16-811 Assignment 6

Emma Benjaminson

21 November 2019

1 Problem 1

References:
(1) Sommer, Pascal. "A gentle introduction to the convex hull problem." `https://medium.com/@pascal.sommer.ch/a-gentle-introduction-to-the-convex-hull-problem-62dfcabee90` Visited 11/10/2019.

(2) Wikipedia. "Graham scan." `https://en.wikipedia.org/wiki/Graham_scan` Visited 11/20/2019.

Please see the code contained in code/q1.m.

Several figures are included below with appropriate captions displaying the solution to the convex hull problem. I display results for a range of points counting between 10 and 100,000.

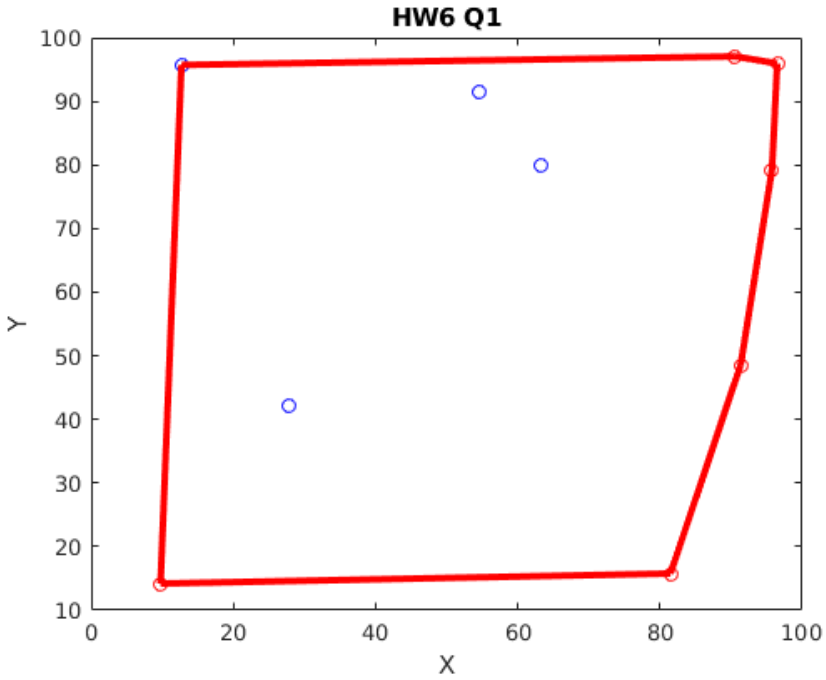

Figure 1: Plot of convex hull algorithm over n = 10 randomly located points. Convex hull result is shown in red, points are shown in blue.

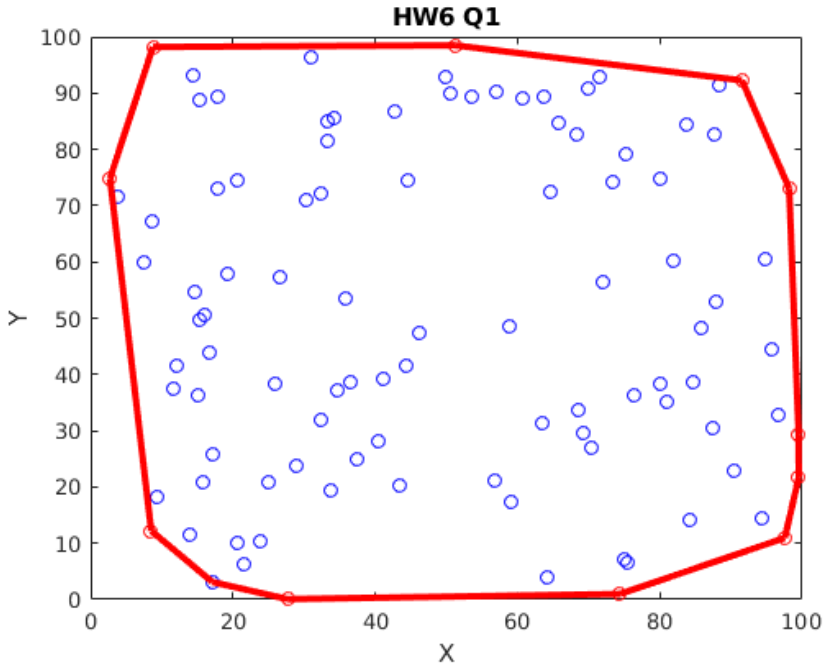

Figure 2: Plot of convex hull algorithm over n = 100 randomly located points. Convex hull result is shown in red, points are shown in blue.

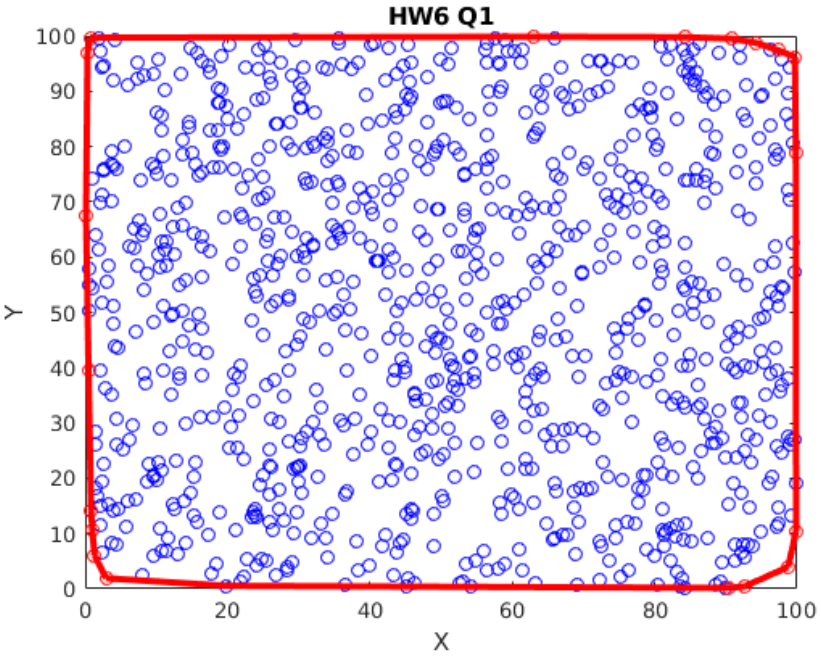

Figure 3: Plot of convex hull algorithm over n = 1000 randomly located points. Convex hull result is shown in red, points are shown in blue.

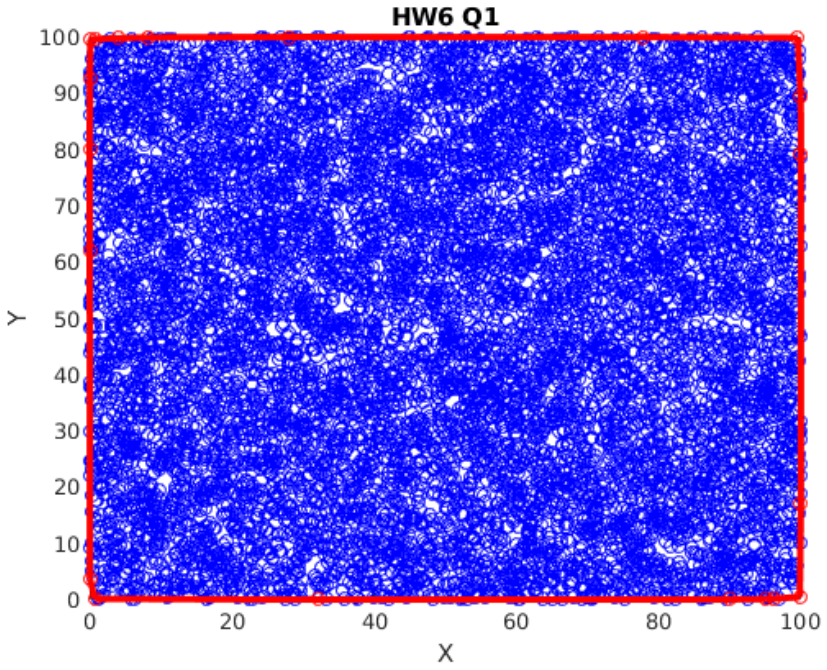

Figure 4: Plot of convex hull algorithm over n = 10000 randomly located points. Convex hull result is shown in red, points are shown in blue.

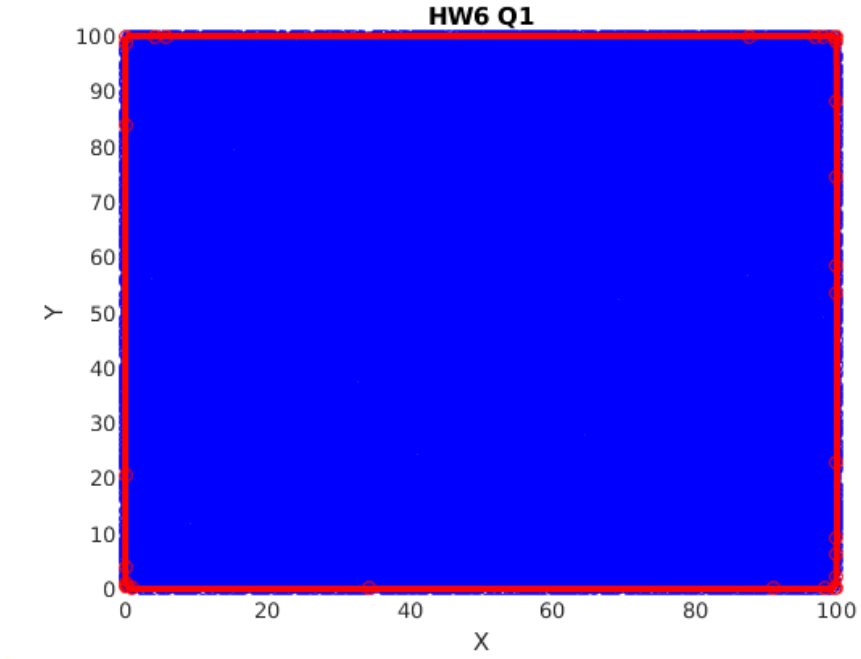

Figure 5: Plot of convex hull algorithm over n = 100000 randomly located points. Convex hull result is shown in red, points are shown in blue.

2 Problem 2

References:
(1) Stack Overflow. "How can I check if two segments intersect?" `https://stackoverflow.com/questions/3838329/how-can-i-check-if-two-segments-intersect` Visited 11/19/2019.

(2) Koneru, Haarika. "Visibility graphs." Kent University. `http://www.cs.kent.edu/~dragan/ST-Spring2016/visibility%20graphs.pdf` Visited 11/19/2019.

(3) Joshi, Vaidehi. "Finding the Shortest Path, with a Little Help from Dijkstra." `https://medium.com/basecs/finding-the-shortest-path-with-a-little-help-from-dijkstra-6` Visited 11/20/2019.

(4) Ably, Thaddeus et al. "Dijkstra's Shortest Path Algorithm." `https://brilliant.org/wiki/dijkstras-short-path-finder/` Visited 11/20/2019.

Please see the code contained in code/q2.m.

Please note in the figures for this problem that the start and goal positions

are indicated by the red circles. The shortest path found is shown in red. The obstacles are drawn as polygons in different colors. Please note that I used the built-in MATLAB function "polyshape()" to draw the obstacles, but I did not use the function anywhere else to generate or store the obstacle information - the rest of the data management and processing of the obstacle data was done by me. The code is written to end the script and return an error informing the user if the start or end point is located inside a polygon.

I used my convex hull algorithm from Problem 1 to generate the obstacles. I created visibility graphs to identify all the possible piece-wise linear paths that the robot could take through the environment from the start to the goal location (the graph itself is hidden in the final figures presented here to make the results easier to see). To help generate the correct visibility graph, I wrote functions that could identify when two lines intersected, when a point was inside or outside a convex polygon, and when a line segment was located inside or outside a convex polygon. Finally, I wrote an implementation of Dijkstra's algorithm to find the shortest series of nodes in the visibility graph that led to the goal location.

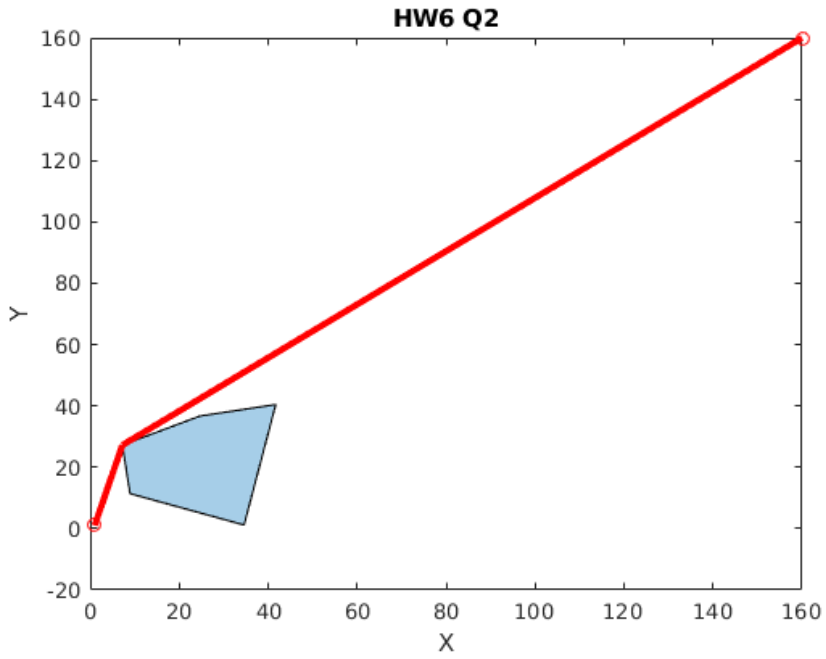

Figure 6: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works initially for a single convex obstacle. The shortest path is drawn in red. The start and goal locations are shown as red circles.

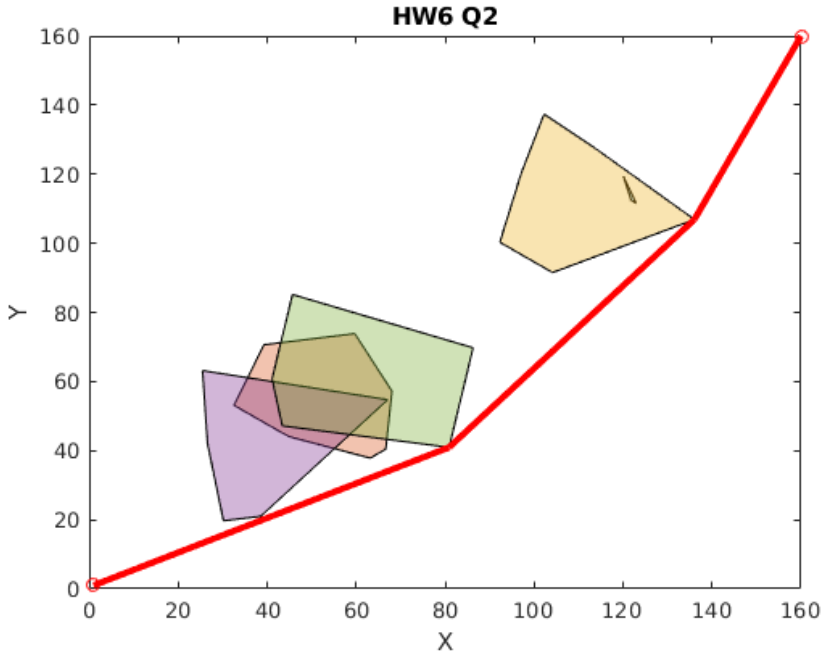

Figure 7: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works for 5 overlapping convex obstacles. The shortest path is drawn in red. The start and goal locations are shown as red circles.

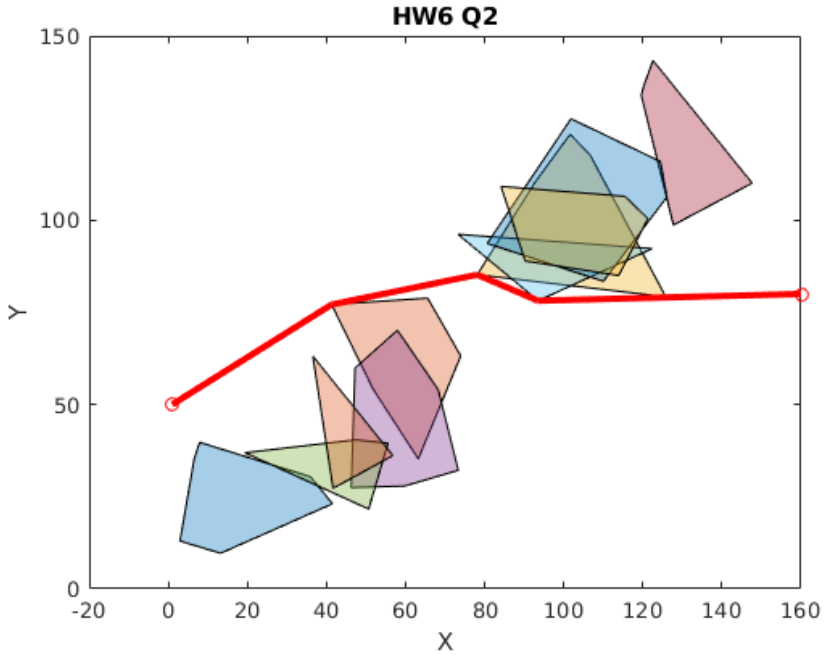

Figure 8: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works for 10 overlapping convex obstacles. I have also shifted the start and goal locations to show the algorithm is robust. The shortest path is drawn in red. The start and goal locations are shown as red circles.

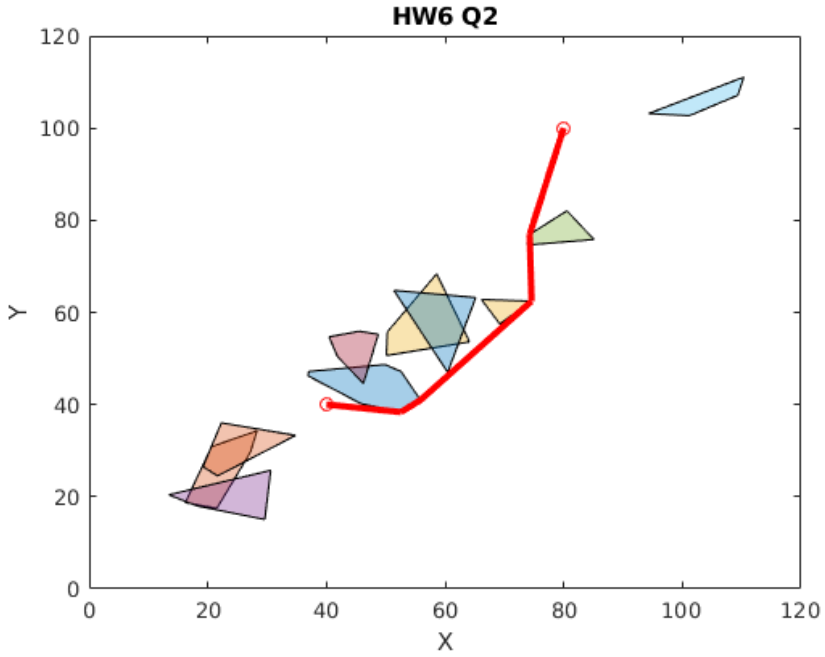

Figure 9: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works for 10 overlapping convex obstacles. I have again shifted the start and goal locations to further demonstrate that the algorithm is robust. The shortest path is drawn in red. The start and goal locations are shown as red circles.

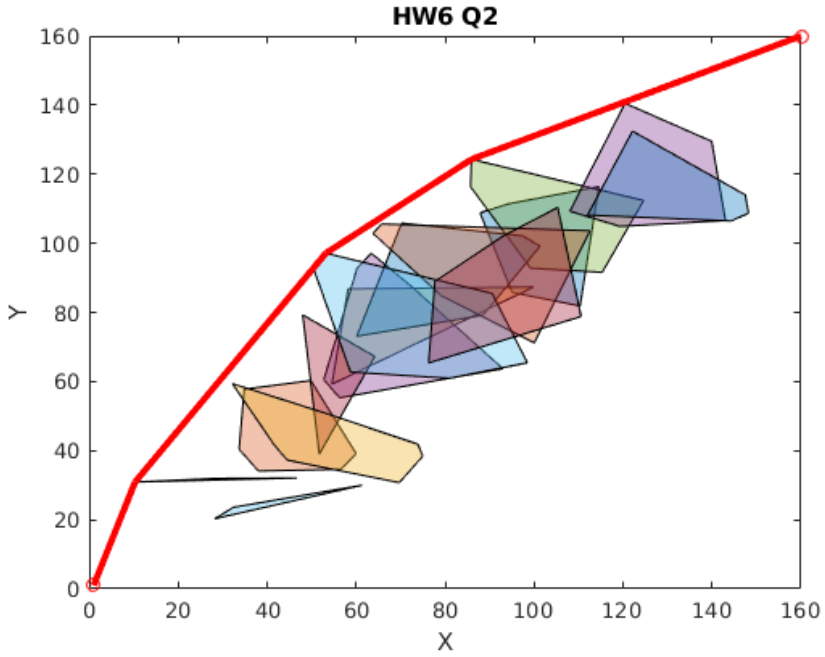

Figure 10: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works for increasing numbers of overlapping convex obstacles, in this case n = 15. The shortest path is drawn in red. The start and goal locations are shown as red circles.

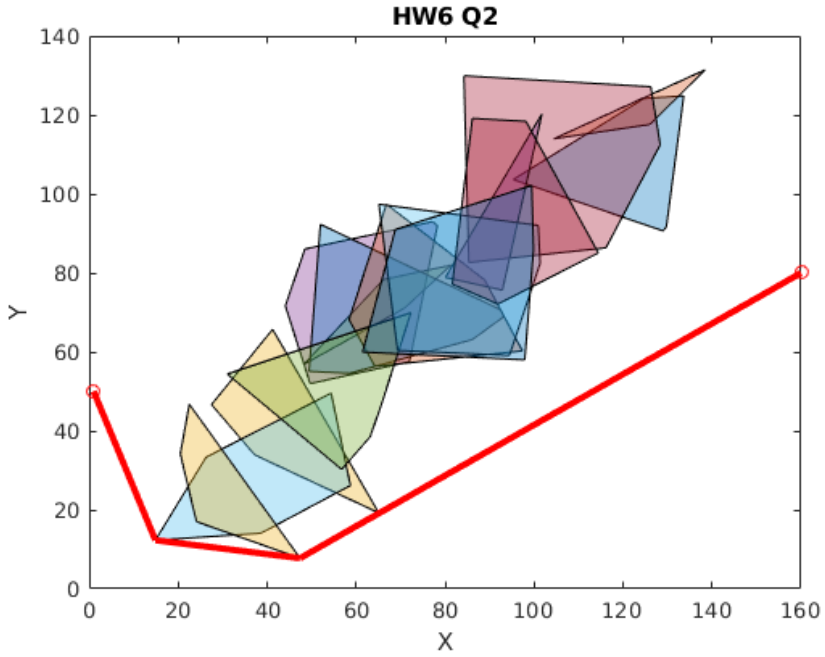

Figure 11: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works for increasing numbers of overlapping convex obstacles, in this case n = 15. Again, to demonstrate robustness I have shifted the start and goal locations. The shortest path is drawn in red. The start and goal locations are shown as red circles.

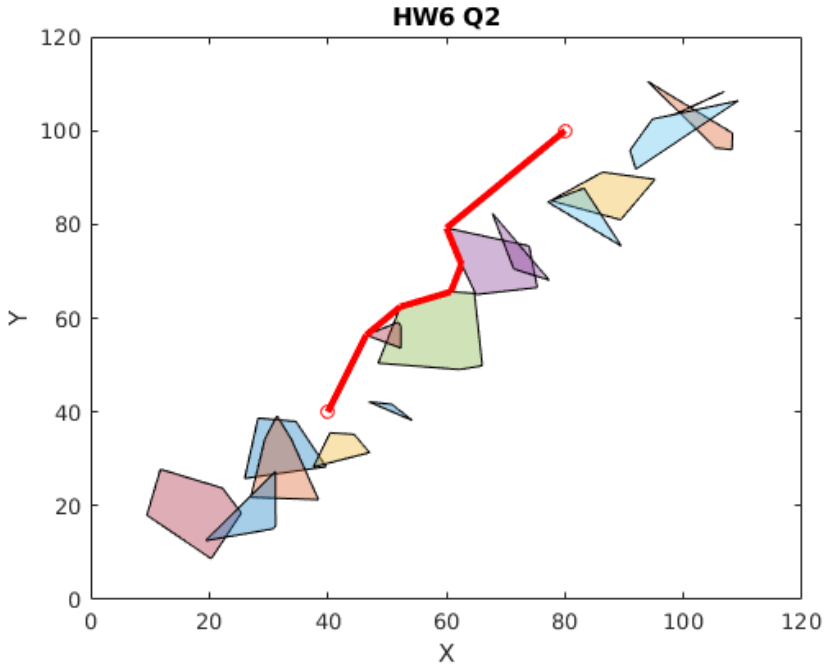

Figure 12: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works for increasing numbers of overlapping convex obstacles, in this case n = 15. Again, to demonstrate robustness I have shifted the start and goal locations. The shortest path is drawn in red. The start and goal locations are shown as red circles.

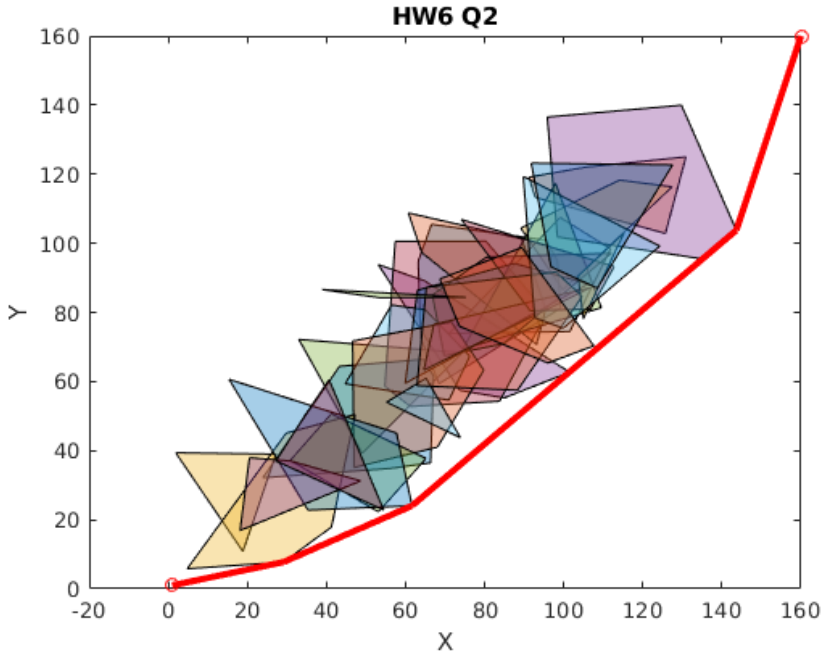

Figure 13: Solution to finding shortest path around 2D convex obstacles for a point robot. In this figure I demonstrate that the algorithm works for increasing numbers of overlapping convex obstacles, in this case n = 30. Again, to demonstrate robustness I have shifted the start and goal locations. The shortest path is drawn in red. The start and goal locations are shown as red circles.

3 Problem 3

References:
(1) Allen, Peter K. "Robot Path Planning." `http://www.cs.columbia.edu/~allen/F17/NOTES/lozanogrown.pdf` Visited 11/20/2019.

Please see the code contained in code/q3.m.

Please note in the figures for this problem that the start and goal positions are indicated by the red circles. The shortest path is shown in red. The obstacles are drawn as "grown" obstacles with their original and augmented shapes overlaid in different tones of the same color. The obstacles' vertices are also displayed as circles in different colors for different obstacles. The robot is shown as a small convex polygon at the start position; please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs. My code generated the robot's body as either a triangle or a rectangle, to be chosen at random during

program execution. The code is written to end the script and return an error informing the user if the start or end point is located inside a polygon.

I used the code previously described for Problems 1 and 2, and added a function that reflected the robot's body about the x and y axes, added the body to the vertices of the obstacles, and redrew the obstacles as larger convex polygons that accommodated these new vertices.

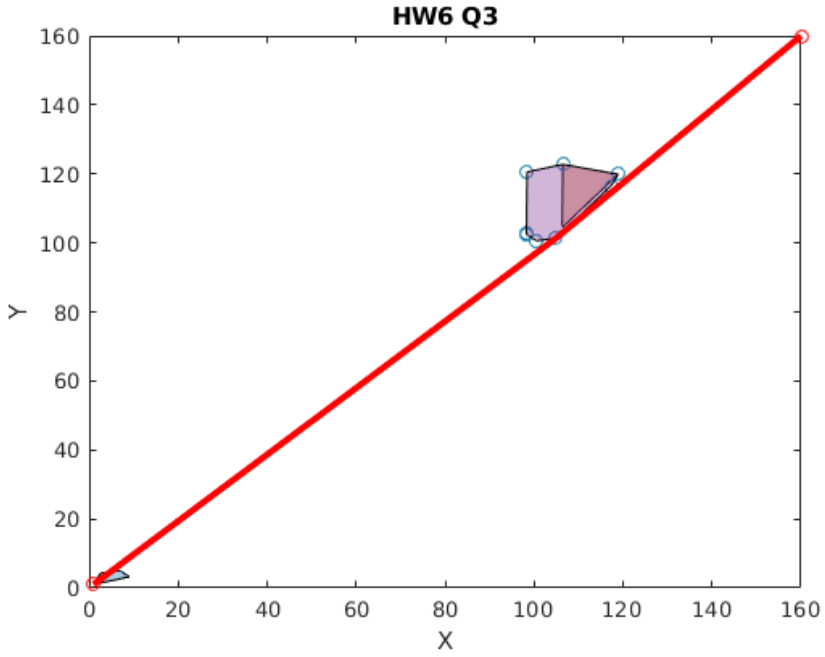

Figure 14: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate that the algorithm works initially for a single convex obstacle. The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.

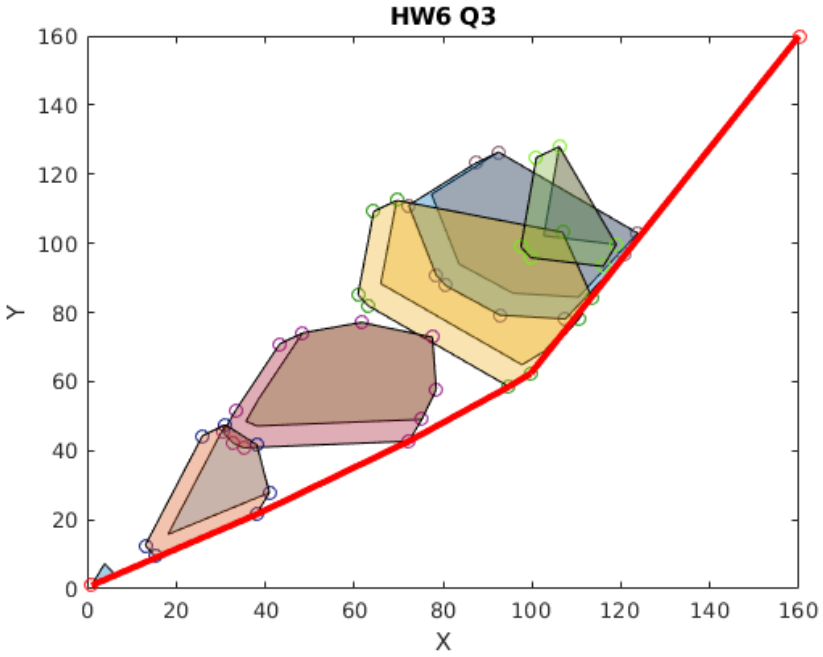

Figure 15: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate that the algorithm works for 5 overlapping convex obstacles. The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.

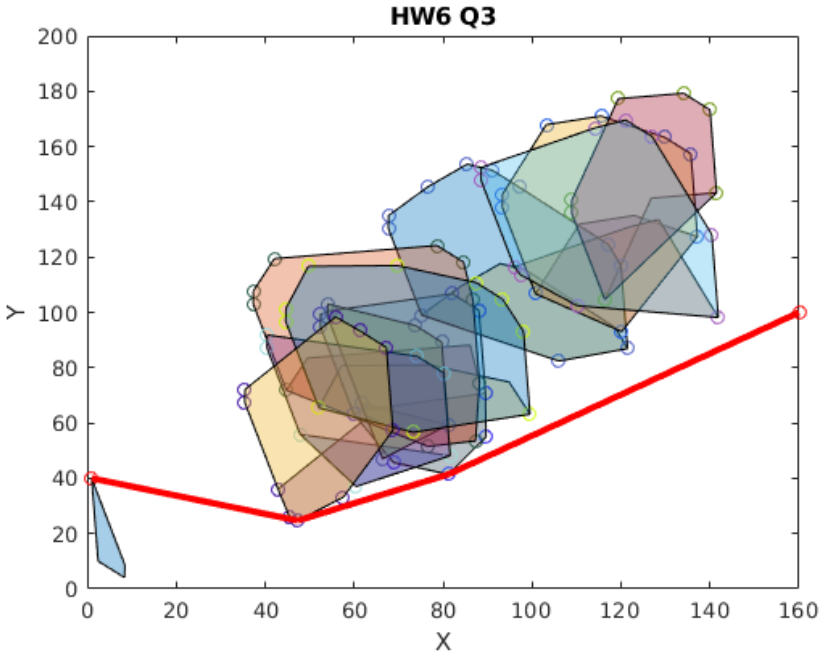

Figure 16: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate that the algorithm works for 10 overlapping convex obstacles. I also shifted the start and goal locations to show the algorithm is robust. The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.

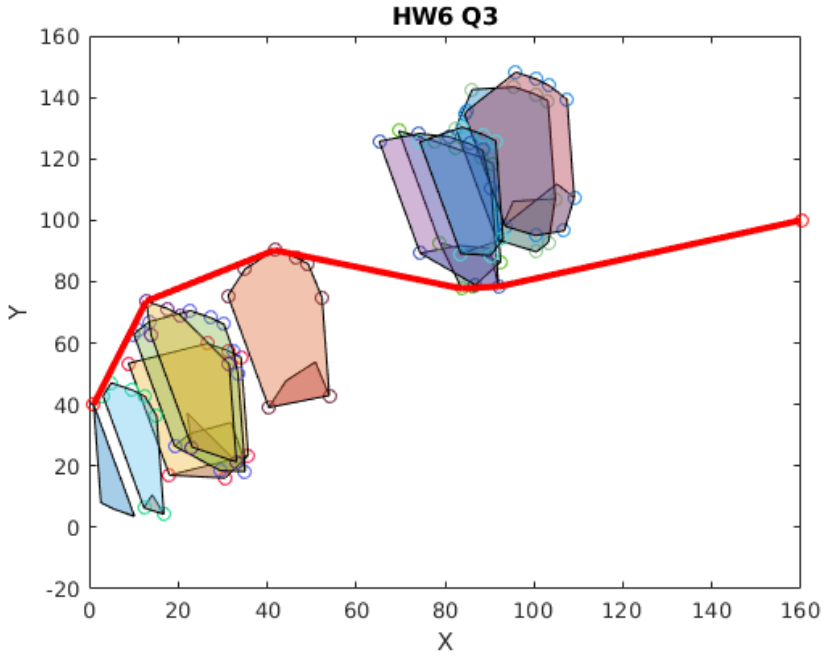

Figure 17: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate again that the algorithm works for 10 overlapping convex obstacles. The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.

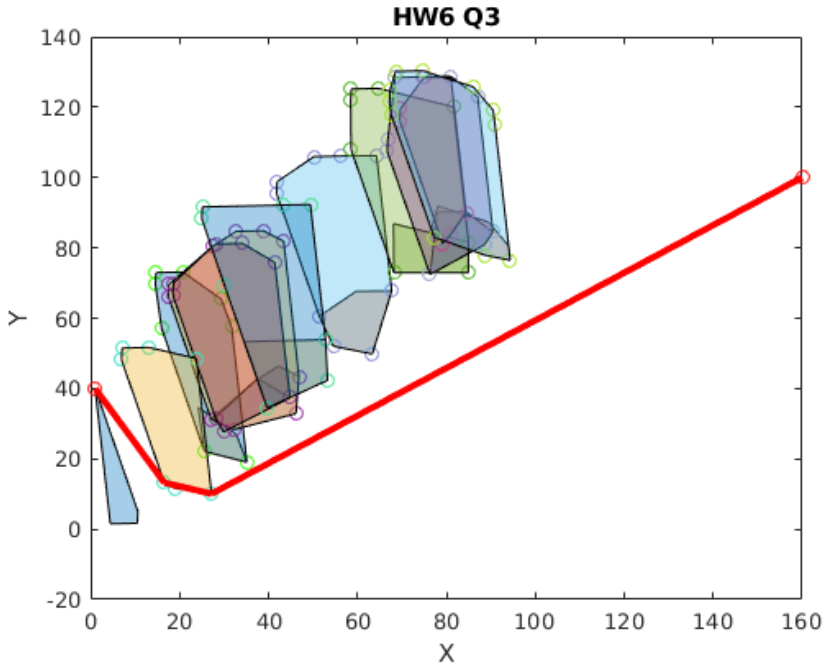

Figure 18: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate again that the algorithm works for 10 overlapping convex obstacles, and for a rectangular robot (most previous demonstrations had a triangular robot). The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.

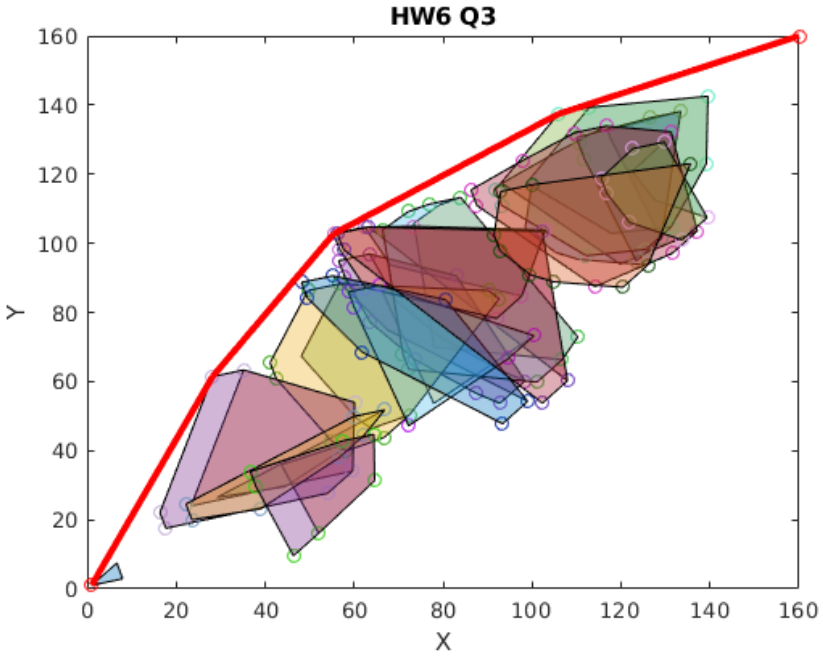

Figure 19: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate that the algorithm works for increasing numbers of overlapping convex obstacles, in this case for n = 15 obstacles. The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.

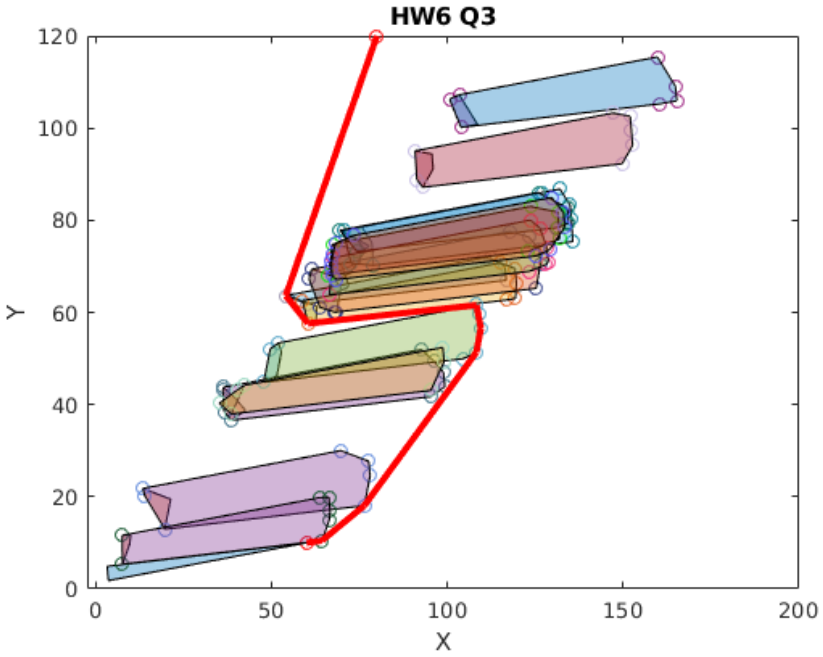

Figure 20: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate again that the algorithm works for increasing numbers of overlapping convex obstacles, in this case for n = 15 obstacles. The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.

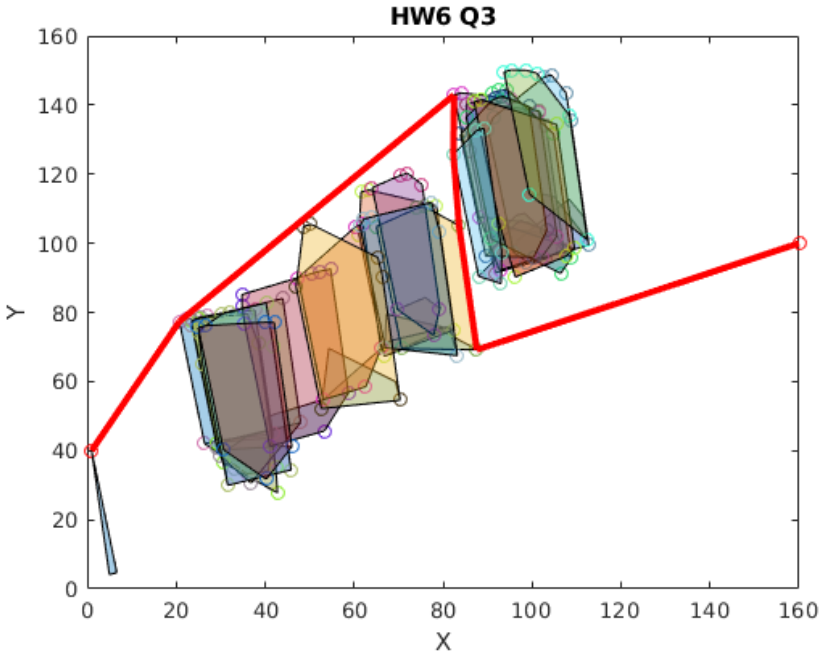

Figure 21: Solution to finding shortest path around 2D convex obstacles for a polygon robot. In this figure I demonstrate that the algorithm works for increasing numbers of overlapping convex obstacles, in this case for n = 20 obstacles. The two-tonal color of the obstacle shows its shape before and after it was grown to show the inaccessible space due to the robot's shape. The shortest path is drawn in red. The start and goal locations are shown as red circles. Please note that the robot's body is not considered as an obstacle so the path may traverse through the robot's body as shown at the start position during some runs.