**MECH 550**

**Project 2**

Peiguang Wang, Sichao Zhang

1. A succinct statement of the problem that you solved.

Problem 1: Check whether a point(random location) is inside the given obstacles.

Problem 2: Check whether a circle(random location and size) is inside or intersect with the given obstacles.

Problem 3: Check whether a square(random location, size and rotation angle) is inside or intersect with the given obstacles.

2. A short description of the robots (their geometry) and their configuration spaces.

Problem 1: The point robot. A point has two parameters, its x-axis coordinate(x) and y-axis coordinate(y). x and y defines its location in the workspace. Its configuration space is the same as the workspace.

Problem 2: The circle robot. A circle has three parameters, the x-axis(x) and y-axis(y) coordinates of its center and the radius. x and y defines its location in the workspace and radius defines its size. Its configuration space is the same as the workspace.

Problem 3: The square robot. A square has four parameters, the x-axis(x) and y-axis(y) coordinates of its center, the length of its edge SideLength and the rotation angle with respect to its center theta. x and y defines its location in the workspace, SideLength defines its size and theta defines its rotation state. Its configuration space is the workspace after transformation( translation(x,y) and rotation(theta(CCL))).

3. 1)A summary of your experiences in implementing the different collision checkers.

Collision checker 1 and 2

2)Were there any cases that were particularly easy/difficult?

Yes, Problem 3 is much more difficult than the other two and it is harder than what we have thought. The square robot has four parameters to define its shape, location and rotation in the space. And this problem is more complicated because we need to consider different cases, to solve each case we need to consider both rotation and translation of the square, while doing other two, considering translation of the robot is just enough. As the rotation in the work place is based on the origin of the work place, so when we to try to deal with the rotation angel theta, we have to transform the coordinates of every point of robots and obstacles. While doing this we meet many corner cases. And finally it takes us seven hours to finish the codes and pass the test but we still have one case not included in.

3)Did you run into any numerical precision issues or other similar complications?

No.

4)Does your implementation accurately classify the given test sets?

Yes.

5)How did you debug your implementation?

We output the parameters of the mistake robots( point, circle and square), and visualized the given obstacles and them in the MATLAB to see and try to explain why these robots were not fit with the codes. If we could not see obvious problems in the output graphics or codes, then we wrote debug codes to see which case appeared to went wrong. If it still did not work, we output every parameters of the object and obstacles, which we used functions to calculate, to see whether they were wrong.

6)Does your code accurately classify all of the optional test sets?

Yes.

4. 1)Rate the difficulty of each exercise on a scale of 1–10 (1 being trivial, 10 being impossible).

Problem 1: 4

Problem 2: 4

Problem 3: 7

2)Give an estimate of how many hours you spent on each exercise, and detail what was the hardest part of the assignment.

Problem 1: 3 hours

Problem 2: half an hour

Problem 3: 7 hours

The hardest part of the assignment is to solve the problem 3. We considered two cases(case1: one or more corner points of the square are in collision with the obstacles and case 2: one or more corner points of the obstacles are in collision with the square) to cover all the possible situations to solve this problem at first time. However, there turned out a lot of mistakes when we finished the codes. Then we realized that the rotation in the work place was with respect to the origin point of the work place but not the center of the square. So it took us some time to do the calculation and then we wrote a rotation function that enabled a random point in work place can rotate around a given point in a given angle. However, we still met a lot of corner cases and in the end we had one case(case 3: the edges of the square and obstacles intersect but with no corner point within each other) that was not included in the codes although our collision checker passed the test.