

vc68 Chung-Yu (Vincent) Chang Project 2

1. In this project, we aim to finish three collision checking functions for point, round and square robots given robot's x, y coordinate, orientation (for square one), side length (for square one) and rectangular obstacles' coordinate. Collision checking is essential for robot planning algorithms to verify whether certain points in the configuration space are valid in robot's working space
2.
 - A. Point Robot: a robot locating at (x, y) , no extra size in assumption.
The configuration space is $R^2 (x, y)$
 - B. Round Robot: a robot centered at (x, y) with radius r .
The configuration space is $R^2 (x, y)$
 - C. Square Robot: a robot centered at (x, y) with side length L and orientation θ .
The configuration space is $R^2 \times S^1 = SE(2) (x, y, \theta)$
3.
 - A. For a point robot, since the case is relatively simple, I just use the intuition to derive the formula for implementation.
 - B. For a round robot, At the beginning I forgot edge cases around the corner.
Directly pad obstacles would over-extend corners to square, causing areas near corner vertices between r and $\sqrt{2}r$ become invalid.
Separating the big square into three parts can solve the issue:
The extended width rectangle, the extended height rectangle, and 4 corners with Euclidean distances calculation.

- C. For a square robot, since both the robot and obstacles are convex polygons (rectangle), I look up and eventually find Separating Axis Theorem (SAT) to examine whether two convex polygons intersect with each other.

If two convex polygons didn't intersect with each other in the 2D space, we can find a line to separate them. i.e. we can find a projection axis that is perpendicular to the separating line, and the projected segment of two segments shouldn't overlap with each other. The candidate projection axes are normal vectors of edges of the two polygons.

In this case, the candidate projection lines are 8 normal vectors from 8 edges including the robot's edges and the obstacles edges.
Algorithm steps:

- i. Perform rotation on the origin and perform translation to get robot's coordinates.

O(1)

- ii. Calculate 4 robot's edges and get 4 normal by those edges
 $(-(y_2-y_1), x_2-x_1)$

O(1)

- iii. Calculate 4 normal vectors of obstacle 4 edges for every obstacle

O(E)

- iv. For every obstacle, we iterate all 8 candidate normal vectors (4 robot + 4 obstacle) and perform vertices projection to see whether the segments overlap with each other to find one axis to separate the robot and every obstacle.

If we do, break and head to next obstacle.

If we don't, return the status invalid.

O(E)

After iterating all obstacle and no intersection is detected, return the status is valid.

O(E) in total, E equals to number of the obstacles

- D. I passed all the testcases including the optional one and didn't encounter numerical precision issue.
4. 1+2: 15 mins
3: 3 hours

The most difficult part of this assignment is the loop arrangement in the square robot part. Correct arrangement when implementing SAT (loop by obstacle – normal vectors – vertices) enables the escape once a single axis is found, which can save lots of time. It took me a while to figure out how to arrange them.