ECE 5553 Project/Homework 5

[PART 1]:

This HW is based on the MATLAB path planning example "Plan Mobile Robot Paths using RRT".

https://www.mathworks.com/help/nav/ug/plan-mobile-robot-paths-using-rrt.html

We have included the example in this lice script. While defining the state space for the vehicle, the minimum turnn radius for the vehicle is assigned. For different minimum turning radius values (0.5, 1, 1.5, x , 3) repeat the simulation, where x is user defined minumum turning radius value. You can choose 'x' such that 1.5 <x <=2.

- For each minumum turning radius: Show the occupancy map. Plot the search tree from the `solnInfo`. Interpolate and overlay the final path. Hint: see the second figure in the example.
- Check the lenght of the found path, number of iterations to get solution and genreated number of nodes for the solution. Hint: These can be accessed through by checking: "pthObj.pathLength", "solnInfo.NumIterations", "solnInfo.NumNodes". Present these simulation results as a table.
- Compare the resulst and comment on how the minumum turn radius effects the RRT solution.
- Note for some of the minumum turning radius selections, there mihgt not be a solution. If you see this result, just comment on why RRT could not find a solution.

[PART 2]:

Set minumum turning radius to 1m. For different maximum connection distances (0.5, 1, 1.5, 2) repeat the simulation.

- For each maximum connection distance: Show the occupancy map. Plot the search tree from the `solnInfo`. Interpolate and overlay the final path. Hint: see the second figure in the example
- Check the lenght of the found path, number of iterations to get solution and genreated number of nodes for the solution.
- Comment on how different maximum connection distances effects the RRT solution. (You can experiment with your choice of maximum connection distance values to further analyse your thoughts).

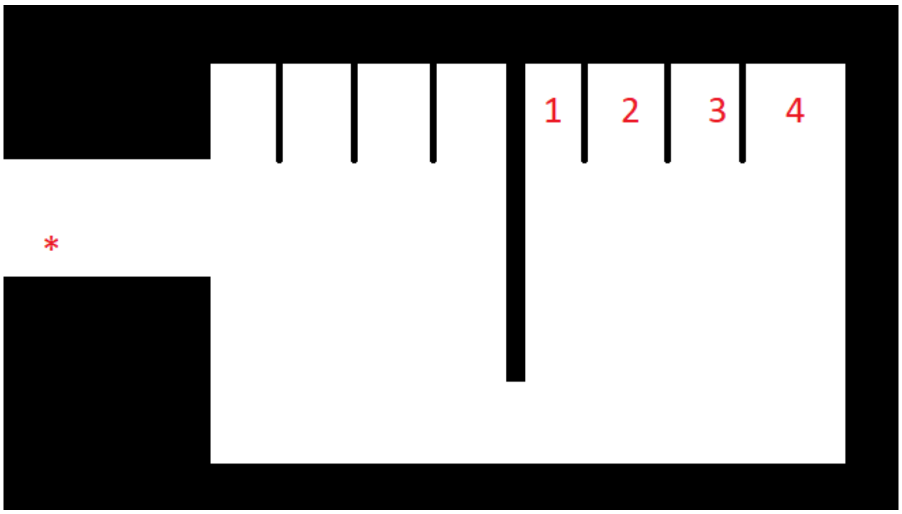
[PART 3]:

Create your own occupancy map from an image. For this part a parking lot image is provided for you: **"plot.png"**.

Tips:

1. To create your own occupancy you can check the example at https://www.mathworks.com/help/nav/ref/binaryoccupancymap.html#buvr49e.
2. To adjust the dimensions of the parking lot set the resouluiton to 20.
3. MAP = binaryOccupancyMap(P, RES) creates a binaryOccupancyMap object from matrix, P, with RES specified in cells per meter.

- Park the car lacated at ' * ' to your choice of numbered parking lots. For this part, set the minumum turning radius to 5 m. Set the maximum connection distance to 2m. This task requires you to initialize the start and goal points and the initial directions of the vehicle.
- For the parking task show the occupancy map. Plot the search tree from the `solnInfo`. Interpolate and overlay the final path.
- Check the lenght of the found path, number of iterations to get solution and genreated number of nodes for the solution.
- Measure the time elapsed for planing the path. Tip: Type "help tic" to command window.



[PART 4]:

Use the same settings used in Part 3.

- Change your path planner to "plannerRRTStar". Tip: type "help plannerRRTStar" to command window to get familiarize with RRT* path planner.
- Park the car lacated at ' * ' to your choice of numbered parking lots (Choose the same parking lot you chosed at Part 3). For this part, set the minumum turning radius to 5 m. Set the maximum connection distance to 2m. This task requires you to initialize the start and goal points and the initial directions of the vehicle.
- For the parking task show the occupancy map. Plot the search tree from the `solnInfo`. Interpolate and overlay the final path.
- Check the lenght of the found path, number of iterations to get solution and genreated number of nodes for the solution.
- Compare the performance of RRT solution with RRT*.
- Measure the time elapsed for planing the path. Tip: Type "help tic" to command window.

[PART 5]:

Use the same occupancy map used in Part 3 and Part 5.

- Change your path planner to "plannerHybridAStar". Tip: type "help plannerRRTStar" to command window to get familiarize with HybridAStar path planner.

- Park the car lacated at ' * ' to your choice of numbered parking lots (Choose the same parking lot you chosed at Part 3 & 4). For this part, set the minumum turning radius to 5 m. Set the MotionPrimitiveLength to 2m.

Sample Code: planner= plannerHybridAStar(stateValidator,'MinTurningRadius',5,'MotionPrimitiveLength',5);

- For the parking task show the occupancy map. Plot the A* search braches and and overlay the final path.Use show function to display the path.

Example: https://www.mathworks.com/help/nav/ref/plannerhybridastar.html#mw_d2c3defd-2480-484e-8e38-4e40171630a4

- Repeat the simulation for MotionPrimitiveLength = 5 m.
- Measure the time elapsed for planing the path. Tip: Type "help tic" to command window.
- Comment on the performance of the planner.

Example Code from MATLAB : Plan Mobile Robot Paths using RRT

This example shows how to use the rapidly-exploring random tree (RRT) algorithm to plan a path for a vehicle through a known map. Special vehicle constraints are also applied with a custom state space. You can tune your own planner with custom state space and path validation objects for any navigation application.

Load Occupancy Map

Load an existing occupancy map of a small office space. Plot the start and goal poses of the vehicle on top of the map.

```
load("office_area_gridmap.mat", "occGrid")
show(occGrid)

% Set the start and goal poses
start = [-1.0, 0.0, -pi];
goal = [14, -2.25, 0];

% Show the start and goal positions of the robot
hold on
plot(start(1), start(2), 'ro')
plot(goal(1), goal(2), 'mo')

% Show the start and goal headings
r = 0.5;
plot([start(1), start(1) + r*cos(start(3))], [start(2), start(2) + r*sin(start(3))], 'r-' )
plot([goal(1), goal(1) + r*cos(goal(3))], [goal(2), goal(2) + r*sin(goal(3))], 'm-' )
hold off
```

Define State Space

Specify the state space of the vehicle using a `stateSpaceDubins` object and specifying the state bounds. This object limits the sampled states to feasible Dubins curves for steering a vehicle within the state bounds. A turning radius of 0.4m allows for tight turns in this small environment.

```
bounds = [occGrid.XWorldLimits; occGrid.YWorldLimits; [-pi pi]];

ss = stateSpaceDubins(bounds);
ss.MinTurningRadius = 0.4;
```

Plan The Path

To plan a path, the RRT algorithm samples random states within the state space and attempts to connect a path. These states and connections need to be validated or excluded based on the map constraints. The vehicle must not collide with obstacles defined in the map.

Create a `validatorOccupancyMap` object with the specified state space. Set the `Map` property to the loaded `occupancyMap` object. Set a `ValdiationDistance` of 0.05m. This distance discretizes the path connections and checks obstacles in the map based on this.

```
stateValidator = validatorOccupancyMap(ss);
stateValidator.Map = occGrid;
stateValidator.ValidationDistance = 0.05;
```

Create the path planner and increase the max connection distance to connect more states. Set the maximum number of iterations for sampling states.

```
planner = plannerRRT(ss, stateValidator);
planner.MaxConnectionDistance = 2;
planner.MaxIterations = 30000;
```

Customize the `GoalReached` function. This example helper function checks if a feasible path reaches the goal within a set threshold. The function returns `true` when the goal has been reached, and the planner stops.

```
planner.GoalReachedFcn = @exampleHelperCheckIfGoal;
```

```
function isReached = exampleHelperCheckIfGoal(planner, goalState, newState)
isReached = false;
threshold = 0.1;
if planner.StateSpace.distance(newState, goalState) < threshold
isReached = true;
end
end
```

Plan the path between the start and goal. Because of the random sampling, this example sets the `rng` seed for consistent results.

```
rng(0,'twister')

[pthObj, solnInfo] = plan(planner, start, goal);
```

Plot the Path

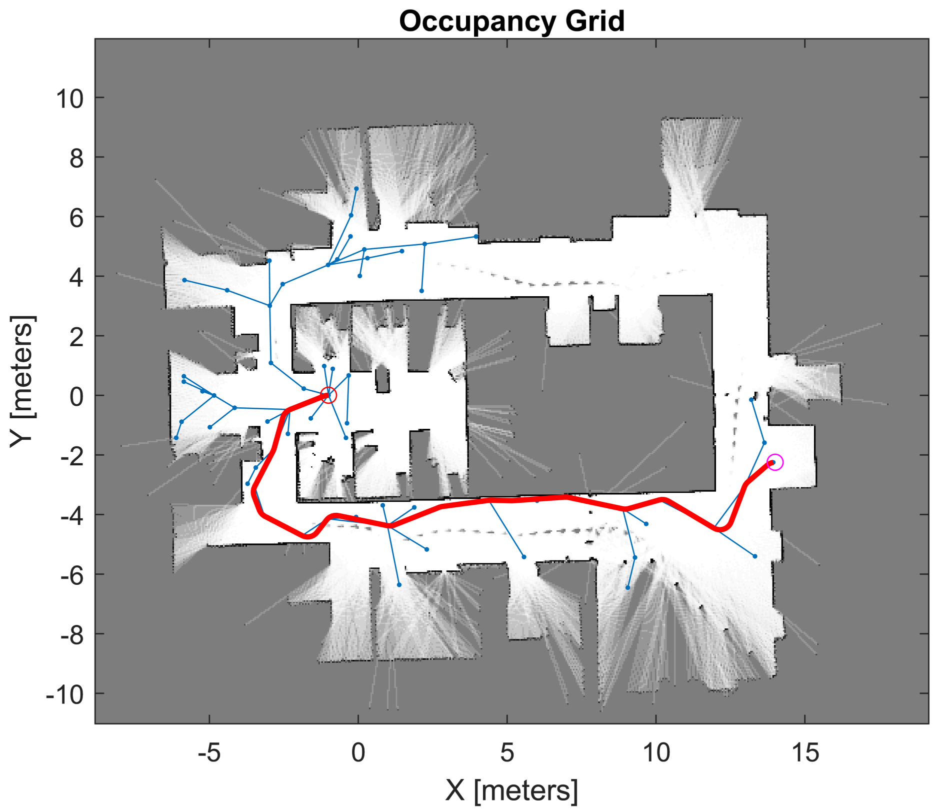
Show the occupancy map. Plot the search tree from the `solnInfo`. Interpolate and overlay the final path.

```
show(occGrid)
hold on

% Search tree
plot(solnInfo.TreeData(:,1), solnInfo.TreeData(:,2), '.-');

% Interpolate and plot path
interpolate(pthObj,300)
plot(pthObj.States(:,1), pthObj.States(:,2), 'r-', 'LineWidth', 2)

% Show the start and goal in the grid map
plot(start(1), start(2), 'ro')
plot(goal(1), goal(2), 'mo')
hold off
```



Occupancy Grid
10
8
6
4
2
0
-2
-4
-6
-8
-10
Y [meters]
-5
0
5
10
15
X [meters]