is provided to help you verify your A* solution. We will use the same autograder with a new map to verify your code. Run with `python3 autograder.py testcase1.json`

`map1.json` – Map definition file containing test scenario for A*.

`testcase1.json` – Autograder file corresponding to `map1.json`

`emptygrid.json` – Map file defining only start location, for use with Cozmo portion of assignment. When using Cozmo run the code with `python3 planning.py emptygrid.json`

Path planning on the robot [60 points]: Integrate your path planner with Cozmo to enable Cozmo to safely navigate to a goal using its odometry. Specifically, your goal is to enable Cozmo to navigate to face Cube 1 from a predefined direction, while also avoiding Cubes 2 and 3 if they are present. You will implement the following functionality:

1. **Update Map:** Update the map to correctly represent all cubes detected by the robot.
2. **Navigate to Goal:** Successfully navigate the robot from a known start location to Cube 1,

and stop within 4” of Cube 1.

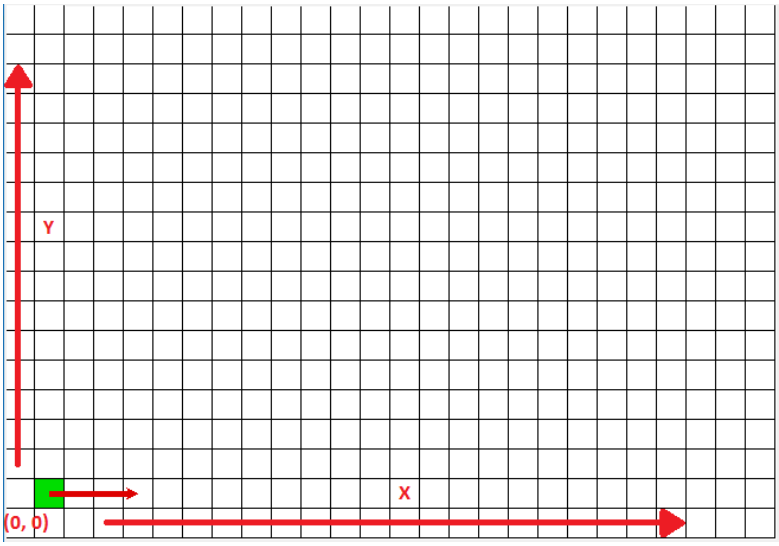
3. **Cube Orientation:** Extend your navigation code to consider which side of the cube the robot is approaching. Have the robot approach the side of the cube that looks like this:

The orientation of the cube can be found in `cube.pose.rotation.angle_z`. The robot should stop within 4” of the cube, facing toward the correct side of the cube but without touching it.

4. **Obstacle Avoidance:** Extend your code to have Cozmo successfully avoid coming into contact Cubes 2 and 3 if they are in the space.

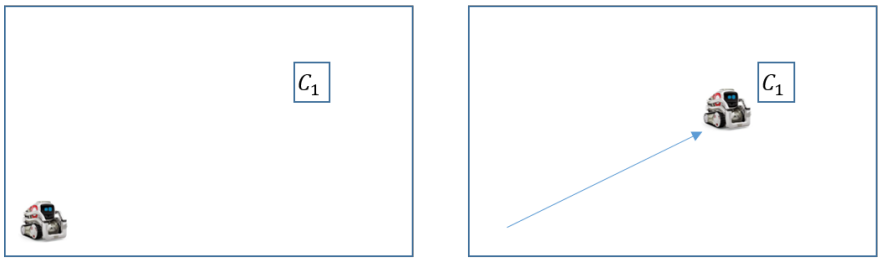
Notes:

- We will not use the ball at all in this lab.
- Points will be deducted if the robot touches any of the cubes or the walls of the arena at any time.
- Your robot will be given its own start pose, but not the pose of any cubes (you must rely on the vision system to find them).
- Cube 1 will be in the arena from the beginning. If your robot does not see Cube 1 from its start position, have it navigate to the center of the arena and turn in place until the cube is seen.
- Cubes 2 and 3 may be added to the arena at any time. Your code must support continuous detection, map updates and replanning.
- Grid coordinates are formatted as `(x, y)`, with the origin in the lower left corner, x increasing to the right, and y increasing to the top. Cozmo will start facing the +x direction, as shown

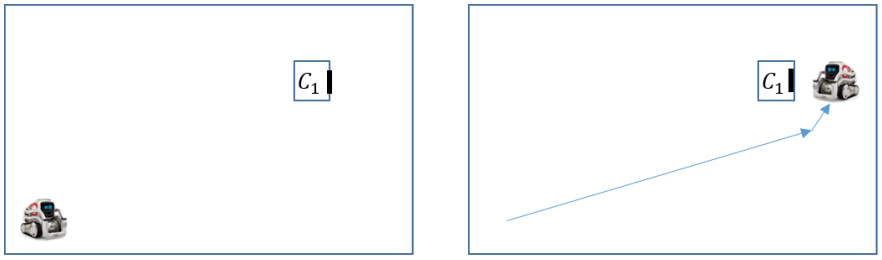


Examples of intended robot behavior:

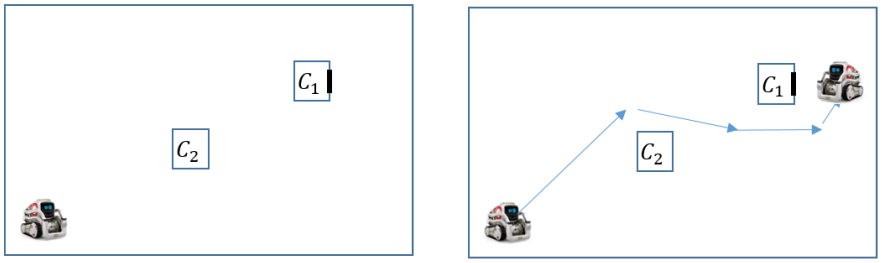
Start *End*



Robot behavior at step 2 above: ignoring cube orientation, no obstacles.

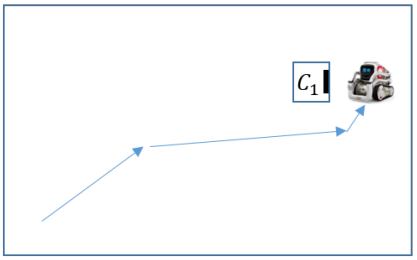


Robot behavior at step 3 above: accounting for cube orientation, no obstacles.



Robot behavior at step 4 above: accounting for cube orientation and obstacles.

If the robot does not see Cube 1 immediately, it may drive to the center of the arena. It should update its path as soon as it sees the cube (as shown on right).



See separate grading rubric for additional details on the evaluation.

You will demo your code for grading during class on the day the assignment is due.

Submission: Submit only your `planning.py` file, make sure you enter the names of both partners in a comment at the top of the file. Make sure the file remains compatible with the autograder. Only one partner should upload the file to T-Square. If you relied significantly on any external resources to complete the lab, please reference these in the submission comments.