Project 4: Motion Planning for A Simple Car

100 pts

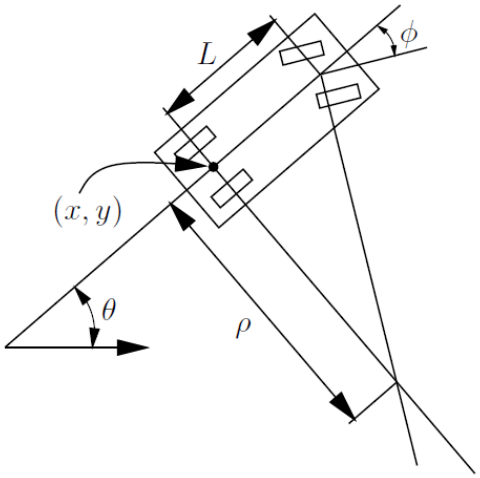
Due: April 9, 11:59pm. Please submit via Canvas.

In this project, you will develop a path planning algorithm for systems with differential constraints, specifically a car-like system.

Part I. General Description

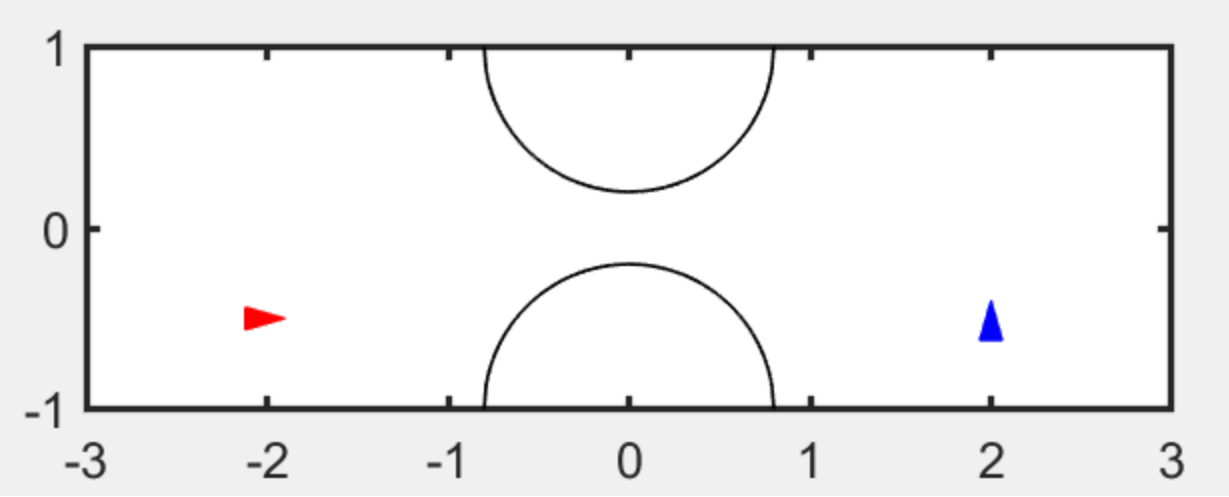
Simple Car

The state of a simple car is represented by (x, y, θ), where (x, y) is its position (the center of its real axis) and θ its orientation (the direction it is pointing). See Figure 1, taken from the textbook by LaValle. (ϕ represents the direction the front wheels. It is negative when it is turned to the right, positive when turned to the left. However, ϕ does not come into play in this homework).



The work space and obstacles

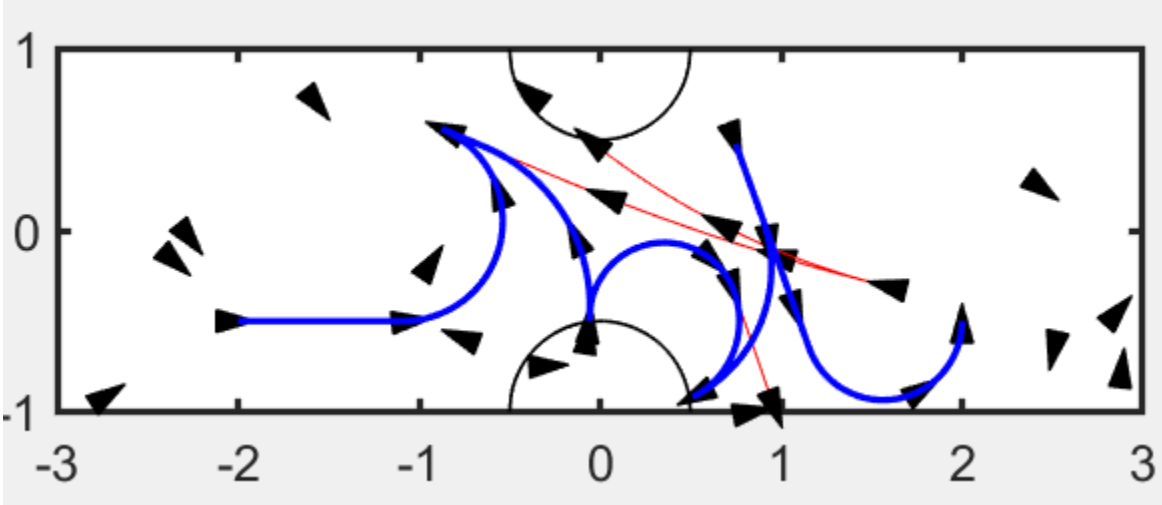
Consider the following work space and obstacles. The dimension of the space is 6 by 2. The obstacles are defined by two half circles that center at (0, -1) and (0, 1) respectively, both having a radius of 1-dt, where dt = 0.2 (All of these are the same as Project 3 except a different dt value is used here). Anything within the two half circles are considered obstacle regions.



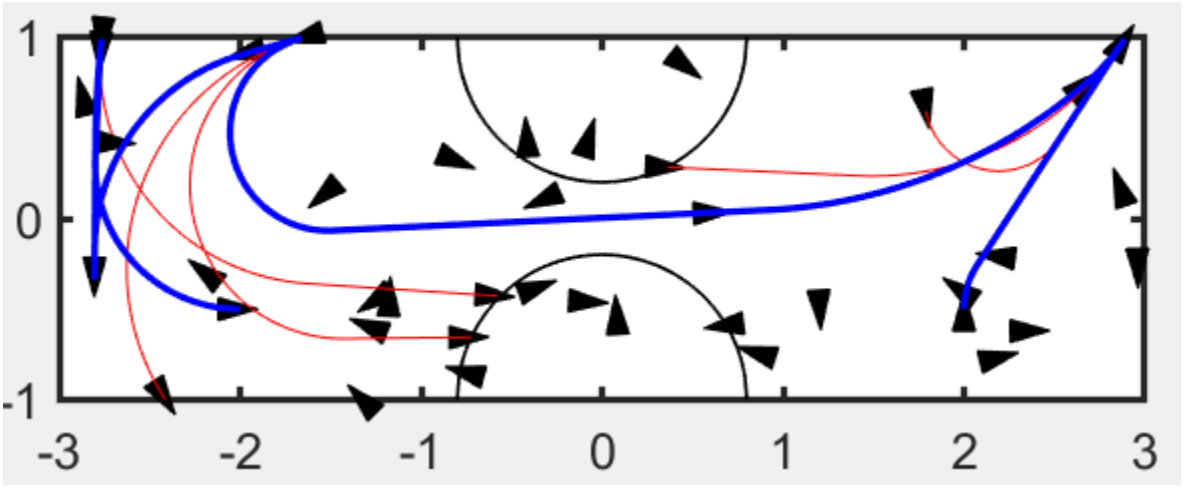
The initial configuration q_I is at (-2, -0.5, 0), in red, and the goal configuration q_G is at (2, -0.5, $\pi/2$), which is in blue.

Part II. Solve the planning problem using RRT

Now solve the planning problem using RRT. Using the single-tree search outlined in sections 5.5.3 and 14.4.3. You should check periodically if the tree can be connected to q_G. This can be easily done by setting $\alpha(i)$ as q_G with a certain probability. The book recommends 0.01. You could have this as an input parameter to your program. The following figure shows how a path to goal is found (I used 0.2 for the probability). As you can see, it is not optimal at all, but nevertheless it is a feasible path. The isolated configurations (black triangles) in the figure are $\alpha(i)$'s that didn't get connected to the graph. All edges are in red. The final path from start to goal is in blue.



Here is another solution:

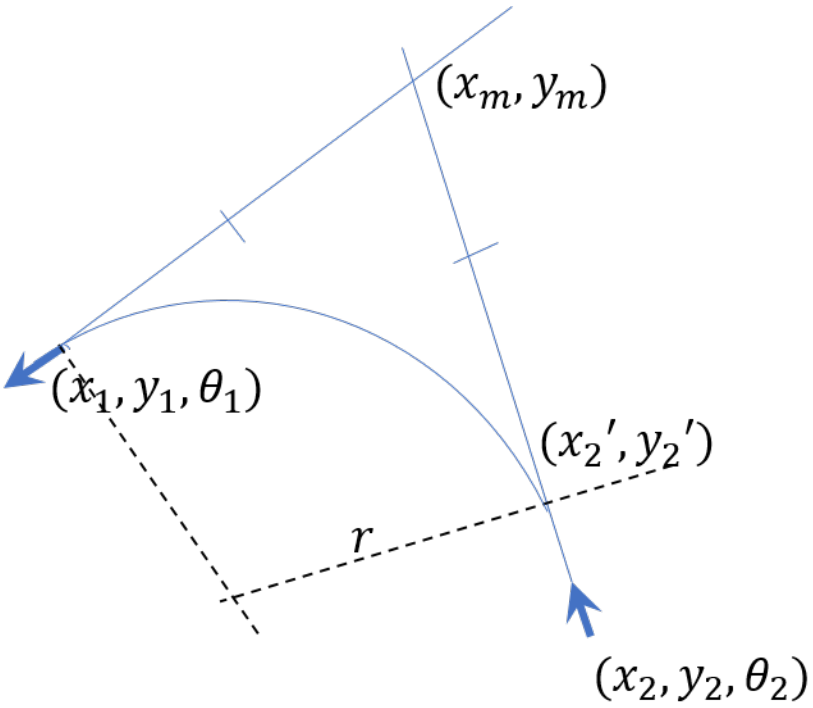


In applying RRT, you would need the following (see section 14.3):

1. a way to sample the state space. You should sample θ in the range of [0, 2π).
2. a distance metric that can determine the nearest neighbor. This can be determined by which configuration has the shortest driving distance to $\alpha(i)$, ignoring obstacles.
3. a collision detection that checks if there is any collision. Currently, our car robot is **just a point with a direction** (like pointed triangles shown in the figure above). So collision checking is easy.
4. a local planner method (**LPM**) that returns a path that connects two vertices, or an empty path if the two vertices cannot be connected.

The last component is a BVP problem and is often challenging. The figure below shows one way we can connect two configurations (x_1, y_1, θ_1) and (x_2, y_2, θ_2). We first extend lines along the directions they are pointing. This will give us an intersection point (x_m, y_m). Assuming without loss of generality that the distance between (x_m, y_m) and (x_2, y_2) is greater. Find another point (x_2', y_2') that is equidistant to (x_m, y_m) as (x_1, y_1) (see Figure below). Draw two dashed lines perpendicular to the two line segments as shown. The intersection of the two dashed lines are the turning center. r is the turning radius. Finally, the path connecting configurations (x_1, y_1, θ_1) to (x_2, y_2, θ_2) is the arc path from (x_1, y_1) to (x_2', y_2'), followed the line segment from (x_2', y_2') to (x_2, y_2). And it is a reverse motion. There are two conditions which may invalidate this path. One is when the car is pointing at opposite directions at configurations (x_1, y_1, θ_1) to (x_2, y_2, θ_2). The other is that $r < r\_min$, the minimum turning radius. For your convenience, I have included my MATLAB implementation of such a local planner (called LPM.m). It takes four parameters, the first two of which are the two configurations to be connected. The third and fourth parameters are minimum turning radius rMin and stepSize (the step size taken in discretizing a path) respectively. For this project, use rMin=0.4, stepSize = 0.02. The LPM.m script returns a sequence of configurations, a path, if it exists.

The matlab code (LPM.m, arcPath.m) can be found here:
http://web.cs.iastate.edu/~gsong/ComS476s20/project4



Part III. Use PRM (section 5.6) to solve the planning problem.

This last part of the assignment is required for graduate students only. 25 bonus points will be given to any undergraduate student who correctly completes the task.

Please use the PRM algorithm described in Figure 5.25 in the textbook to solve the same planning problem given above. As for the number of nodes N, try a different value until you can solve the problem. When connecting to the nodes in the neighborhood, use Nearest K and set K = 15.

What to turn in on Canvas:

1. Your code.
2. A description of what you did.
3. Images of solution paths or movie files.