Team members:

I-Chen Jwo, ijwo

Chien-Chih Ho, chienchh

Luka Eerans, lte

1. Run the RRT planner for the 2D configuration space several times (10 or more). Report average path length, average plan time and average number of vertices in the tree. Submit a video of at least one run. What goal sampling probability did you use? (5 pts)

   Goal sampling: 0.2

RRT	distance	time	vertices
	22	0.26014209	31
	32	0.34433484	41
	72	0.72389698	76
	24	0.42818618	47
	48	0.38778496	49
	102	0.65685105	101
	24	0.30922198	32
	42	0.35664392	44
	78	0.75457406	129
	20	0.33697796	38
Average	46.4	0.4558614	58.8

2. Run the bidirectional RRTConnect planner for the 2D configuration space several times (10 or more). Again report average path length, average plan time and average number of vertices in the tree. How does the bidirectional version compare to the regular RRT? (5 pts)

   The percentage is the distance/time/vertices saved comparing to RRT.

RRT CONNECT	distance	time	vertices
	68	0.65195394	68
	20	0.1632669	16
	44	0.30919313	42
	60	0.332371	50
	40	0.25317502	42
	64	0.30327201	40
	36	0.20921493	20
	56	0.43809199	54
	50	0.38666987	66
	38	0.25343323	34
Average	47.6	0.3300642	43.2
Percentage	-3%	28%	27%

3. Run the RRT planner and the bidirectional RRTConnect planner for the WAM arm. Report aver- age path length, average plan time and average number of vertices in the tree. How do the two algorithms compare? Is it more, less or equally useful to use bidirectional planning in the higher dimensional configuration space? Why? Submit a video of at least one run. (5 pts)

The percentage is the distance/time/vertices saved comparing to RRT. It is more useful in the high dimensional space since it takes in much more parameters and consume more time to find a path. On average, it takes 115.5 vertices to complete a path. It is twice as the 2-D RRT takes. The more complicated it is, the more useful RRT Connect is. RRT Connect expand faster and get out of local minimum by expanding from start and goal config.

RRT	distance	time	vertices
	93	1.87397194	70
	75	1.86698008	63
	77.5	0.82414699	40
	347.5	4.32344103	147
	265	4.20416188	118
	267.5	2.7116859	129
	340	4.5186131	145
	150	3.22639203	94
	197.5	6.44570208	213
	282.5	4.13756323	136
Average	209.55	3.41326582	115.5

RRT CONNECT	distance	time	vertices
	30	0.30356598	14
	65	0.9474051	40
	247.5	3.02392602	140
	80	1.10299778	50
	80	0.87681317	36
	180	2.25613093	92
	72.5	2.13229799	42
	72.5	1.14984608	46
	120	2.30740595	60
	65	0.60544395	26
Average	101.25	1.4705833	54.6
Percentage	52%	57%	53%

4. Utilize the path shortening algorithm to shorten the path. What cost function do you use in the shortening algorithm? Run the planner 10 times and use the path shortening algorithm to reduce the path. What is the average reduction in path cost? Submit a video of a path before and after shortening. (5 pts)

RRT Original Path	Shortest Path	Reduction in Path Cost
382.5	27.5	355
165	165	0
50	22.5	27.5
142.4	117.5	24.9
105	85	20
202.5	110	92.5
90	72.5	17.5
172.5	125	47.5
35	12.5	22.5
55	40	15

For the path-shortening algorithm, there are several ways to quantity how well you are shortening the path. Ideally a cost function would return a scalar feedback signal whose minimization in value would quantify progress “in the right direction”. After all, a cost function of a path shortening algorithm, is nothing but the cost of shortening one way, whose magnitude could be compared with the cost of shortening another way in order to decide on which of the 2 ways to pick.

The most straightforward and obvious cost function would be the total length of the path. We will thus judge how well the path shortening algorithm works based on how much shorter the shortened path is than the RRT path that was generated.