### COMP3702

### Assignment 2

Group 666

Linfeng Liu 44563376

Yunwen Wu 44816371

Fan Yang 44594213

1. Please define the Configuration Space of the problem and describe which method you use for searching in continuous space.

The dimension of the C-space is dependent on how many line segments the robot has. For example if the robot has three segments, then we can define the C-space as below:

C = ,

Where (x,y) is the position of the grappled end effector in the workspace, a1 is angle between the first segment and the grappled end point, a2/a3 are joint angles ordered relative to the end effector, L1/L2/L3 are the lengths of each link which are also ordered relative to the end effector.

So for 3 segments robot, C-space is 8th dimension. If the segments increases so will the dimension as the number of angles and lengths will increase equally.

Method used for this assignment is Probabilistic roadmaps. Full demonstration on how the method is applied in this assignment is showed in question 2.

1. Describe the strategy you applied to develop each of its components.

The whole algorithm consists of three main parts. First sample a configuration q uniformly at random. In the sampling process, also test whether the configuration is valid or not. Then check whether q is a neighbour of the existing configurations. This step consists of two parts. 1) Check the distance as robot can not be moved more than 0.001 units between each primitive step. 2) Check whether the path between two configuration is not in collision. How the whole process is implemented is shown as below:

For I in range(2000) {

Q = Uniformly\_sample();

For c in All\_Configurations{

If distance between q and c less than 0.001 units{

If path between q and c is valid{ add q and c to each others’ neighbour }

}

}

}

After connecting all the edges, use uniformed search algorithm to find the solution.

More detailed description for each part and how the function in implemented in code is shown below.

* 1. Sampling

Each configuration is sampled uniformly at random. After finding all the angles and lengths using function Random, apply test to make sure that the configuration is not in collision with the obstacle, bounds and itself.

* 1. Add neighbours

Now we have one sampling point q and it needs to be connected to other configuration. Applying two steps below to see if q can be added to an existing node’s neighbour.

* + 1. Check distance

A function must be applied to check whether the distance between q and other configurations is under requirement. Compute the difference between two same sequence angles and find the maximum one among these. If the maximum distance is larger than the primitive step limit plus the tolerance than it cannot be connected to this node.

* + 1. Check path

If the step size is not greater than primitive step limit, then check whether the path between two configurations is valid or not. To achieve this goal, we find another 20 configurations between the two steps and if these 20 won’t be in collision then we can say that the path is safe. To find the configurations in between, first calculate the distances of angles and lengths of the same sequence. Then increase the angles and lengths in equal proportions each time until we have enough nodes. Check each time whether the points is in collision to make sure the path between is valid. Implementation in code is shown below:

List\_x = distance between angles

List\_y = distance between lengths of segments

For I in range(20){

For n in range( number of angles/segments){

New\_angle[n] = q\_angle[n] + list\_x[n]\*0.05\*n

New\_length[n] = q\_length[n] + list\_y[n]\*0.05\*n

Using new angles and length to construct new configuration W

If W is in collision with obstacle/bounds/itself { return False}

}

Return True

}