CS49-ROB Assignment 4
Sampling-based Motion Planning

In this assignment, you will implement the Rapidly-exploring Random Trees star (RRT*) method for three different robot systems, i.e., 2D point-mass, circular rigid body, and a rectangular rigid-body in a 2D environment. Algorithm 1 outlines a standard RRT* algorithm. However, if we remove the lines highlighted in red, it becomes a traditional RRT algorithm. The supplementary python code file provided with this assignment contains an implementation of the standard RRT algorithm. Please extend that python code to address the following questions.

Algorithm 1 RRTstar algorithm

```
1: $\mathcal{T} \leftarrow$ InitializeTree();
2: $\mathcal{T} \leftarrow$ InsertNode($z_{init}, \mathcal{T}$);
3: for $i = 1$ to $i = N$ do
4:     $x_{rand} \leftarrow$ Sample($i$)
5:     $x_{nearest} \leftarrow$ Nearest($\mathcal{T}, x_{rand}$)
6:     $r_{valid}, r_{cost} \leftarrow$ Steer($x_{nearest}, x_{rand}$);
7:     if $r_{valid}$ then
8:         $X_{near} \leftarrow$ Near($\mathcal{T}, x_{rand}$);
9:         $x_{min} \leftarrow$ ChooseParent($X_{near}, x_{nearest}, x_{rand}$);
10:        $\mathcal{T} \leftarrow$ InsertNode($x_{min}, x_{rand}, \mathcal{T}$);
11:        $\mathcal{T} \leftarrow$ Rewire($X_{near}, x_{rand}, x_{min}, \mathcal{T}$);
12:    end if
13: end for
```

Question 1 [3+3+3=9pts]

In this question, implement and add the following functions to turn the provided RRT code into RRTstar:

- Near (Algorithm 1, Line 8): Given a sample $x \in X$, the tree $T = (V, E)$ and the ball region $\mathfrak{B}_{x,r}$ of radius r centered at x, the set of near vertices is defined as: $\text{Near}(x, T, r) := \{v \in V : v \in \mathfrak{B}_{x,r}\} \mapsto X_{\text{near}} \subseteq V$. More specifically, $X_{\text{near}} = \{v \in V : d(x, v) \leq \gamma(logi/i)^{1/n}\}$ where i is the number of vertices, n represents the dimensions and γ is a constant.

- ChooseParent (Algorithm 1, Line 9): This procedure is used to search the list X_{near} for a state, x_{min} which provides the shortest, collision-free path from the initial state x_{init} to the random sample x_{rand}.

- Rewire (Algorithm 1, Line 11): Here, the algorithm examines each vertex $x' \in X_{\text{near}}$ lying inside the ball region centered at x_{rand}. If the cost of the path connecting x_{init} and x' through x_{rand} is less than the existing cost of reaching x' and if this path lies in obstacle free space x_{free}, then x_{rand} is made the parent of x'. If these conditions do not hold true, no change is made to the tree and the algorithm moves on to check the next vertex. This process is iteratively performed for every vertex x' present in the X_{near}. *Note:* If a vertex x' is rewired, its cost must be updated to reflect the change; and if x' has descendants, they should also be updated. Your rewire() implementation should be sure to do this.

Once implemented, include in your report:

- The images of tree and final path found for both RRT and RRT*.
- The following table comparing RRT and RRT* algorithms. You can run the planning algorithm N times(N $\geq$ 30 to avoid bias) and get the average.

2D point-mass	**RRT**	**RRT***
path cost	-	-
computation time	-	-

Question 2 - Circular Rigid Body [2 pts]

In this question, extend both RRT and RRTstar algorithms to plan for a circular rigid body of radius 1 unit (Fig 1). **Hint** Think of adapting the collision checker and final trajectory

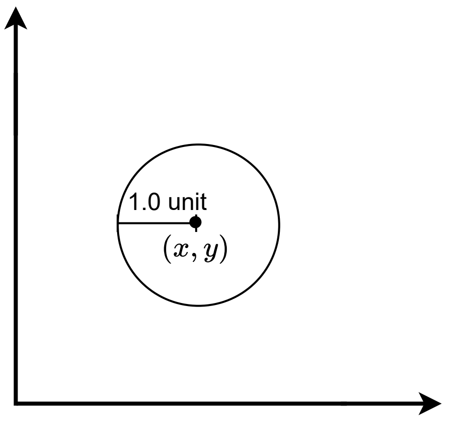

Figure 1: Circular Rigid Body of radius 1 unit.

visualizer. In the report include the following:

- The images of tree and final path found for both RRT and RRT*.
- The following table comparing RRT and RRT* algorithms.

Circular Rigid body	**RRT**	**RRT***
path cost	-	-
computation time	-	-

Question 3 - Rectangular Rigid Body [4 pts]

In this question, extend both RRT and RRTstar algorithms to plan for a rectangular rigid body of dimensions $L = 1.5, W = 3$ units. The rigid body has three dimensional configuration space, i.e., (x, y, θ), as shown in Fig 2.

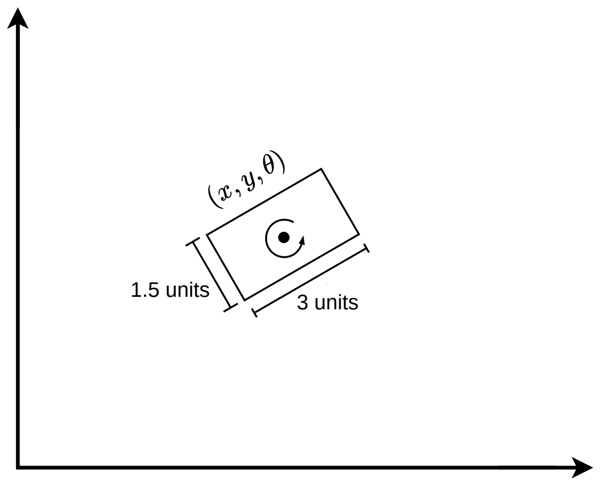

Figure 2: Rectangular rigid body.

Hint You will need to modify the random sampler to sample positional (x,y) and rotational (theta) coordinates. In addition, you will also need to adapt the collision checker and final trajectory visualizer. For collision-checking, consider robot and obstacles as boxes and check collision by finding the overlap between boxes. In the report include the following:

- The images of tree and final path found for both RRT and RRT*.
- The following table comparing RRT and RRT* algorithms.

Rectangular Rigid body	RRT	RRT*
path cost	-	-
computation time	-	-

Using the provided code

The code provided in assign1.py contains a CLI and animations to visualize how the program is running. Run "python3 assign1.py" to watch a point-mass robot navigate its environment using RRT. The CLI contains several useful flags, including options to configure the algorithm used (RRT or RRT*), and the geometry of the robot (point mass, circle, or rectangle). Only the RRT/point mass combination is implemented; in the assignment, you'll add code for the other combinations.
Be sure to read over all the code, and understand what it is doing. For question 1, you will only have to implement 3 functions. For questions 2 and 3, you will have to make modifications in various places in the code. Feel free to make any changes you feel are

necessary to complete the assignment - we'll compare your submission against the original when grading.

Submission instructions

Your submission should consist of two items:

- Code to solve all 3 questions.
- A PDF report. As discussed above, this report should contain results (path costs, computation times, and visualizations) for both RRT and RRT*, using each of the robot geometries. (point mass, circle, and rectangular rigid body).

  Note: if you have any special explanations of how your code works, or instructions for running it, feel free to place them either in the report or in a separate README file.

Bundle all files into a zip or tarfile, and submit it via Brightspace. Name your submission "<your_username>.zip" or "<your_username>.tar.gz". For example, I would name my submission "Hanwen_Ren.zip".