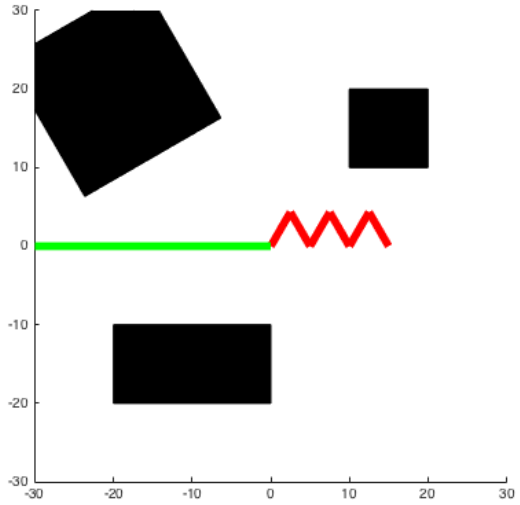
1 Introduction

In this assignment you will be writing a program to help guide the six link robot shown in the figure below from one configuration to another while avoiding the objects in the workspace. This assignment will give you some experience working with planning methods based on random sampling. The robot shown in the figure is comprised of six revolute links and its configuration can be specified with a vector $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ where each entry is an angle in degrees between 0 and 360. In the code provided the function `SixLinkRobot` computes the layout of all of the links as a function of the 6 parameters.



2 Assignment: Probabilistic Roadmap [15 points]

2.1 Files in this Assignment

`PRM.m` [*] - The function for computing a probabilistic roadmap (PRM) of configuration space

`SixLinkPRMScript.m` - The script for simulating the planned motion

`CollisionCheck.m` - The function for determining if two sets of triangular faces overlap

`ShortestPathDijkstra.p` - The function for finding the shortest path through a sparse graph using Dijkstra's algorithm

`RandomSampleSixLink.m`, `DistSixLink.m`, `LocalPlannerSixLink.m` - Helper functions called by `PRM`

`AddNode2PRM.m` - The function for adding a node to thePRM

`SixLinkRobot.m` - The function for computing the layout of our six link robot as a function of the six joint angles, $\theta_1 \cdots \theta_6$

`appendFV.m`, `boxFV.m`, `transformFV.m`, `my_linspace.m` - Helper functions for constructing the simulation environment

`evaluate.p` - Code that evaluates your probabilistic roadmap

`submit.m` - Script which calls the `evaluate` function for generating submission result

* indicates the file you will need to edit

2.2 Tasks

For this assignment you are being asked to add code to complete the implementation of a Probabilistic Roadmap planner that will guide the robot safely from one configuration to another. Before you begin, you will need to copy over the `triangle_intersection` code that you wrote for the previous assignment since this code uses the same `CollisionCheck` function to determine whether a given set of configuration space parameters will lead to a collision.

Your main job for this assignment is to complete the `PRM` function which computes a probabilistic road map of the configuration space.

```
1 function roadmap = PRM (RandomSample, Dist, LocalPlanner, nsamples, k)
```

The input arguments to this function are explained below:

`RandomSample`: a function which returns a random sample in freespace.

`Dist`: a function which can be used to compute the distance between a given random sample and all of the samples generated so far

`LocalPlanner`: a function that can be used to test whether two configuration space points, x1 and x2, can be joined by a straight line. That is `LocalPlanner (x1, x2)` will only return true if the straight line between x1 and x2 does not intersect any configuration space obstacles.

`nsamples`: the total number of random samples to be generated

`k`: number of neighboring samples to be considered during PRM construction

The section of code that you are asked to complete should perform the following steps:

1. Use the array `distances` generated by `Dist` function to determine the indices of the k nearest neighbors. (Hint: you may find the sort function useful here.)
2. For each of those neighbors, it should determine whether or not it can forge a path between those two locations using the `LocalPlanner` function.
3. If a path exists, it should then update the set of edges as follow:

- Add an entry to `edges` array indicating the indices of the two samples that have just been joined.
- Add a corresponding entry to the `edge_lengths` array indicating the length of this new edge.
- Increment the `nedges` variable to reflect the fact that a new node has been added.

These edges and edge lengths constitute a graph which will be used later by the `ShortestPathDijkstra` routine to plan paths through the roadmap.

Once you are done you can test your code by running the `SixLinkPRMScript` script which uses the `PRM` function to construct a roadmap and then attempts to add edges to the start and goal location and plan paths through the graph using Dijkstra's algorithm. If it succeeds in planning a route it shows an animation of the resulting motion.

Note that because of the stochastic nature of this procedure the scheme may sometimes fail to find a path even when one exists. You can experiment with the number of samples, and the number of neighbors as well as the layout of the obstacles.

2.3 Submission and Grading

To submit your result to our server, you need to run the command `submit` in your MATLAB command window. A script will then evaluate your probabilistic `roadmap` and generate output file (`PRM.mat`) to be uploaded to the Coursera web UI. If you pass our test cases, you will get the full score in this assignment. You may submit your result multiple times, and we will count only the highest score towards your grade.

Part	**Submitted File**	**Points**
Probabilistic Roadmap	`PRM.mat`	15

3 Stretch Goal

For those of you that are interested in a much more substantial challenge, I invite you to try to implement the Rapidly Exploring Random tree method to plan paths between the start and the end goal. You can use the PRM code as a starting point but it will need to be modified quite extensively to implement the tree construction scheme.