1 get_minkowski_sum

1.1 Complexity

Let n be the number of vertices in the input polygon and let m be the number of vertices of the robot.

First, we make sure that the input correspond with the criterias of the C-Space sonstruction algorithm.

1. Forming the arrays is $O(m + n)$ (adding one point at a time)
2. Checking that the vertices are organized are ccw is $O(n)$ (multiplying consecutive pair of edges)
3. reversing the order is $O(n)$ (adding one point for each point)
4. Finding the minimum y index as the first one and reorganizing the polygon accordingly is $O(n)$ (checking each point and adding points at the correct order)

Next, the loop over the points. In each iteration we promote at least one index for the polygon vertices (of $n + 2$ vertices) and one for the robot vertices (of $m + 2$ vertices). We have maximum of $n + m + 4 = O(n + m)$ iteration. In each iteration:

1. Updating values - $O(1)$
2. Adding vertex to the output - $O(1)$
3. Calculating angles - $O(1)$
4. Updating indices - $O(1)$

In total - $\boldsymbol{O(m + n)}$.

1.2 Why is the obstacle has to be convex?

For a convex polygon, in each iteration, each angle's value doesn't decrese (the angle increases or remains the same). If we have a non-convex polygon, for some polygon index k, the angle of the obstacle decreses and therefore, the index for the robot might increase. Once the index increases, there's no way back, so the direction vector is set to be the one of the next robot's vertex even though the angle for the $k + 1$ polygon index is lower.

For example, for a non-convex polygon and a squared robot, for $i = 2$ and $j = 3$ the C-Space obstacle vertex is placed inside the workspace obstacle and therefore doesn't even contain the workspace obstacle.