ME 570: Robot Motion Planning

Homework 1 Report

By Cameron Cipriano

09/14/2021

Question 1.1 \_code\_: Polygon.plot()

It has been a while since I have coded in python using NumPy and I don’t have too much experience with matplotlib, so this question forced me to get back into these packages and the language itself.

A picture containing shape

Description automatically generatedShape, rectangle

Description automatically generated with medium confidence

Figure 1.1: Filled Polygon Figure 1.2: Hollow Polygon

Question 1.1 \_optional\_: Polygon.is\_filled()

For this method, I found a particular formula that roughly calculates the area underneath a line segment, which equates to its average height times its horizontal length . Summing each of these areas for each edge, we will either obtain a positive or negative value corresponding to the orientation of the polygon. Negative values indicate counter-clockwise orientation while positive indicate clockwise.

Question 1.1 \_report\_: edge\_angle description

Question 1.2 \_code\_: Edge.is\_collision()

This method makes use of the orientation of a 3-point line segment to determine whether the two line segments being compared are intersecting.

Defined in me570\_geometry.py is a function called ‘orientation’ which determines if a line segment defined by 3 points is completely straight (collinear) or curling in the clockwise/counterclockwise direction using the midpoint as the joint. The formula I used to calculate this is derived below:

Set equal to determine relationship:

From this, we do not need them to necessarily be equal, rather let the subtraction become to determine the result

­­

Question 1.3 \_code\_: Polygon.is\_self\_occcluded()

The most difficult part of this was understanding how to make use of the angle() function to determine what was inside the occlusion zone and what was not and also be orientation agnostic. Involved a lot of test cases.

Question 1.4 \_code\_: Polygon.is\_visible()

This function was not as challenging as the self-occlusion check as it was mostly about putting existing code together. The check made use of the self\_occluded function and a custom function I made in order to test if the vertex-point segment was intersecting among any of the polygon’s edges.

Question 1.2 \_report\_: polygon\_is\_visible\_test

Chart

Description automatically generatedChart

Description automatically generated

Figure 1: Solid polygon1 Figure 2: Hollow polygon1

Chart

Description automatically generatedChart

Description automatically generated

Figure 3: Solid polygon2 Figure 4: Hollow polygon2

Question 1.5 \_code\_: Polygon.is\_collision

I do not feel I had too much difficulty with this method as it was simply making use of the is\_visible() function in.

Question 1.3 \_report\_: polygon\_is\_collision\_test

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Figure 1: Solid Polygon1 Collisions Figure 2: Hollow Polygon1 Collisions

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Figure 1: Solid Polygon2 Collisions Figure 2: Hollow Polygon2 Collisions

**Problem 2: Poor-man’s Priority Queue**

Question 2.1 \_code\_: Priority.insert()

Question 2.2 \_code\_: Priority.min\_extract()

Question 2.3 \_code\_: Priority.is\_member()

Question 2.1 \_report\_: priority\_test

Question 2.2 \_report\_: Priority Queue