CS460 Section 2
Assignment 2 - Sampling-based Motion Planning

Students: Laishek, Tso (lt359) Jeeho Ahn (ja709) Kanhang Li (kl621)

1 Part A

1.1

In SE(3) space, the configuration of the piano will be: q = [x, y, z, α, β, γ], where x, y, z are the coordinates and α, β, γ are the orientations Configuration space will be in 3 dimension. Obstacles will be decided by PQP collision checking library. obstacles and piano are represented in triangles according to the PQP library. Random samples(configurations) will be generated in the configuration space(3D space). And for each sample, we check whether the x, y, z position coordinates and euler angles (α, β, γ) are free from obstacles. We used the built in random function from python to generate random values for the samples uniformly accross the entire configuration space

1.2

Configuration space in 3D world would take both states and orientations into consideration. By using the following formulas. 3D Euclieadean function: d(p,q) = $\sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$ We will use this function to calculate the distance between 2 sample points p, q
If angular rotation is involved, we can use: quaternion Euler angles and rotational matrix. We belive that Euler angle has an idea of handling orientation described by three successive rotations (θ, ϕ, η) about certain sets of three axes (v1, v2, v3). Benefits of using Euler angles are compact with exactly three angles, numerically stable, and computationally efficient. Furthermore, it is intuitive because an object can rotate in three ways; roll, pitch, yaw.

1.3

Local planner decides rather there is a path between 2 sample points, construct a set of evenly spaced samples on a straightline between 2 configurations, and check all the intermediate samples are collision free. We increased resolution by dividing a path into 10 different samples. We used PQP collision checking on each samples. If one of the sample is indeed colliding with an obstacle, then we discard the path. We used Dijkstra's Algorithm to find shortest path.

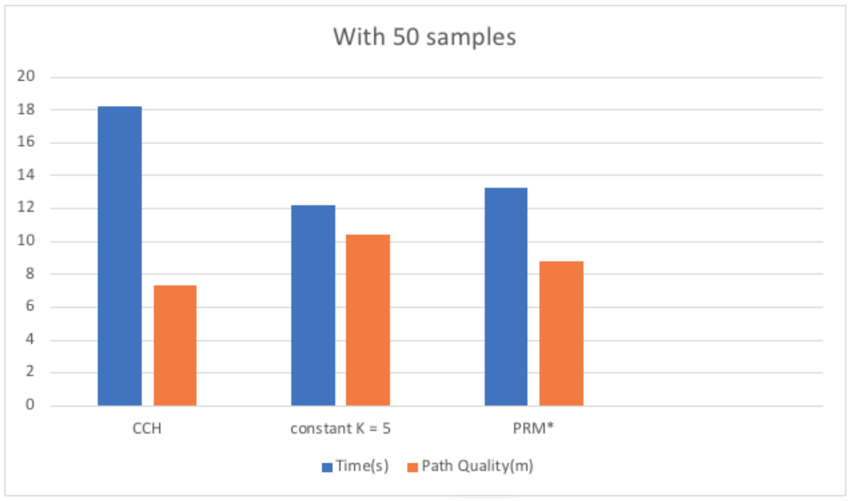
1.4

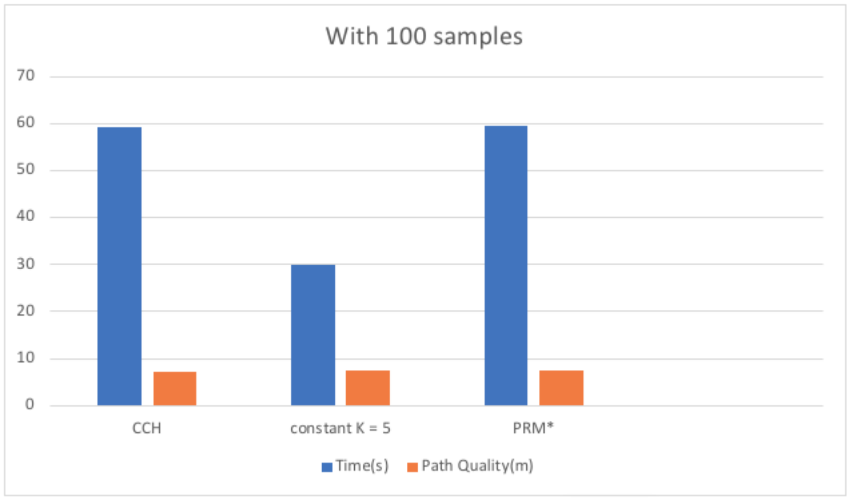
1. The connected components heuristic version: If a sample is visited we marked it visited, so there will be only one path to the sample.
2. The constant k version: We set the number of neighbors to a constant number and we will not visit any other neighbors once we reached k neighbors.
3. The PRM star version: We dynamically set limits of neigbors to k(n) $\geq$ e(1 + 1/d)log(n)

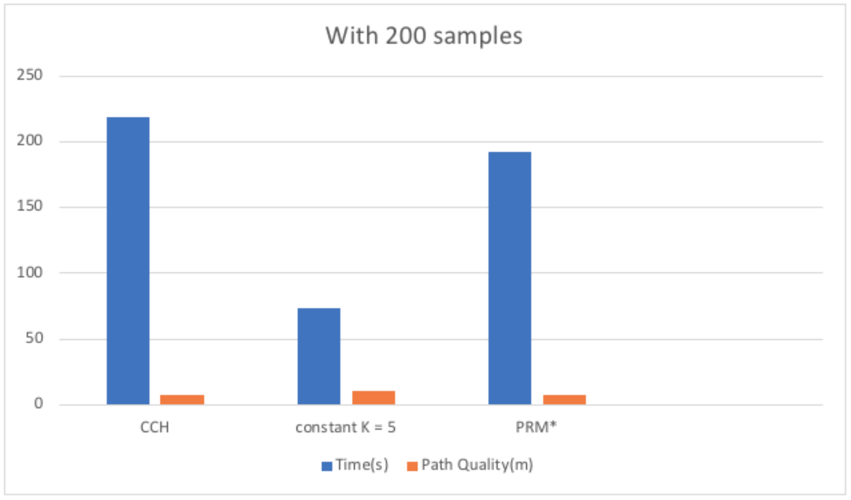
1.5

Done on demostration

1.6







We used the start configuration (5, 9, 0.4, 0, 0, 0), and goal configuration (6, 3, 0.4, 0, 0, 90). As we increase the number of samples from 50, 100, 200, the runtime for all 3 version are exponentially increasing. The paths are relatively equal, being around on average 7.2-7.4 m. But the constant K has a big advantage on runtime, while Connect Component Heuristic shows a slight advantage on the path quality.

1.7

Done on demostration

2 Part B

2.1

Done on demostration

2.2

Done on demostration

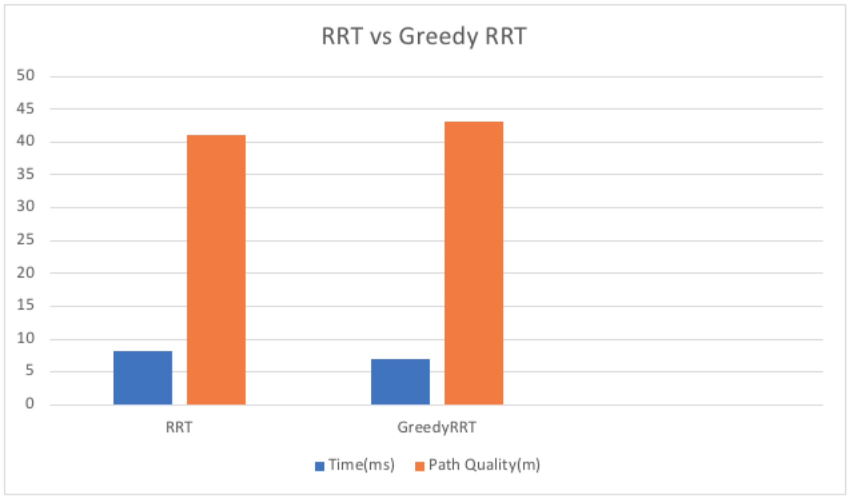
2.3

To achieve voronoi-bias in the state space with multi-dimensional configuration, the distance formula will involve the difference between the quaterion and distance of a state and its sample states. Since the object includes not only the state of the position but also the orientation, the choosing a distance function requires extra implementation to calculate the achieve our goal. To do so, we utilized weight to distinguish two different situations. When the space is narrow, we will have a higher weight value on the difference in quaterion, otherwise the difference in distance will always have a higher weight value and vise versa. In our map, the path is not narrow, so we will prioritize in the difference in distance. So when a sample with position that is farther away from the current state, it is more likely to be used to expand the tree.

2.4

Done on demostration

2.5



The start configuration is Top left corner, (-7.5, -6), and the goal configuration is bottom right corner, (9,5). For greedy RRT, we allow a 10% chance of branching toward the goal state. The greedy RRT is as expected faster than the regular. The distance difference is insignificant, with only 2m.