Lab 6: Vehicle Path Planning with RRTs

In this lab you will implement a Rapidly-exploring Random Tree (RRT) algorithm to perform path planning for a vehicle.

Rapidly-exploring Random Tree (RRT)

RRT is a random search algorithm designed to explore nonconvex and high dimensional spaces. For this lab the goal of RRT is to find a path from an initial state x_0 to a terminal state or set of states X_{goal}. Furthermore, this path must always stays in a subset of the space X_{safe}. We will first implement a standard RRT and then we will adapt this to take into account the kinematic vehicle model.

Figure 1 displays an RRT with

$$
\begin{aligned}
x_0 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
X_{safe} &= \left\{ x \in \mathbb{R}^2 \mid 0 \leq x(1) \leq 100, 0 \leq x(2) \leq 100 \right\} \\
X_{goal} &= \left\{ x \in \mathbb{R}^2 \mid 70 \leq x(1) \leq 75, 45 \leq x(2) \leq 50 \right\}.
\end{aligned}
$$

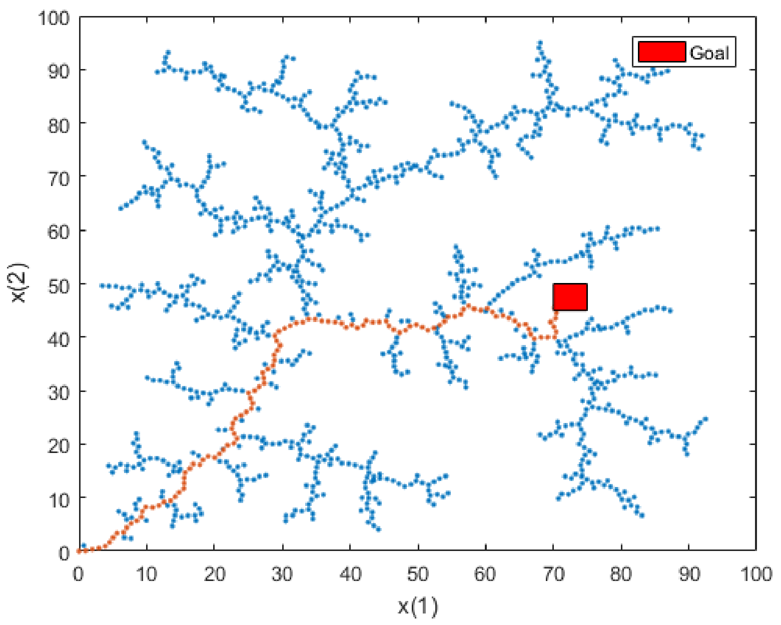

Figure 1: RRT

Each dot in the figure is a node of the RRT. As can be seen all of the nodes of the tree are in X_{safe}

and the red nodes display a path found from x_0 to X_{goal}. For this RRT the distance d between nodes was chosen to be 1.

The following psuedocode describes the standard RRT algorithm. It returns the nodes $x = (x_0, x_1, x_2, ...)$ and the index of each nodes parent $p = (p_1, p_2, ...)$ (note that there is no p_0 since the first node x_0 does not have a parent node). The algorithm uses a while loop so that it will run until it finds a node in X_{goal}.

Algorithm 1 RRT

```
Inputs: $x_0$, $X_{goal}$, $X_{safe}$, $d$

Initialize: $k = 0$
while $x_k \notin X_{goal}$ do
  Choose a random $x_{rand} \in X_{safe}$
  Find node $x_j$ that is closest to $x_{rand}$
  $x_{new} \leftarrow d\frac{(x_{rand}-x_j)}{\|x_{rand}-x_j\|}$     (from $x_j$ move distance $d$ towards $x_{rand}$)

  if $x_{new} \in X_{safe}$ then
    $k \leftarrow k + 1$     (Increment $k$)
    $x_k \leftarrow x_{new}$     (Add new node)
    $p_k \leftarrow j$     (Save index of parent node)
  end if
end while
return $x, p$
```

Note, there are many variations of this algorithm, so it most likely will be presented differently elsewhere. For example often the algorithm is used just to randomly sample a space, not determine a path. In this case there is no X_{goal} set and the algorithm terminates after a specified number of iterations.

1. Implement a standard RRT with

$$\begin{aligned} x_0 &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ X_{safe} &= \{x \in \mathbb{R}^2 \mid 0 \leq x(1) \leq 100, 0 \leq x(2) \leq 100\} \\ X_{goal} &= \{x \in \mathbb{R}^2 \mid 70 \leq x(1) \leq 75, 45 \leq x(2) \leq 50\} \\ d &= 1. \end{aligned}$$

   Plot the resulting nodes of the RRT. As in Figure 1 indicate the sequence of nodes that reach X_{goal}. Also report the total number of nodes in the tree and the number of nodes in the sequence that reaches X_{goal}.

2. Implement a standard RRT with X_{safe} as shown in Figure 2 and

$$\begin{aligned} x_0 &= \begin{bmatrix} 0 \\ 2 \end{bmatrix} \\ X_{goal} &= \{x \in \mathbb{R}^2 \mid 900 \leq x(1) \leq 950, 1 \leq x(2) \leq 1.5\} \\ d &= 1. \end{aligned}$$

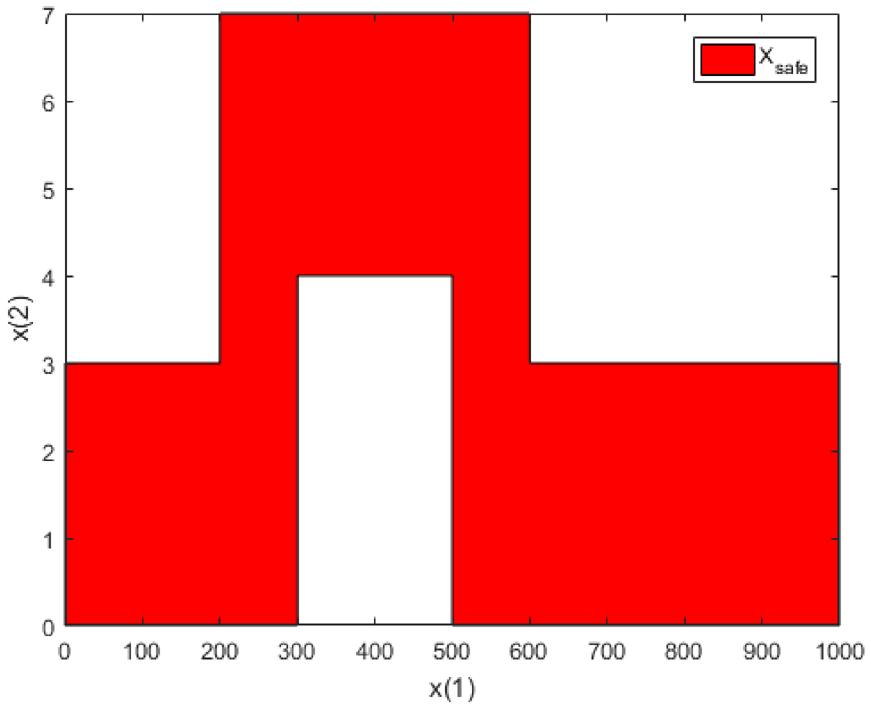

Figure 2: X_{safe}

Plot the resulting nodes of the RRT and indicate the sequence of nodes that reach X_{goal}. Also report the total number of nodes in the tree and the number of nodes in the sequence that reaches X_{goal}.

RRT using a Vehicle Kinematic Model

Generating a path with a standard RRT algorithm is problematic because there are no guarantees that the vehicle will be able to follow the path. However, we can modify the RRT algorithm so that only paths that the vehicle can follow will be generated.

In this lab we will use a simple kinematic vehicle model:

$$\dot{X} = v_x \cos(\psi)$$
$$\dot{Y} = v_x \sin(\psi)$$
$$\dot{\psi} = \frac{v_x}{L}\tan(\delta)$$

where X and Y are the lateral and longitudinal position and ψ is the heading angle. The steering angle δ is the input.We assume that $v_x = 30$ m/s is constant and the wheelbase $L = 3$ m. We denote the state vector $x := \begin{bmatrix} X \\ Y \\ \psi \end{bmatrix}$.

In order to incorporate the constraints imposed by the vehicle model we only need to modify the step of the algorithm were we assign a value to x_{new}. In the standard RRT we just choose x_{new} in the direction of x_{rand}, but this may not be possible for a vehicle. Therefore, we want to select a control input δ such that the vehicle moves towards x_{rand}.

As discussed in class this can be done many different ways (PI control, MPC, sliding mode, etc) however for this example we will use a simpler, brute force approach. To get x_{new} we will try multiple different steering angles and then check which one is the best. Specifically, for each value of $\delta \in (-20, -18, ..., 18, 20)$ simulate the vehicle kinematic model (using ode45) for $T = 0.1$ seconds starting at x_j and check that the entire state trajectory is in X_{safe}. Of the safe trajectories, the final state that ends closest to x_{rand} is assigned to x_{new} If none of the trajectories are safe then we return to the beginning of the while loop and select a new x_{rand}. Otherwise, once x_{new} is selected the algorithm continues as described.

3. Implement a RRT with the vehicle kinematic model and brute force control strategy described above with

$$x_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$
$$X_{safe} = \{x \in \mathbb{R}^3 \mid 0 \le X \le 1000, -20 \le Y \le 20, -\pi \le \psi \le \pi\}$$
$$X_{goal} = \{x \in \mathbb{R}^3 \mid 900 \le X \le 950, -1 \le Y \le 1, -\pi/6 \le \psi \le \pi/6\}\}$$

   Plot the X and Y components of the resulting nodes of the RRT and indicate the sequence of nodes that reach X_{goal}. Also report the total number of nodes in the tree and the number of nodes in the sequence that reaches X_{goal}.

4. Now modify the safe set so that X and Y are constrained to X_{safe} in Figure 2 and $-\pi \le \psi \le \pi$. Implement the RRT with the vehicle kinematic model with this safe set and

$$x_0 = \begin{bmatrix} 0 \\ 2 \\ 0 \end{bmatrix}$$
$$X_{goal} = \{x \in \mathbb{R}^3 \mid 900 \le X \le 950, 1 \le Y \le 1.5, -\pi/6 \le \psi \le \pi/6\}\}$$

   Plot the X and Y components of the resulting nodes of the RRT and indicate the sequence of nodes that reach X_{goal}. Also report the total number of nodes in the tree and the number of nodes in the sequence that reaches X_{goal}.

   How does this RRT compare to the one found in Problem 2? Roughly, how much more computation time does it take to run?

5. The RRT algorithm presented here is very basic. There are many modifications and variations in RRT algorithms for different applications. What are some potential drawbacks or issues with the RRT algoriothm? What are some possible modifications that could be made to the algorithm to improve it's performance for vehicle path planning?

6. Suppose we want to implement a more advanced vehicle model (like the dynamic model used in Lab 5) with the RRT algorithm. Briefly describe what modifications to the algorithm would be required to do this.

Optimization-Based Navigation

On bCourses there are 2 script files that implement the various optimization-based navigation strategies discussed in class:

- `navigation_NLP.m` - navigation problem formulated as a nonlinear program
- `navigation_MIQP_bigM.m` - navigation problem formulated as a mixed-integer quadratic program

For the following problems you will modify these files.

For the optimization-based navigation you will need to install the following Matlab packages:

- Multi-Parameteric Toolbox (MPT) - This toolbox includes Yalmip and other tool for optimization based control and path planning. It is available from `http://people.ee.ethz.ch/~mpt/3/`.
- IPOPT - This is an interior point solver for nonlinear problems. The easiest way to install it is via the OPTI Toolbox which can be downloaded from `https://www.inverseproblem.co.nz/OPTI/index.php/DL/DownloadOPTI`
- Gurobi - This is a solver for integer programming problems. It can be downloaded from `http://www.gurobi.com`. To use Gurobi you will need to request an academic license. This license allows full use of Gurobi, but must be activated while connected to internet on campus(i.e. AirBears2). After activation it can be used anywhere. Instructions for installing Gurobi in Matlab can be found here: `http://www.gurobi.com/documentation/7.0/quickstart_windows/matlab_setting_up_gurobi_f.html`

7. Modify the `navigation_NLP.m` script to find a feasible path contained in the safe space described in Figure 3 with

$$x_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$
$$x_T = \begin{bmatrix} 850 \\ 1 \end{bmatrix}$$

   where x_0 is the initial condition and x_T is the terminal constraint. Since the state space for this problem is much larger than the example in the file you may need to change the variables `N`, `sampling`, `dzmin`, and `dzmax` to get a good solution in a reasonable amount of time. I got reasonable results changing `N = 50` and `dzmax = -dzmin = [20; 2]`.

   Include the plot of the obstacles and resulting trajectory. Comment on the computational time compared to the RRT method.

8. Repeat Problem 7 using the MIQP formulation in `navigation_MIQP_bigM.m`.

   Include the plot of the obstacles and resulting trajectory. Comment on the computational time compared to the RRT and NLP methods.

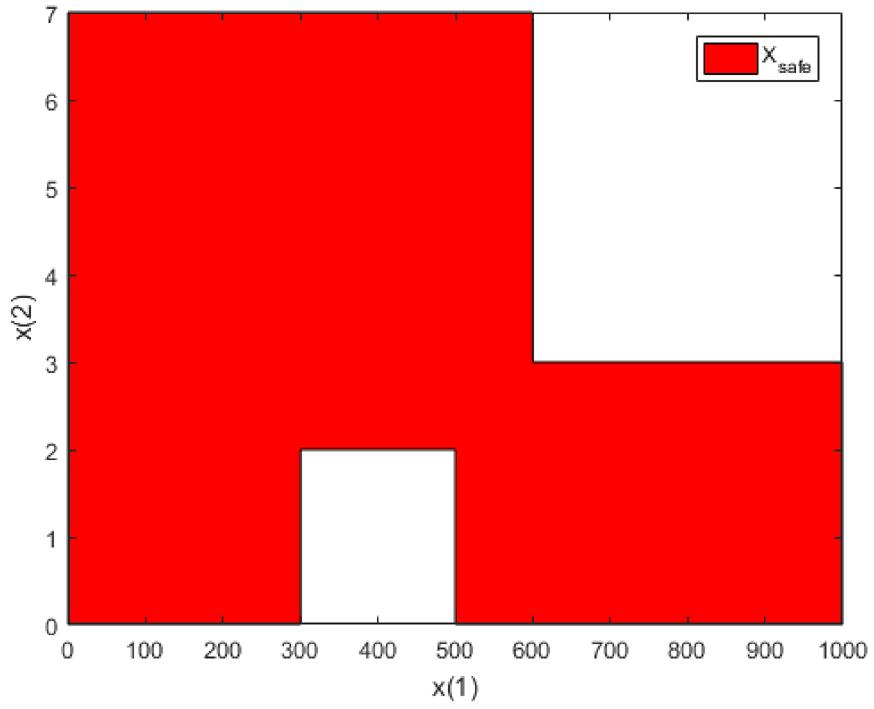

Figure 3: X_{safe}