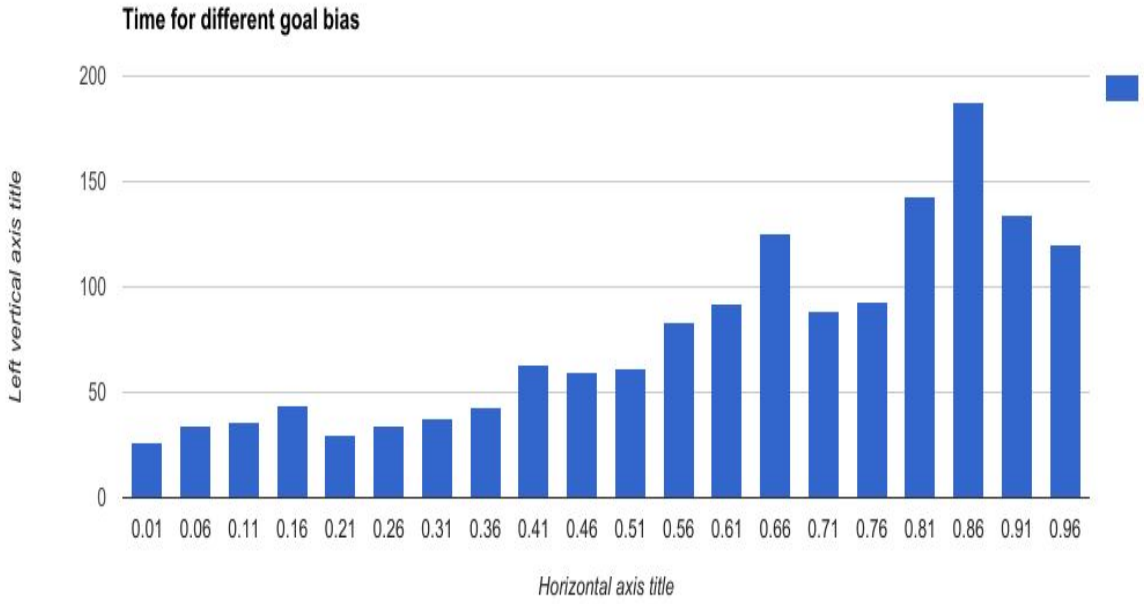
HW3 RBE 550 Motion Planning

1. The classes RRTNode.h and NodeTree.h are present in the rrtplugin/include directory

2. The nearest neighbor function is present in the rrtplugin/src/NodeTree.cpp file.

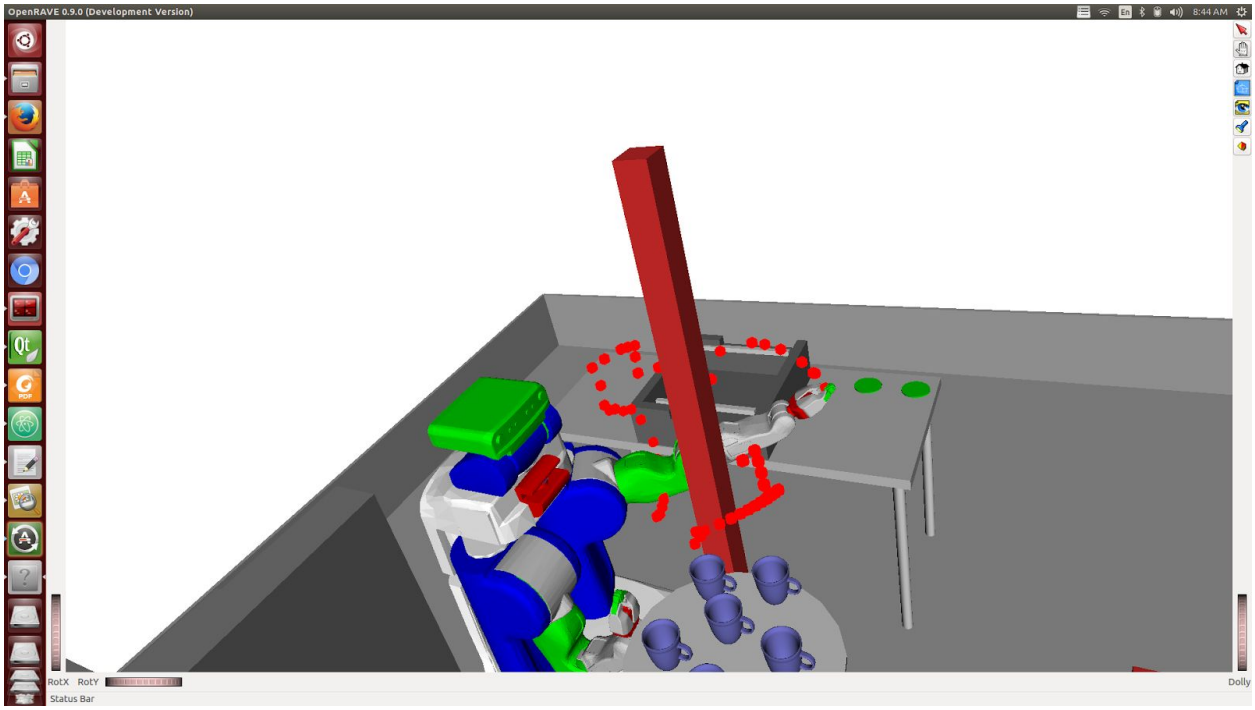
3.

a) The graph for average computation time vs goal bias is as follows:



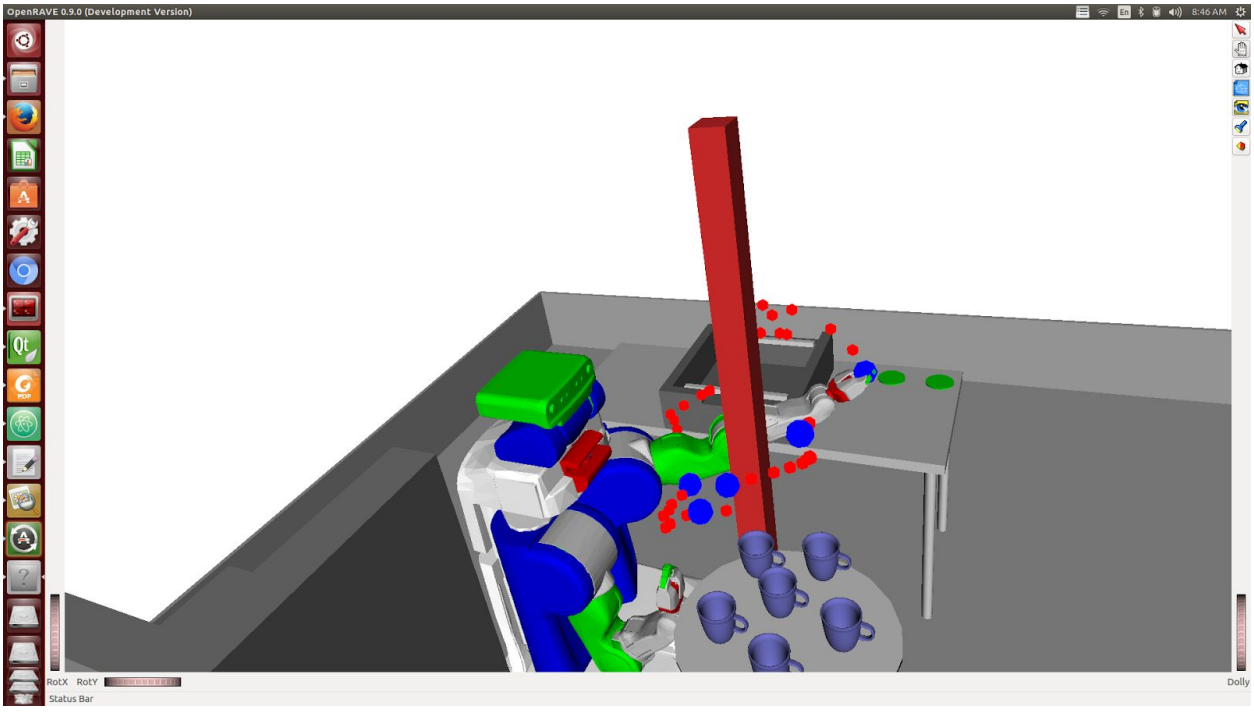
Explanation:
The pole(obstacle) is placed right beside the arm of the robot. For this reason, the goal bias does not work in our favour which increases the computation time and number of nodes explored. The algorithm works best for goal bias in the range of 0.01-0.36 **.** In the beginning the algorithm samples random points in the environment. Once it is past the pole (obstacle), a single random point chosen as goal point will solve the path. This explains why it works better for values of lower bias. We observe a peak at 86% bias.

b) Screenshot showing the path (also included in zip)



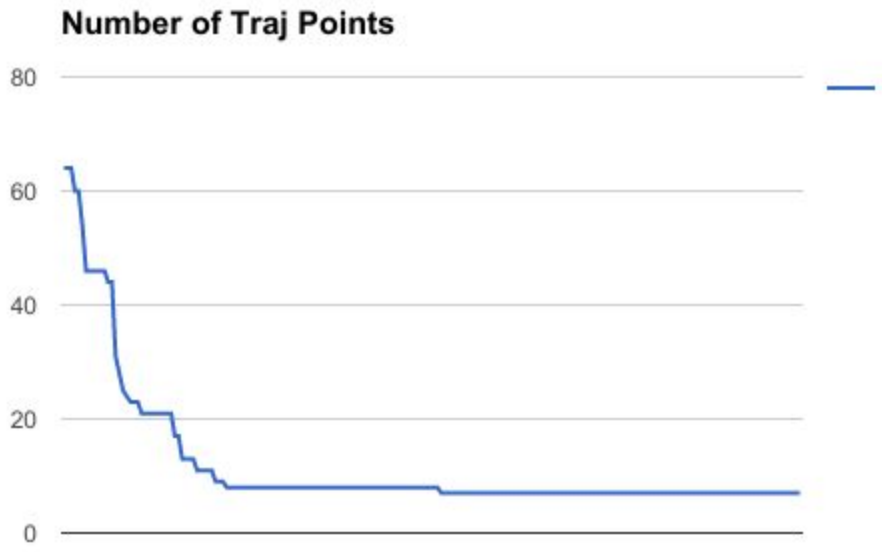
4.

a) Screenshot showing the path (also included in zip)



The smoothed path points are not close to each other because the average ratio of smoothed/original points is less than 10 % . For instance, for about 80 points, the smoothed trajectory contains less than 8 points.

b) Length of Path Vs the Number of Smoothing Iterations

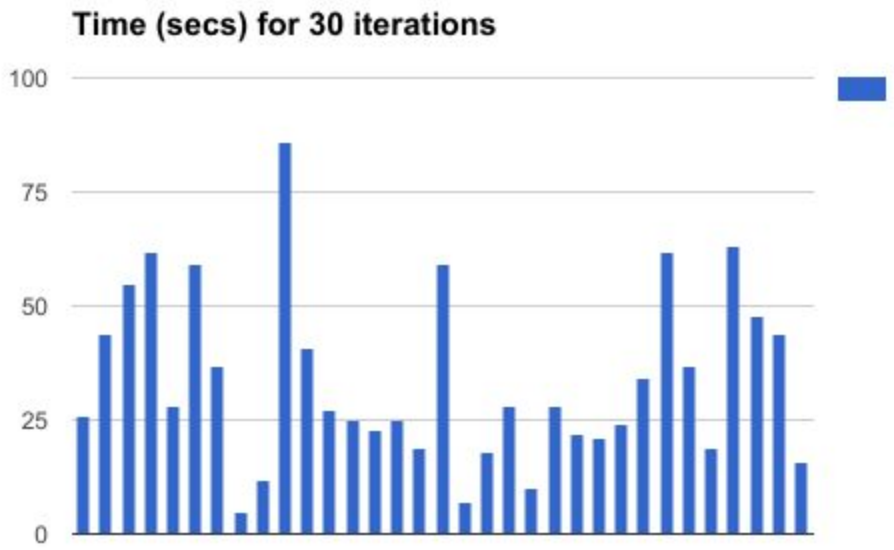


5. Analysis after running algorithm for 30 times

	RRT Time(s)	Smoothing Time (s)	Number of Nodes(#)	Path Unsmoothed(#)	Path Smoothed(#)
	26	2	1743	53	6
	44	1	2650	88	5
	55	2	3368	48	4
	62	2	2748	59	6
	28	2	1628	56	5
	59	3	2729	56	5
	37	3	2032	60	7
	5	2	1415	47	4
	12	1	810	49	4
	86	2	3300	55	6
	41	3	1700	58	7
	27	2	1663	60	7
	25	1	1714	59	6
	23	3	1084	63	9
	25	2	1482	63	6
	19	3	1191	56	8
	59	2	2695	56	6
	7	1	791	39	6
	18	2	1315	55	6
	28	3	1818	60	4
	10	2	719	59	5
	28	1	1166	67	4
	22	2	1142	53	5
	21	2	943	64	4
	24	3	1091	48	6
	34	2	2109	66	6
	62	1	2876	69	8

	37	2	1578	51	6
	19	3	1054	58	6
	63	2	2603	57	6
	48	3	1884	61	6
	44	1	2341	64	5
	16	2	1005	68	7
Average:	**33.76**	**2.06**	**1769.30**	**58.33**	**5.79**
Variance	**370.0643939**	**0.4962121212**	**566514.8428**	**69.85858586**	**1.547348485**

These values indicate that the RRT time and number of nodes in tree to find path vary largely because of the random nature of the algorithm. The smoothing time is almost a constant. The path length and reduced path length follow a fixed ratio of about 10:1 .



6. BiRRT
   a) The bidirectional RRT-Connect algorithm is not dependent on goal bias unlike the RRT-Connect which makes it a more generic approach to solve a motion planning

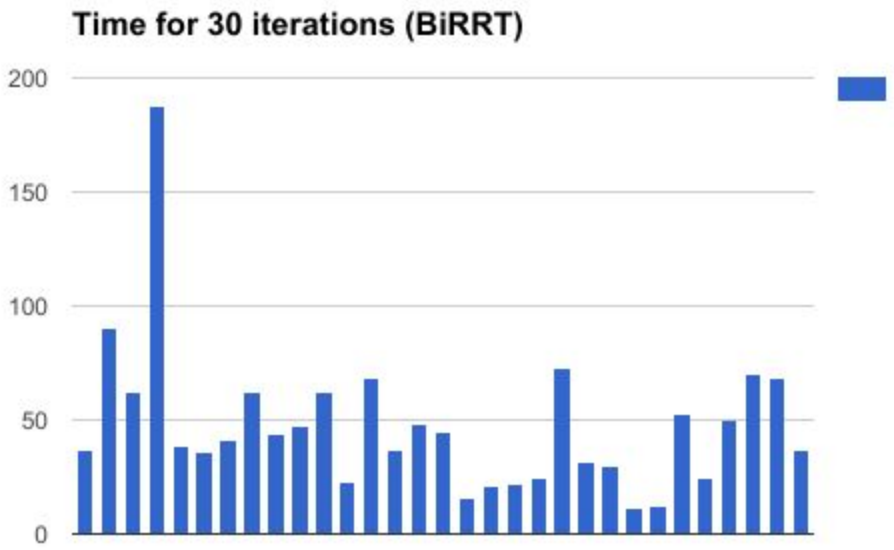
problem. The bidirectional RRT works smarter than RRT-connect because you to try to reach a common path from two different directions.

b)

BiRRT	**RRT Time(s)**	**Number of Nodes(#)**	**Path Unsmoothed(#)**
	37	1465	68
	90	2962	70
	62	2147	53
	188	5948	82
	39	1437	55
	36	1358	58
	41	1606	63
	62	2059	61
	44	2780	90
	47	4368	48
	62	2728	74
	23	1589	56
	68	2729	45
	37	3265	60
	48	1793	67
	45	1663	68
	16	764	57
	21	860	50
	22	919	56
	25	1063	60
	73	2525	58
	32	1173	67
	30	1256	62
	11	542	63
	12	573	63
	53	1945	60
	25	1041	58
	50	1818	61
	70	2310	59
	68	2207	55
	37	1392	67

Average	**47.5483871**	**1944.677419**	**61.74193548**
Variance	**1029.150884**	**1294307.292**	**83.86451613**

After analysing a run of 30 trials, it seems that the RRT-Connect with a goal bias of 20% out performed the Bi RRT which sounds conceptually stronger.



c) You would need to uncomment this line in the code:
#RRTPlugin.SendCommand('StartBiRRT'+' '+str(startconfig).translate(None, "[],")+' '+str(goalconfig).translate(None, "[],"))

Also, you could comment line to just run bidirectional RRT
path=RRTPlugin.SendCommand('StartRRT'+' '+str(startconfig).translate(None, "[],")+' '+str(goalconfig).translate(None, "[],") + " " + str(goal_bias).translate(None, "[],"))