Motion Planning RBE 550

Project 1: Fundamentals

DUE: Thursday, September 7th at (8:30 am). Submit your answers as a PDF to Canvas.

All the written components should be typesetted with LaTeX. A template is provided on Overleaf. However, feel free to use your own format as long as it is in LaTeX.

When you finish the assignment, create a zip file that includes both the written report and the code upload them in Canvas. If any extra libraries are required, clearly document how they can be installed with a README file.

Present your work and your work only. You must *explain* all of your answers. Answers without explanation will be given no credit. **This assignment must be completed individually and not in pairs.**

1. **(30 points)** Let $\mathcal{A}$ be a unit disc centered at the origin in a workspace $\mathcal{W} = \mathbb{R}^2$. Assume that $\mathcal{A}$ is represented by a single algebraic primitive $H = \{ (x,y) \mid x^2 + y^2 \leq 1 \}$. Show that if this primitive is rotated about the origin that the transformed primitive is unchanged. This can be shown by showing that any point within the transformed primitive H' must be within H, and vice versa.

2. **(40 points)** You are given the endpoints to two line segments, A_1B_1 and A_2B_2, in a 2D workspace. The line segments include their endpoints. Provide an algorithm in pseudocode to compute the intersection points of these two line segments, if one exists. Be careful and consider all corner cases. Your algorithm must provide the correct output with every input. Besides providing the pseudocode you must also clearly explain it. **Hint:** There are many ways to represent line segments. Choosing wisely will allow for a shorter and more efficient implementation.

3. **(30 points + 30 Bonus Points)**

   Now imagine that you are given a set of line segments $S = S_1, S_2, S_3$ and you want to compute all the intersection points of all of these segments.

   - **(20 points)** Fill the `segment_pair_intersection()` function of the file `intersections.py` based on the the algorithm you wrote in Question 2 inside. If you run the `main.py` function it should plot 3 test cases with the intersection points marked with a red x. These test cases look for intersections between 2 different set of lines, one colored blue and one purple. For this question you don't need to provide an explanation but only include the three generated plots with the red x in the correct intersection spots.
   - **(10 points)** After you implement `segment_pair_intersection()` run again the `main.py` function which besides the three test cases, also plots the time it takes to compute all the intresection points for a different number of segments. What is the complexity in big O notation, of the already implemented `all_pairs_intersections()` function based on this plot? N in this case would be the total number of line segments.

- (**30 Bonus points**) Implement a more efficient algorithm `efficient_intersections()` in file `intersections.py` that outperforms the naive `all_pairs_intersections()`. One of the most efficient algorithms that exists is the Sweep Line Algorithm but anything you implement that is even marginally more efficient than `all_pairs_intersections()` will receive at least partial credit. You must explain in the written report your reasoning.