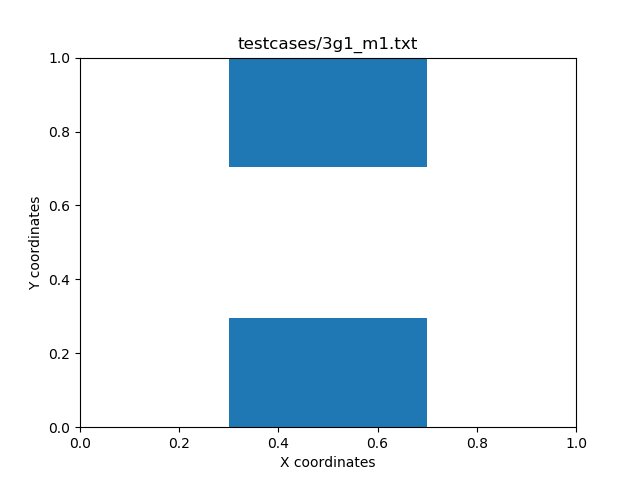
# ASSIGNMENT 2: Canadarm – Continuous Motion Planning

1.1 Configuration space

The configuration space for the problem is the set of all possible robot configurations.

* A configuration is the parameters that uniquely define the position of every point on the robot: q = (q1, q2, … qn)
* [Eex, Eey; A1, A2 ... An; L1, L2 … Ln] makes a configuration
* The forbidden region is made up of the obstacle list
* Free space is the C- space without the forbidden region
* = {x | x is a pose of the robot}



Forbidden region

Free Space

The method used for searching in the continuous search was Probabilistic Road Map (PRM). From the empty graph G, a configuration q is sampled by picking a random arm pose. The random arm pose is checked whether they are inside the min and max conditions, are not inside an obstacle, and do not form any collisions (if q -> Q\_Free, then add to G). This is repeated for N samples. Each of these configurations become a node to be added to the state graph. Using matplot in python, the EE2 of the robot arm can be plotted for a visual representation.

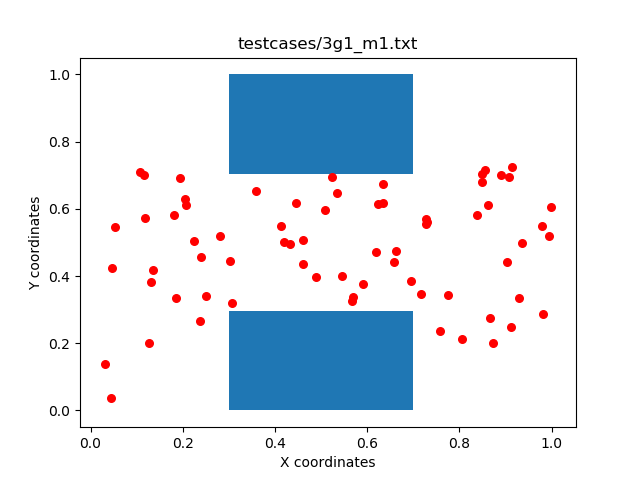
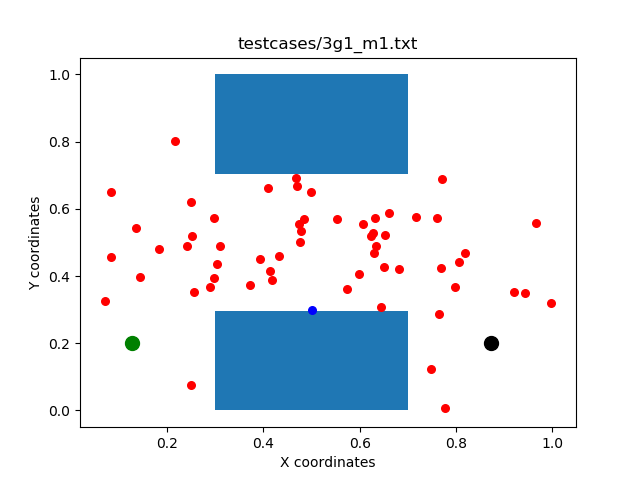
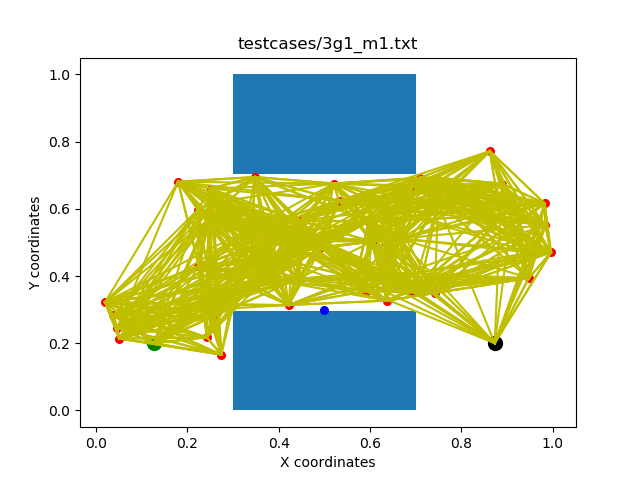


Figure 2: PRM random sampling of points

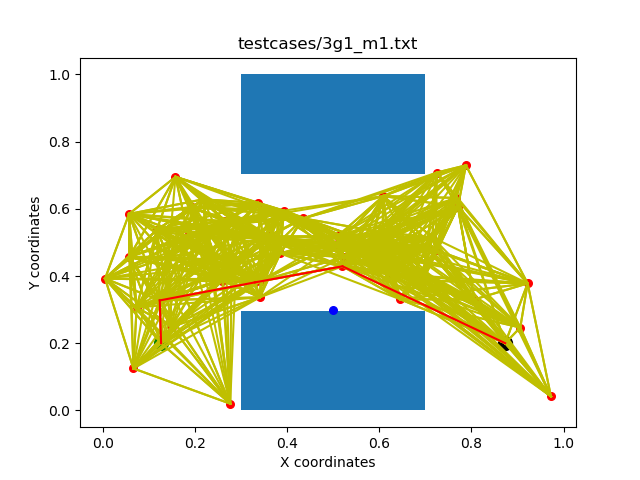
Additionally, the goal (q\_goal) and start (q\_start) are added as nodes. The grapple points are also plotted; however, they are only included as nodes if there is more than one grapple point, in which it will be considered an initial goal point.



To create paths, each node q is linked to its nearest neighbour q’. If there is a free local path, such that there are no collisions, an edge added between q and q’. This is repeated until each node q is connected to its nearest k neighbours.



Using a search algorithm, the path from start to goal can be found:



Question 2.

2.1 Strategy

Sampling strategy:

In order to generate random configurations:

* Random angles are generated for number of arm links
* Random lengths are generated for number of arm links between such that
* The position of the grappled end effector.

The configuration is checked whether it is valid, and if it is, it is appended to a list of nodes (More on valid checking in 2.2.3).

From these input values, the position of the un grappled end effector can be calculated and plotted for a visual representation of the end point.

2.2.2 Connection strategy

The connection strategy involves comparing every valid node q to every other valid node q’. Iterating through the nodes, if the node is not being compared with itself, and there is no collision between the two nodes in a straight line, then the Euclidian distance between the two nodes is found and appended to a list of distances. This list is then sorted from smallest to largest distances, and for the first k distances in the list, their corresponding node q’ is added as a neighbour for q. Additionally, an input parameter ‘force’ can be used, which if defined as ‘True’, forces the goal node’s neighbours to add the goal as one of their neighbours. This ensures that there are nodes that have the goal as a neighbour, if a cluster of random nodes add each other as neighbours instead of the goal node.

2.2.3 Checking if a configuration is valid

To check whether a configuration is valid, the following checks are applied:

* Check the node is inside the specified min and max
* Check the un-grappled end effector is not inside an obstacle
* Check whether the configuration (arm links in addition to EE) are in collision with obstacle
* The check for collision with itself

To check whether a node is inside the specified min and max, the x and y coordinate points are compared such that:

x\_min <= node.x <= x\_max) and y\_min <= node.y <= y\_max

To check the un-grappled end effector is not inside an obstacle, the EE’s coordinates are compared with each obstacle coordinate points such that:

Where x1 and x2 are the x coordinate min and max, and y1 and y2 are the y coordinate min and max for an obstacle.

To check for a configuration collision, a bounding box test is done for each of the point of a configuration against each obstacle, in addition to edge checks for all obstacle edges with robot arm links. To test for an edge collision for sampling, self collision and the connection strategy, the following check is completed as follows:

Each obstacle is a rectangle with four corners, which when paired make up the four edges of a rectangle. Each obstacle edge containing two corners (A and B) is compared with two points (B and C), be they two node points or two points of a robot arm segment. The following cross product is used to find the orientation of three points:

(ax,ay), (bx,by), (cx,cy) = (bx −ax)∗(cy −ay)−(cx −ax)∗(by −ay)

area\_abc = (b[0] - a[0]) \* (c[1] - a[1]) - (c[0] - a[0]) \* (b[1] - a[1])  
area\_abd = (b[0] - a[0]) \* (d[1] - a[1]) - (d[0] - a[0]) \* (b[1] - a[1])  
area\_cda = (d[0] - c[0]) \* (a[1] - c[1]) - (a[0] - c[0]) \* (d[1] - c[1])  
area\_cdb = (d[0] - c[0]) \* (b[1] - c[1]) - (b[0] - c[0]) \* (d[1] - c[1])

If abc and abd have diﬀerent orientations and cda and cbd have diﬀerent orientations, there is a line collision between ab and cd.

if (np.sign(area\_abc) != np.sign(area\_abd)) and (np.sign(area\_cda) != np.sign(area\_cdb)):  
 return True

Otherwise, there is no collision.

3.1 Scenarios

The method used of PRM is well suited for repeated planning between different pairs of start and goal nodes in the same environment. This is useful in the case of multiple grapple points, which can be broken down into multiple search problems between start node to required grapple node, then to grapple node to goal node.

Different gaps