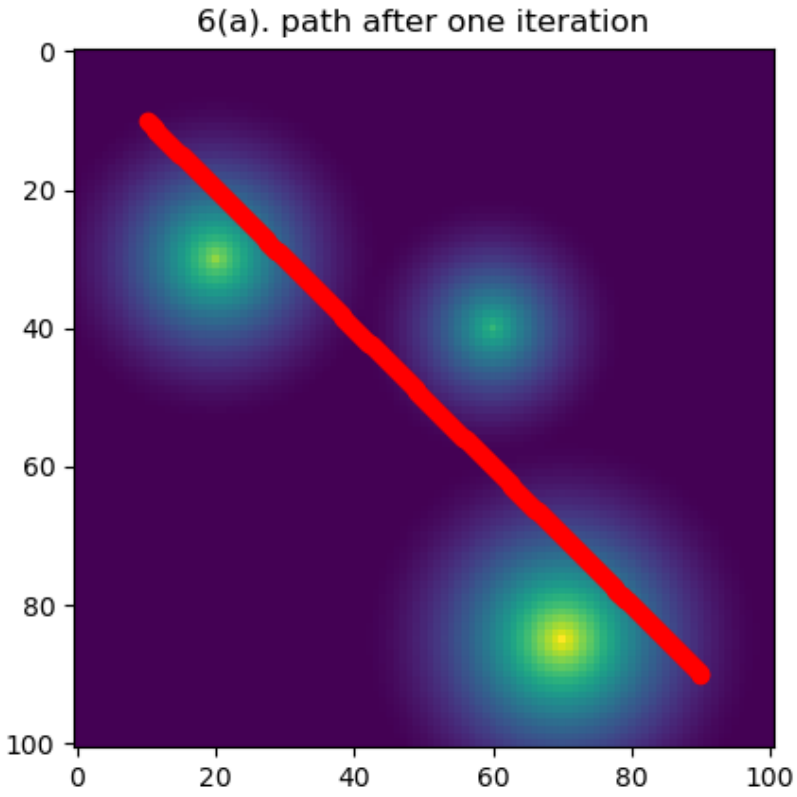
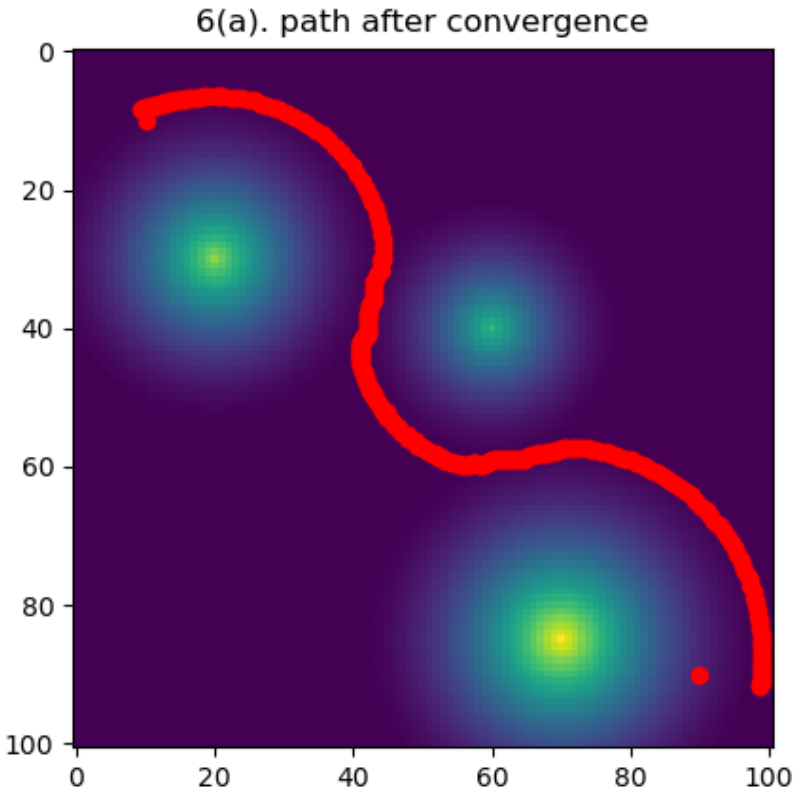
16811 A4

November 2022

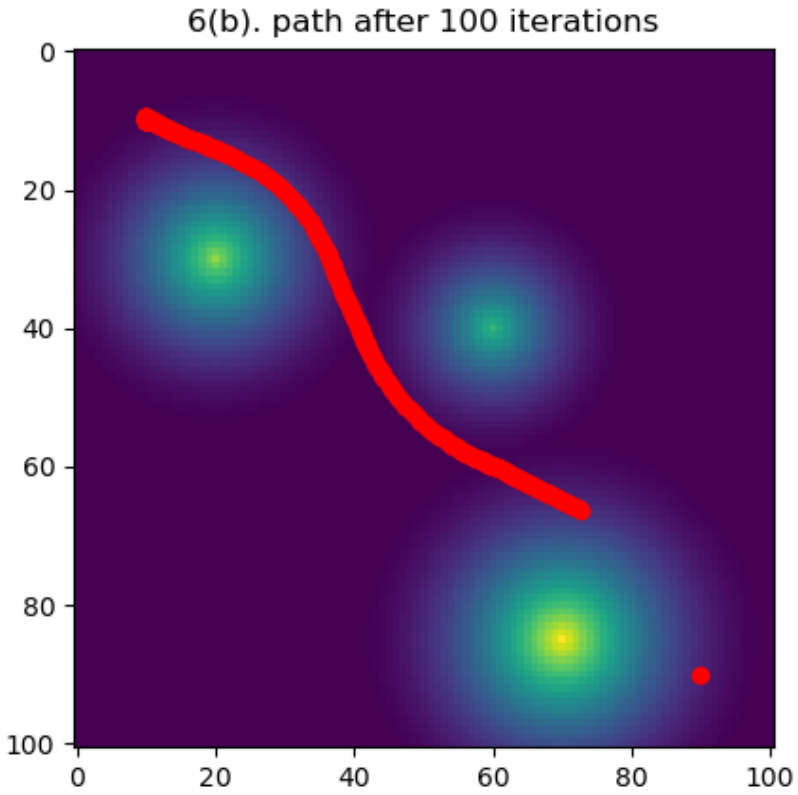
6.

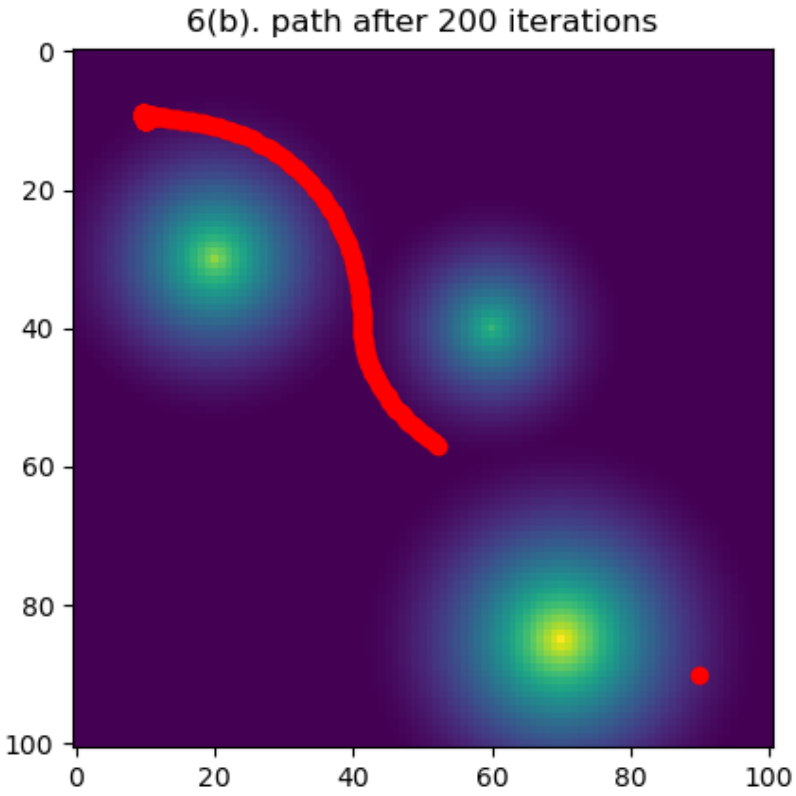
(a) Run *python q6.py* to get the plots of paths.
The first plot below is the path after one iteration. As shown on the 2D plot, we can see the path is moving away from the obstacles.
The second plot below is the path after convergence. I chose 'when the difference of the previous and current point norms less than 0.01' as the convergence criteria, since the if the difference between adjacent updated values is very small, it is likely to be near convergence.
As we can see from the second plot, the optimized path start and end points deviates from the start and end points. This is because our method considers it as independent 2D points when doing optimization, without considering it as a continuous path. Thus the path with locally optimal cost deviate from its desired start and goal after convergence.

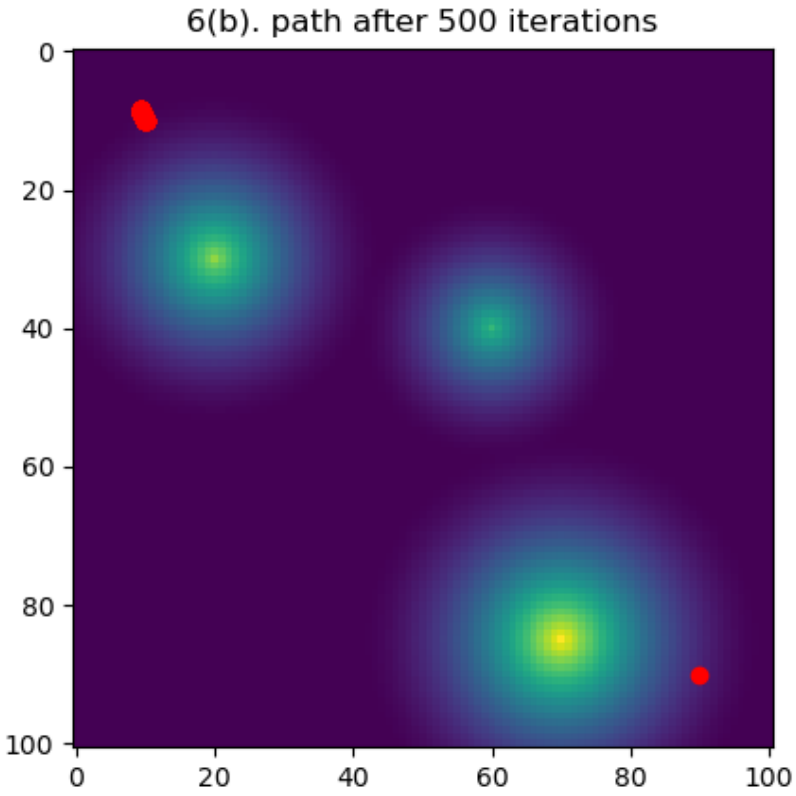



6(a). path after convergence
0
20
40
60
80
100
0
20
40
60
80
100

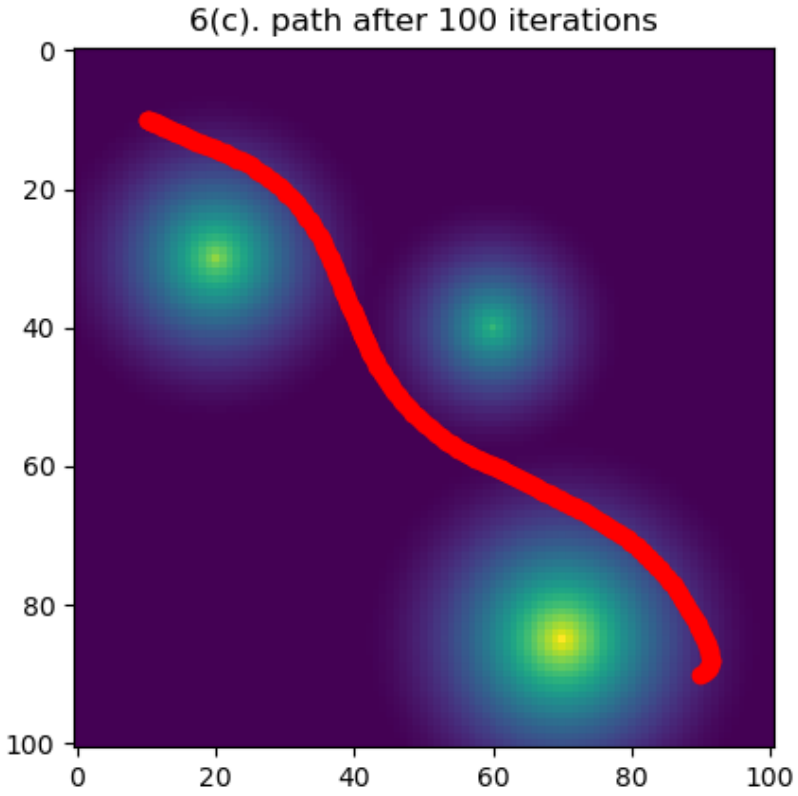
(b) Run *python q6.py* to get the plots of paths.
The plots below show the path after 100, 200, 500 iterations.
As we can see from the plots, the points on the optimized path gets more and more stacked at the positions near the start point, as the total number of iterations increases.
This is because in the augmented gradient updates, we only forced the point on the path to be close to its previous point, without considering its next point. This caused each point gets closer and closer to its previuos pint, .. previous previuos point, ..., the start point, as the iteration increases. Thus they get more and more deviated from the desired goal.

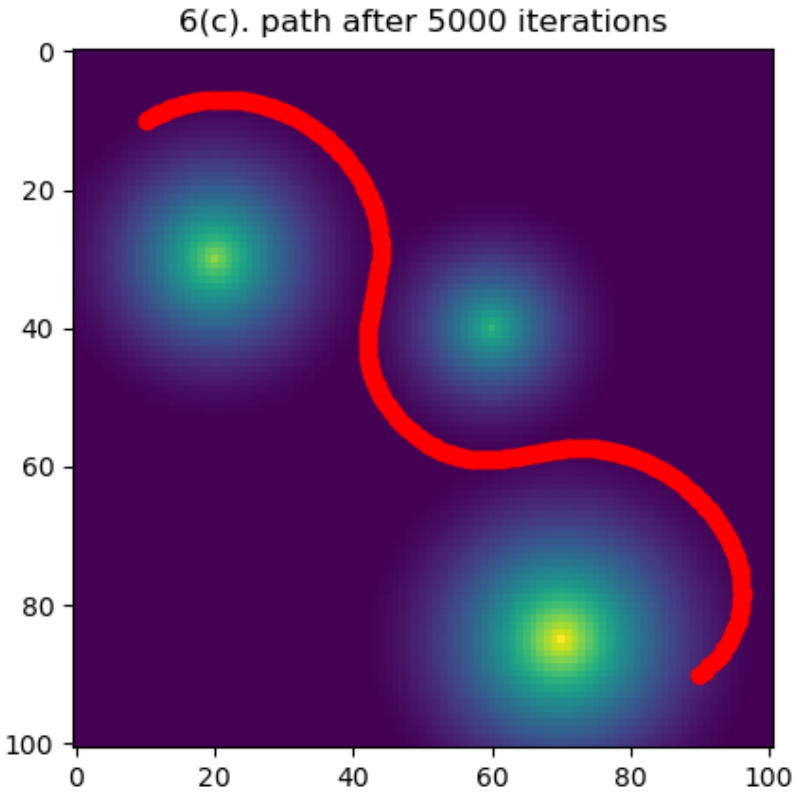



6(b). path after 200 iterations
0
20
40
60
80
100
0
20
40
60
80
100


6(b). path after 500 iterations
0
20
40
60
80
100
0
20
40
60
80
100

(c) Run *python q6.py* to get the plots of paths.
The plots below show the path after 100, 5000 iterations.
This version is better, because in the augmented gradient updates, we forced the point on the optimized path to be close to both its previous and next point. We take into consideration that for a continuous path, a point should not only be close to its previous optimized point, but also close to its next point, and thus moving towards the desired goal.




6(c). path after 5000 iterations
0
20
40
60
80
100
0
20
40
60
80
100

(d) With other conditions the same, but different initial paths, the final optimized path will be likely to be different.
This is because we used the gradient descent to optimize our chosen initialized path, so that the points get more and more away from the obstacles than their initial positions, (and also reach the goal).
We did gradient descent based on our initialized point positions on the path. The gradient will be different with different initialized values, and therefore the optimized results will be different.

(e) We could do hyper-parameter tuning on the weight values we used on the obstacle cost and the smoothness cost.
Since in the previous parts, we add a very large weight on the smoothness cost, compared to the obstacle cost. This might cause the gradient descent did not take much effect when it encounters the obstacle regions.
Thus we could adjust the weights (i.e. increase the weight on the obstacle cost) based on the robot performance.
The weight values are also not necessarily to be constants. We could also increase the weight on the obstacle cost, when the robot cross some high risk obstacle regions, if we have some prior information on the environment.