CALIFORNIA INSTITUTE OF TECHNOLOGY

ME/CS 132b, Spring 2011

Lab #1 Due: May 5th, 2011

In this lab, you will be asked to plan and execute a path to drive the Pioneer robot towards a given goal (see Fig. 1). The robot must not hit the two obstacles in the middle.

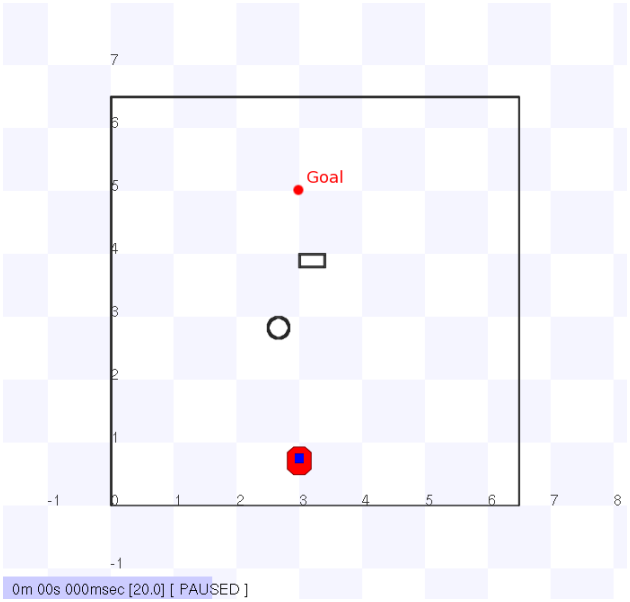

Figure 1: World map. The robot is initially facing in $+y$ direction.

Specifications

Throughout the handout, the lengths will be expressed in meters, and the angles in radians.

- The initial pose of the robot is $(x, y, \theta) = (3.0, 0.75, \pi/2)$.
- The goal is $(x, y) = (3.0, 5.0)$.
- There are two obstacles: the circular one ($r = 0.15$) is centered at $(2.65, 2.85)$; the rectangular one (0.4×0.2) has its upper-left corner placed at $(3.0, 4.0)$.

Part 1: Simulation at Home

You will be able to get started from the tutorial code `me132_tutorial_planning.cc`. It is modified from `me132_tutorial_1.cc` from last term. The only additional feature is that it also converts the laser scanner readings into an occupancy map (grid) for path planning. The occupancy map is a binary (0/1-valued) matrix that indicates whether a given location in the world is occupied by an obstacle. You will need to specify the resolution and the size of the occupancy map in the beginning of `me132_tutorial_planning.cc`.

Description of the Path Planning Algorithm

1. Take a scan of the obstacle and update the corresponding occupancy map. This part has already been implemented in `me132_tutorial_planning.cc`.

2. Dilate the obstacle by the size of the vehicle (i.e. C-space expansion). You will need to modify `SimpleOccupancyGrid::addScan()` in `occupancy_grid.cc`. Treat the robot as a disk with $r = 0.2$ m.

3. Set the vehicle speed at 0.2 m/s. Evaluate a set of N arcs of length s corresponding to different curvatures (i.e. different turn rates, when the vehicle speed is fixed). The arcs should be defined in the body-fixed frame. You should try different values for N and s, and explain your final choice. As a first cut, N should be somewhere between 3 and 50, and $s = 3.0$ m.

4. An arc is called collision free if it does not intersect with any obstacle. Among all the collision-free arcs, select the arc whose terminal position is closest to the goal. Send the corresponding turn rate and speed command to the vehicle.

5. Use the same turn rate and speed until the robot has executed the first l meter of the arc, and command a stop by setting both zero. If you chose $s = 3.0$ m previously, then you can use $l = 1.0$ m as an initial trial. Record the robot's pose during execution in a log file.

6. Go to step 1, and repeat until the vehicle is close to the goal. You should try to bring the vehicle as close to the goal as possible. Normally, the final distance to the goal should be within 0.5 m.

In the lab report:

1. [15 points] Attach the code snippet for the modified `SimpleOccupancyGrid::addScan()` that incorporates C-space expansion, and explain your implementation briefly. Attach a plot that shows the final occupancy map with C-space expansion generated at the end of execution.

2. [35 points] Attach the modified `me132_tutorial_planning.cc` and explains your implementation. Your explanation should include: (1) choice of various parameters (N, s, l etc.) chosen in the final implementation, (2) collision detection for the arcs, (3) path following and replanning.

3. [20 points] Based on the N and s you chose, plot the set of arcs at 3 time instances during the entire path planning: (1) in the beginning ($t = 0$), (2) in the middle ($t \approx T/2$), (3) at the end, immediately before the robot reaches the goal ($t = T$), where T is the total time of execution. Use different colors or line styles to indicate collision-free vs. colliding ones.

4. [15 points] Plot the trajectory $(x(t), y(t), \theta(t))$ of the robot. The trajectory should be a 2D scatter plot overlaid on the world map that includes the goal and two obstacles (use the locations given in the specs) in order to demonstrate that it is collision-free. Indicate the heading of the robot by a properly sized arrow. Report the final distance to the goal.

   Extra credit: Make the robot stay between the lines $x = 2.5$ and $x = 3.5$ at all times (i.e. treat the two lines as virtual walls). Explain any additional features in your implementation if it is different than above.

Instructions for compiling and running the code:

- Under the directory `me132_tutorial_player`, compile the code by typing `make`.
- The Player configuration file is `planning.cfg` located at `configs`. If you are unsure about how to run Player, refer to the page from last term:

https://www.cds.caltech.edu/~murray/wiki/index.php/ME/CS_132a,_Winter_2011,_Lab_1

Part 2: Lab Session

1. Demonstrate to the TA that the robot can safely arrive at the goal in simulation by running your client code in Stage.
2. Copy the same binary onto the robot's onboard laptop and show that it also works on the actual robot.

In the lab report:

- [15 points] Attach a plot showing the actual robot trajectory recorded in the log file. The plot should follow the same format as in Part 1 (3).