Assignment 6

Math Fundamentals for Robotics 16-811, Fall 2018
DUE: Thursday, November 29, 2018

[redacted] (Andrew ID: [redacted])

For this homework, the code is complemented using python 3.5, with package sys, collections, numpy and matplotlib.

1. Implement two-dimensional (2D) convex hull.

   SOLUTION: This part is implemented in script `q1.py`.

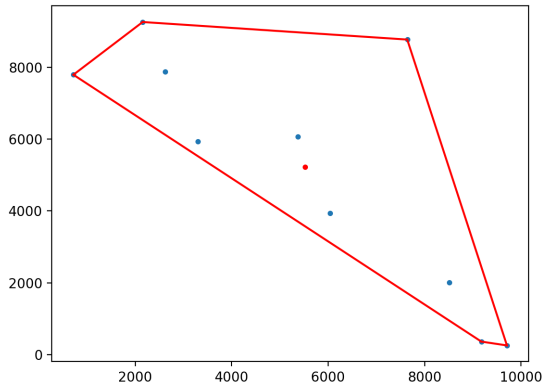
   In this script, the default method to get the points is to generate the points randomly. With random method, the total number of points and the maximum range of the coordinates can be specified in the main function. In such a method, just run the script with command

   ```
   python q1.py
   ```

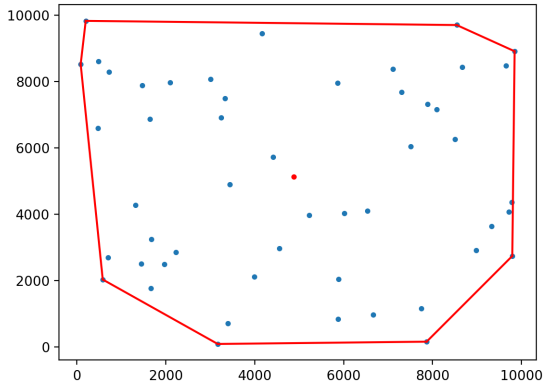
   You can also provide a list of points in the text file, just as shown in the included file `points.txt`. Each line in the text file contains two digits, the first one is the x coordinate of the point, and the second one is the y coordinate. With this method, run the script with command

   ```
   python q1.py points.txt
   ```

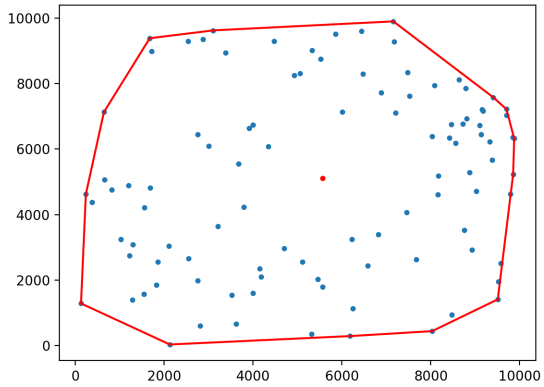
   Fig.1 are some examples of the result with different totall number of points.



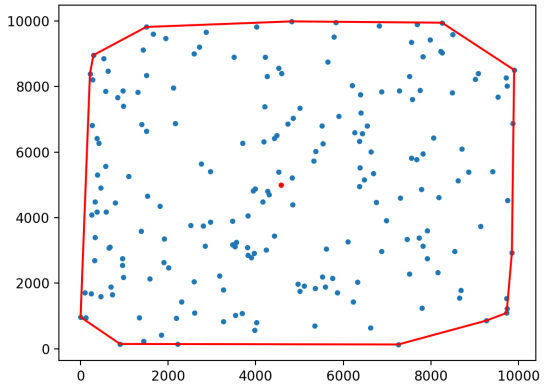
(a) 10 points



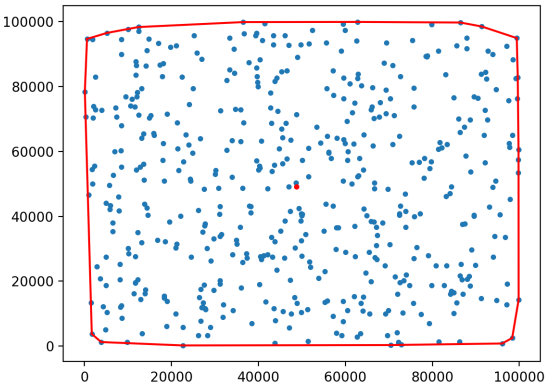
(b) 50 points



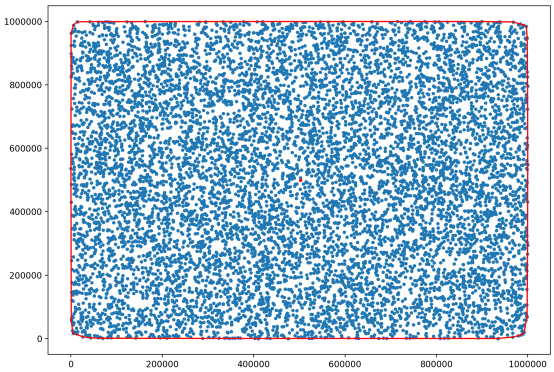
(c) 100 points



(d) 200 points



(e) 500 points



(f) 10000 points

Figure 1: Some examples of the two-dimensional convex hull implementation.

2. Implement an algorithm for finding shortest paths in 2D polygonal environments.

   SOLUTION: This part is implemented in script `q2.py`.

   In this script, all the input parameters are read from a text file with format as in file `polygon.txt`. This file contains information of a collection of nondegenerate convex polygons, a start location, and a goal location. The first line of the file starts with key word "start: " and then follows by a start location with format (x, y) where x is the horizontal coordinate and y is the vertical coordinate. The second line of the file starts with key word "end: " and follows by a goal location. In the following lines, each line represent a polygon, which is consisted by a set of vertices that are separated by commas. To see the format details, please refer to the file `polygon.txt`.

   To run the code, please run the command: `python q2.py polygon.txt`

   If there is no path from the start location to the goal location, then the script would print out the information indicating that there is no path; Otherwise, the script would output a list of points that consist of the entire path from the start location and the goal location, and it would also print out the minimum distance of the path.

   Fig.2 shows some examples. The red lines consist of polygons. The green lines are the all candidate paths. The blue lines form the final path with minimum distance.

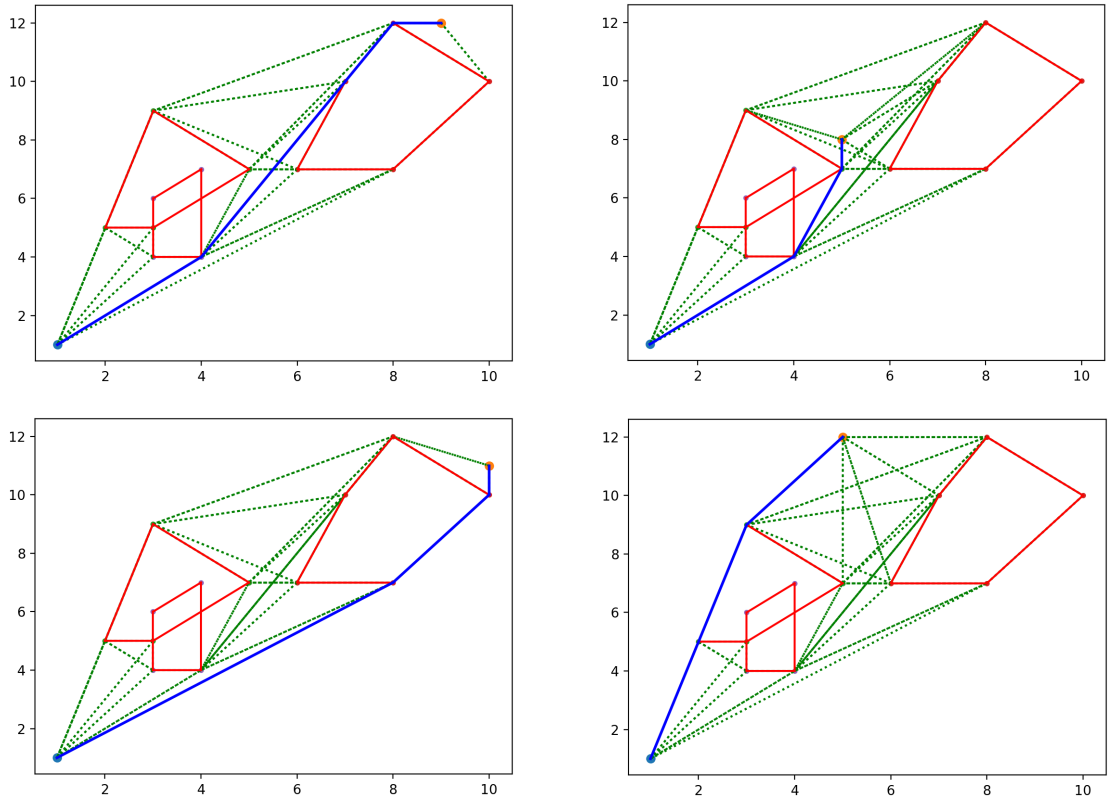


Figure 2: Some examples of finding the shortest path.

3. Combine your code for parts (1) and (2), along with some additional code, to implement shortest-path motion planning for convex polygons in two dimensions (translations only, no rotations)

   SOLUTION: This part is implemented in script `q3.py`.

   In this script, all the input parameters are read from a text file with format as in file `robot.txt`. This file contains information of a collection of a "robot" and its environment, along with start and goal configurations for the robot. The first line of the file starts with the key word "start: " and then follows by a set of points that form the robot at the start location. The second line of the file starts with key word "end: " and follows by a set points that form robot at the goal location. In the following lines, each line represent a set of points, from which we can find the convex hull to form a polygon. To see the format details, please refer to the file `robot.txt`.

   To run the code, please run the command: `python q3.py robot.txt`

   If there is no path for the robot, the code would print out the information indicating that there is no such a path for the robot to go from the start location to the end location; Otherwise, the code would output a list of points that form the entire path for the robot, and the minimum distance also would be given.

   Fig.3, Fig.4, Fig.5 show some examples. In these pictures, the red dotted lines are the initial polygons that computed from the given points. The red solid lines form the polygons that a particular point on the robot would generate when sliding along obstacle boundaries. The green dotted lines are the all candidate paths. The blue solid lines form the final path with minimum distance for the robot going from the start location to the goal location.

   In Fig.3 and Fig.4, the robot is a triangle. The configurations of these output are in file `robot.txt` and `robot2.txt` respectively.

   In Fig.5, the robot is a pentagon. The configuration of the output is in file `robot3.txt`.

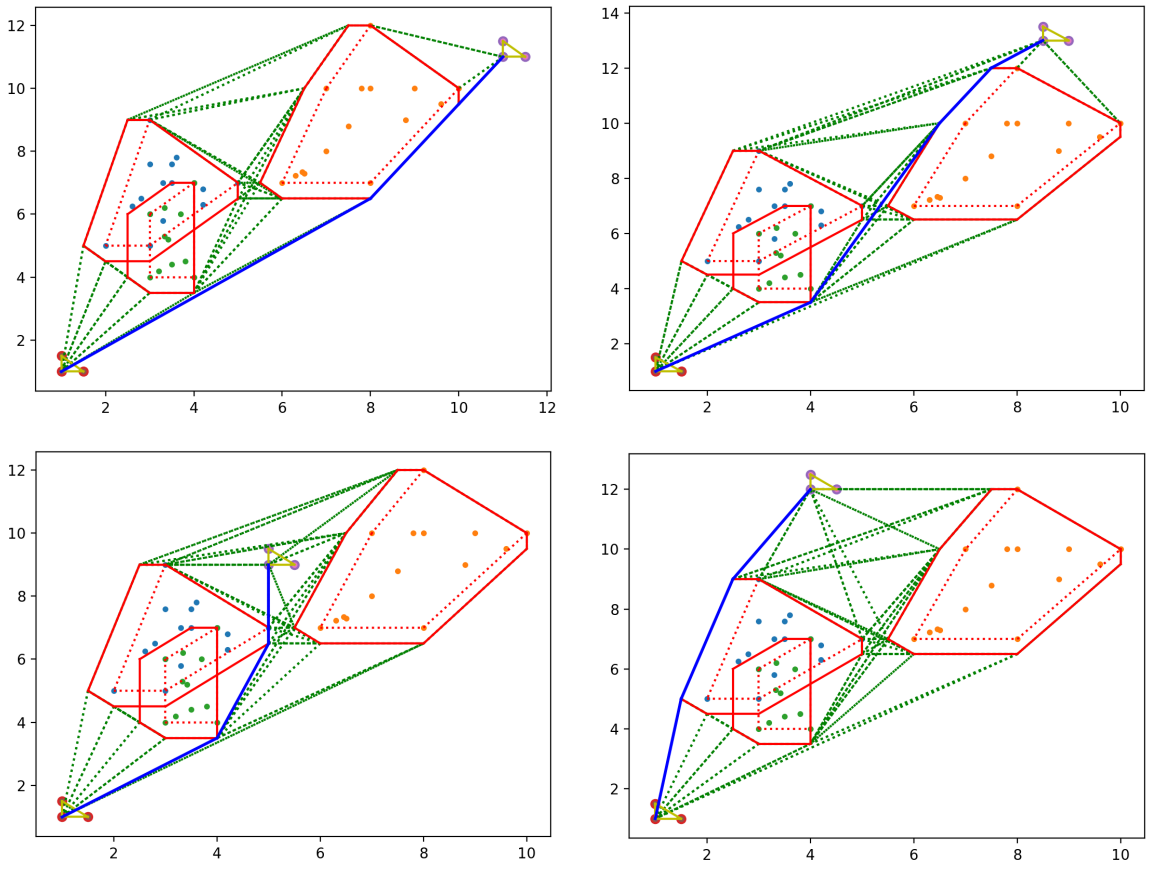


Figure 3: Some examples of finding the shortest path.

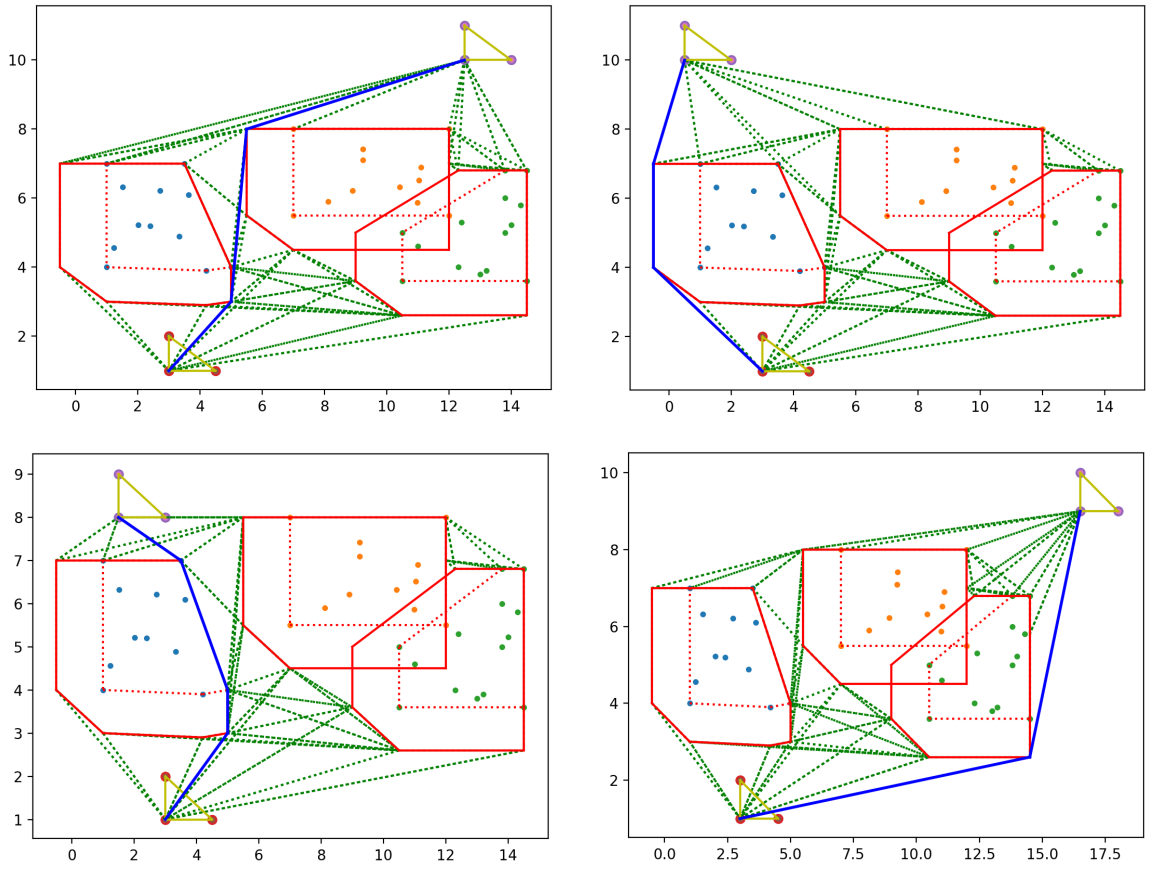


Figure 4: Some examples of finding the shortest path.

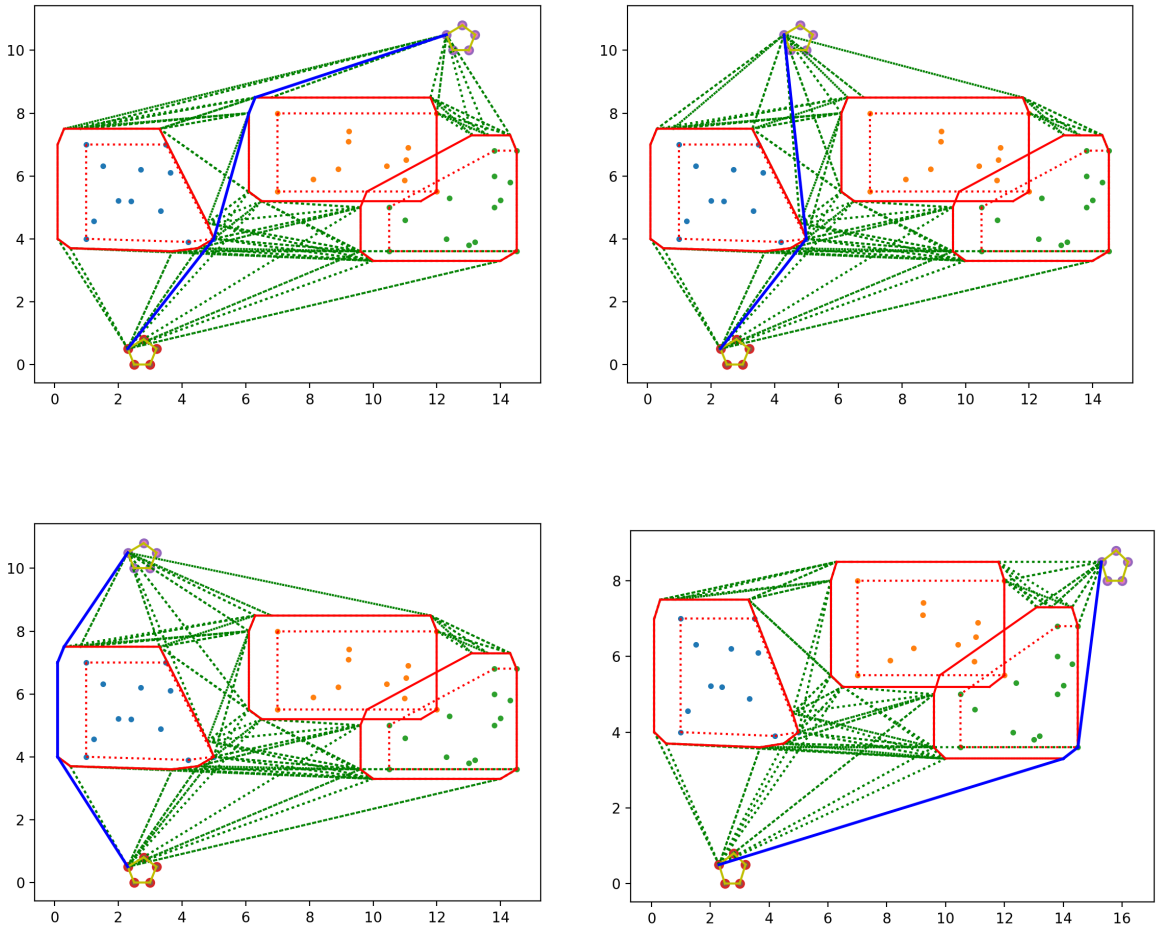


Figure 5: Some examples of finding the shortest path.