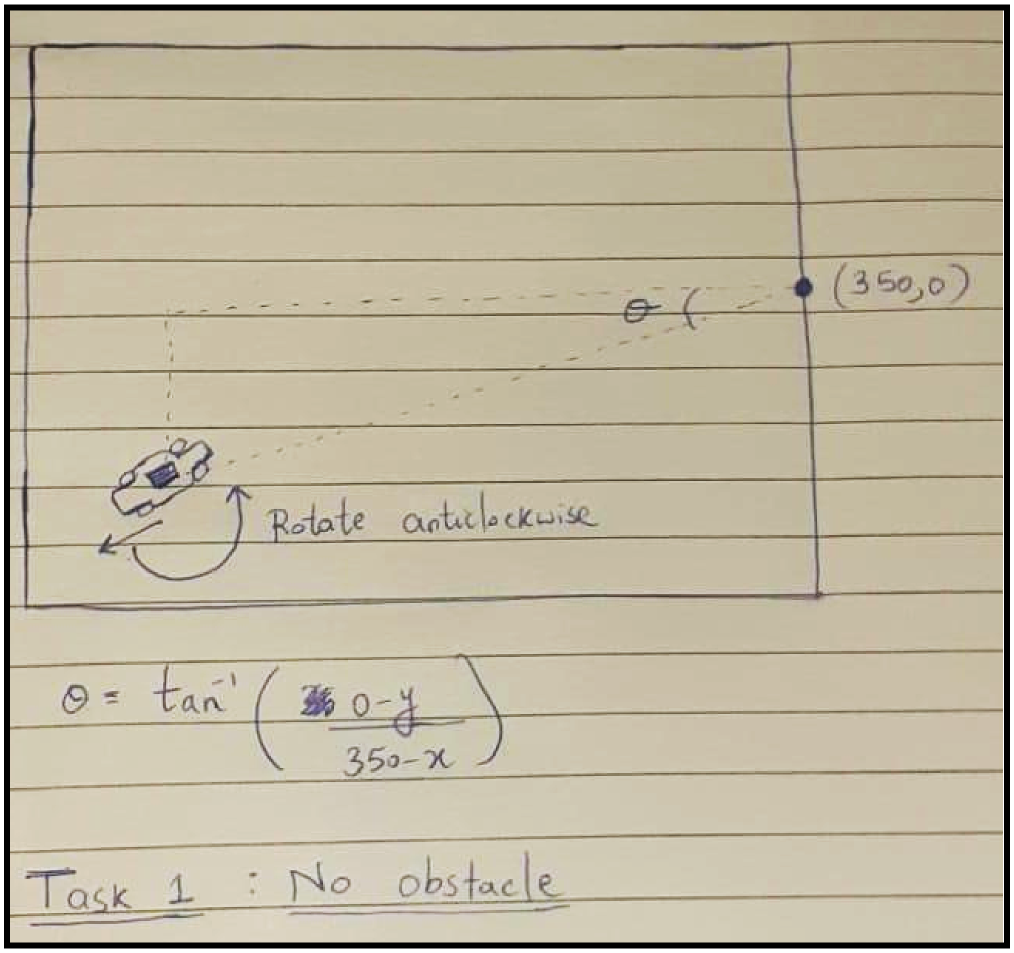
CS747 Assignment 3

Task1:

Since the velocity cannot be negative for task1 I rotated the car along the direction of shortest path from source to destination by calculating the angle as $tan^{-1}(\frac{(0-y)}{(350-x)})$ and providing the action as-

```
def align_state(self,state,final_angle):

        act=[0,0];f=False
        if state[3]>180 :
            state[3]=state[3]-360

        angle=final_angle
        #print(state[3],angle)
        if angle>180 :
            angle=angle-360

        if state[3]-angle<0:
            act=[2,0]

        if abs(state[3]-angle)<2.0:
            f=True

        return f,act
```

1. Checking if rotating anti/clockwise is faster and using the negative acceleration as brake and turning +-3 degrees appropriately.
2. Rotated until the error in angle is less than 2 deg.
3. Then applied full 5 unit acceleration along the path aligned.

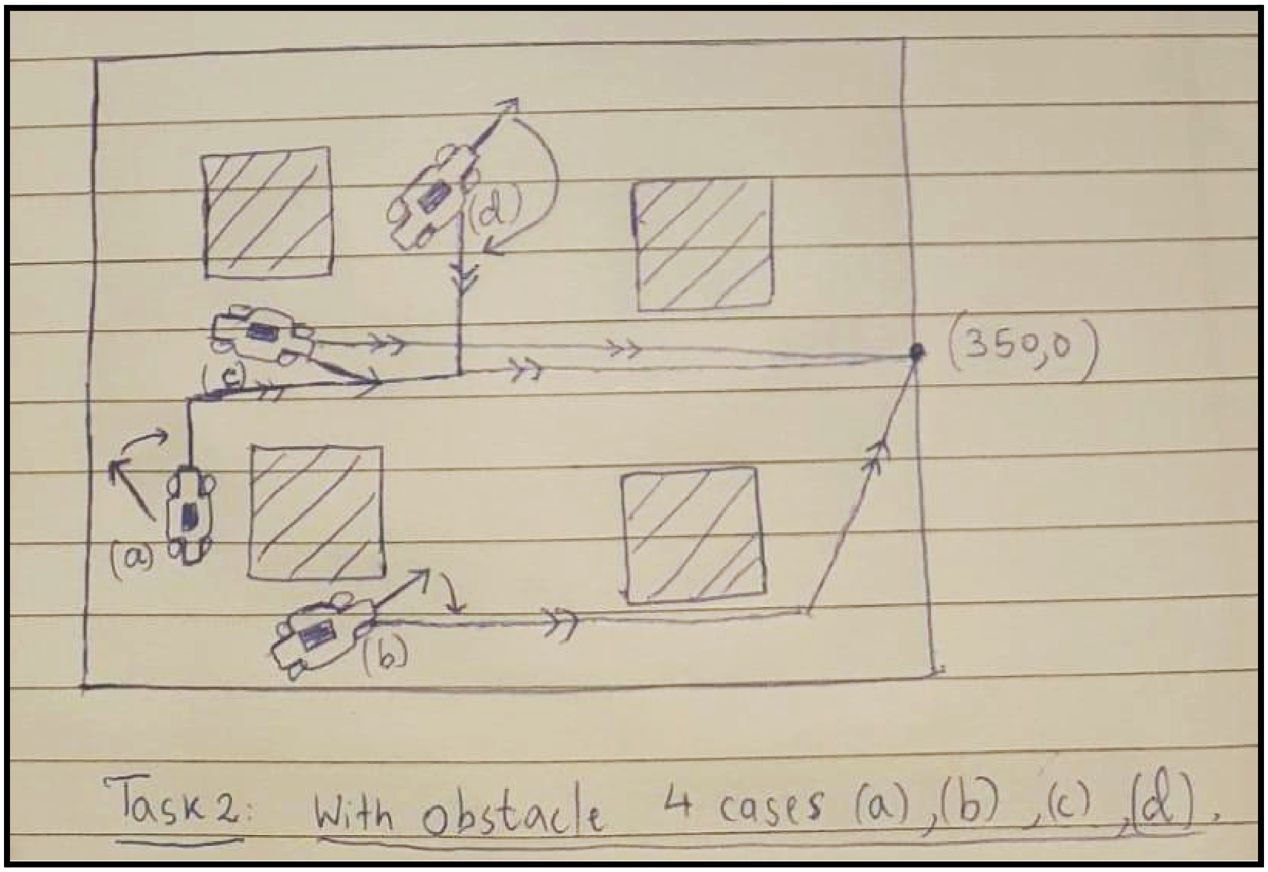


$$\theta = \tan^{-1}\left(\frac{0-y}{350-x}\right)$$

Task 1 : No obstacle

Above figure explains the methodology for task1

Task2:



Case a),d)Car has a path to x axis i.e there is no obstacle on its vertical path

Orient the car along vertical direction

Bring it first to x-axis i.e around y-cord -7 to 7

Halt the car by giving negative acceleration

Then use task1 to bring it to the destination since there is no obstacle in its direct path henceforth in its route.

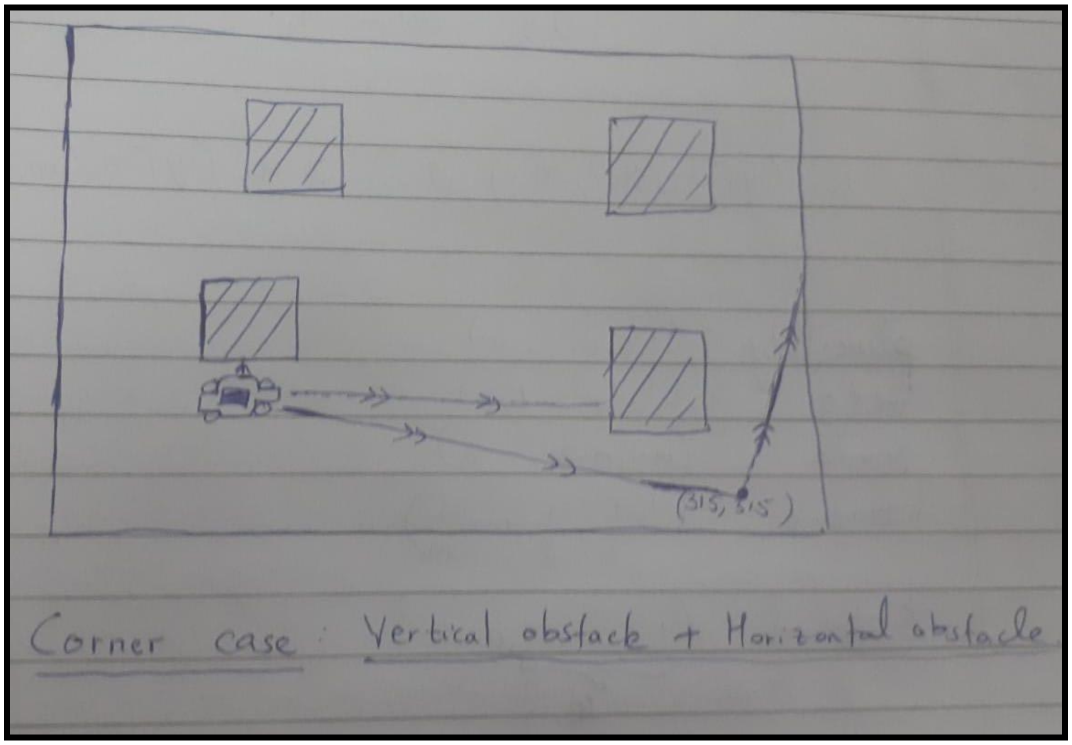
Case b)Since there is obstacle in its vertical path

Orient the car in horizontal direction

Bring it to the end of the grid ie x-cord 310 to 328 since it is guaranteed that there will be no obstacle after x-cord=300

Orient the car along the shortest path and give full acceleration similar to task1.

Case c)Since it is already close to x-axis there is no obstacle in its path(the endpoints of the square are guaranteed to be beyond +-70 and within +-280) hence use task1 approach.



Case e) If both vertical and horizontal obstacle are present then instead of aligning along x-axis align it along the line joining itself to (315,315) in order to avoid the obstacle then use task1 to align itself to destination and give full acceleration.

To detect a obstacle in its verical path used the following piece of code:

```
def vertical_obstacle(self,state,ran_cen_list):
        for i in range(4):
            if ran_cen_list[i][1]*state[1]>0 and
abs(state[1])>abs(ran_cen_list[i][1]):
                if abs(state[0]-ran_cen_list[i][0])<=54:
                    return True
        return False
```

1. If the x-cord of the center of car and center of square obstacle lies within 50 unit distance(54 for safety) then there is an obstacle either above or below the car.
2. To check if it lies in between the vertical path to the x-axis, check if the y-cord lies on the same side of x-axis and the y-cord of the obstacle has to be less than the y-cord of the car.

Conclusion-My solution approach requires at most 2 turns and 1 deceleration hence the car is able to reach the destination in time steps less than 250 in almost all the test cases without hitting the obstacles.

Other approaches tried:

1. Tried maneuvering along the shortest path calculated using BFS.Failed as controlling the car was difficult.
2. Tried value iteration but failed since it was time consuming.
3. Tried sarsa with boolean and rbf kernel features to discretize the continuous space but failed as it was error prone and took a long time to learn the q function.