CS747: Assignment 2

Yashvardhan

October 2022

1 Task1: next_task

- We were given with current location of the car and needed to reach the target centered at (350, 0) coordinates.
- I used this location/state to describe the next action
- I calculated the vector or direction of the shortest path to the road.
- using this I steered the car in that direction and accelerated towards the target.
- Every time the next state function was called I recalculated the direction to take care of the noise.

```
"""

# Replace with your implementation to determine actions to be taken
action_steer = None
action_acc = None

"""
    state = np.array([self.x, self.y, self.vel, self.angle])
"""

action_acc = 2
action_steer = 1

y1 = 0-state[1]
x1 = 350-state[0]
rad = np.pi*(state[3]/180)
cosA_B = np.cos(rad)*x1/(np.sqrt(x1**2+y1**2)) + np.sin(rad)*y1/(np.sqrt(x1**2+y1**2))

if cosA_B < 0.97:
    action_steer = 2
else:
    action_acc = 4
```

2 Task2: next_task

- In this question, 4 potholes were also spawned randomly.
- We needed to avoid them, in addition to our original objective.
- We were given with current location of the car and needed to reach the target centered at (350, 0) coordinates.

- I used this location/state to describe the next action
- In the controller function we generated the potholes
- I passed the list to the next action function of the class.
- There I used if statements to describe the motion of the car.
  - First: If the car is in the middle path and there is no obstacles ahead, then the car accelerates towards the road.
  - Second: If the car is not in the middle path, but there are no obstacles in the y-direction it moves towards, then the car moves in the y-direction to reach the middle safe zone
  - Third: if the car is not in the middle safe zone and there are obstacles in the y-direction, then the car moves either left or right till there are no obstacles in the y-direction.
- Every time the next state function was called I recalculated the direction based on the current state and condition of the field.

```
th = 65
th_safe_y = 40
direct = self.ran_cen_list
safe_y_p = min(direct[0][1], direct[2][1])-th
safe_y_p = min(safe_y_p, th_safe_y)
safe_y_n = max(direct[1][1], direct[3][1])+th
safe_y_n = max(safe_y_n, -th_safe_y)

if state[1] > safe_y_n and state[1] < safe_y_p:
    if state[3] > 3 and state[3] < 357:
        if state[3] > 180:
            action_steer = 2
        else:
            action_steer = 0
    else:
        action_acc = 4
elif(
    (state[0]>0 and state[1]<0 and (abs(direct[1][0]-state[0]) > th or direct[1][1] < state[1]) ) or
    (state[0]<0 and state[1]>0 and (abs(direct[2][0]-state[0]) > th or direct[2][1] > state[1]) ) or
    (state[0]>0 and state[1]>0 and (abs(direct[0][0]-state[0]) > th or direct[0][1] > state[1]) ) or
    (state[0]<0 and state[1]<0 and (abs(direct[2][0]-state[0]) > th or direct[3][1] < state[1]) )
):
    if state[1] > 0:
        if abs(state[3]-270) > 5:
            if state[3]-90 > 0 and state[3] < 270:
                action_steer = 2
            else:
                action_steer = 0
        else:
            action_acc = 4
```

```
        else:
            if abs(state[3]-90) > 5:
                if state[3] > 90 and state[3] < 270:
                    action_steer = 0
                else:
                    action_steer = 2
            else:
                action_acc = 4
    else:
        direction = "right"
        if(
            (state[0]>0 and state[1]>0 and state[0] < direct[0][0] and abs(direct[0][1]-state[1]) > th) or
            (state[0]>0 and state[1]<0 and state[0] < direct[1][0] and abs(direct[1][1]-state[1]) > th) or
            (state[0]<0 and state[1]>0 and state[0] < direct[2][0] and abs(direct[2][1]-state[1]) > th) or
            (state[0]<0 and state[1]<0 and state[0] < direct[3][0] and abs(direct[2][1]-state[1]) > th)
        ):
            direction = "left"

        if direction == "right":
            if state[3] > 3 and state[3] < 357:
                if state[3] > 180:
                    action_steer = 2
                else:
                    action_steer = 0
            else:
                action_acc = 4
        else:
            if abs(state[3]-180) > 5:
                if state[3] > 180:
                    action_steer = 0
                else:
```

```
ran_cen_1x = random.randint(120, 230)
ran_cen_1y = random.randint(120, 230)
ran_cen_1 = [ran_cen_1x, ran_cen_1y]

ran_cen_2x = random.randint(120, 230)
ran_cen_2y = random.randint(-230, -120)
ran_cen_2 = [ran_cen_2x, ran_cen_2y]

ran_cen_3x = random.randint(-230, -120)
ran_cen_3y = random.randint(120, 230)
ran_cen_3 = [ran_cen_3x, ran_cen_3y]

ran_cen_4x = random.randint(-230, -120)
ran_cen_4y = random.randint(-230, -120)
ran_cen_4 = [ran_cen_4x, ran_cen_4y]

ran_cen_list = [ran_cen_1, ran_cen_2, ran_cen_3, ran_cen_4]
eligible_list = []
```