CS 747 Programming Assignment 3

Adish Shah

6th November, 2022

Contents

1 Task 1 **2**

1.1 Overall strategy 2

1.2 Code explanation and certain subtleties 2

1.3 Guarantee of reaching the road 2

2 Task 2 **3**

2.1 Overall strategy 3

2.2 Code explanation and subtleties 3

2.3 Guarantee of reaching the road 4

3 Video for demo **4**

1 Task 1

1.1 Overall strategy

The technique used here is of path planning. Since, in this task the location of the road is fixed and there aren't any restrictions placed on the points that we can visit inside the boundary, we can choose the destination point of the car to be (350,0). Given the starting location of the car, I calculated the angle(orientation wrt the x axis) needed to reach (350,0). So, we rotate such that our car's orientation matches as what is required. Subsequently, the car moves forward along that path at full acceleration without steering in any direction.

1.2 Code explanation and certain subtleties

One thing to note is that the initial angle obtained from the state list, has negative angles, whereas subsequently angles are always between 0 and 360 degrees. Hence, to avoid ambiguity, my code ensures angle lies within 0 and 360.

Coming to the code, I have a `self.set` boolean which is False at the beginning of each episode and set to True after the first state has been seen. Here we evaluate the `self.angle_needed variable` as $\tan^{-1} \frac{0-y_{car}}{350-x_{car}}$. The function `np.arctan2` helps us achieve this with the suitable range. Using the `set_steer` function, the direction of initial rotation is set in the variable `self.direction`, depending on what action achieves the angle quickly(clockwise or anticlockwise). To check if we have reached the required angle, we set a **threshold of 2.1**, i.e if the current angle is within this threshold, we can stop rotating and move forward safely. The value of 2.1 was found to be better using trial and error.

1.3 Guarantee of reaching the road

Since the acceleration and angle of the car are arbitrarily added with some amount of noise, it may seem that the car doesn't reach the road. But given the variance in the noise is just 0.005, it is not possible for the car to not reach the road, as we have sufficient room in either direction (up or down of 110 units), since we have aimed to reach the center point. Mathematically can be verified, by taking the edge case.

2 Task 2

2.1 Overall strategy

The technique used here again is of path planning and exploiting the knowledge of the location of the pits. My approach here was to move only right and vertically up or down (right here means in terms of our view of the simulation). At each step when deciding what action to take, I first check if the absolute value of the y coordinate of the car is within 30, because it is guaranteed that there will be no pits in this range. If this is the case, we can simply drive right, which is what `drive_right` function does. Our goal is to reach this area and continually drive right from here, if it isn't already the case. To do so, we check for any pits in the vertical direction (done by the `close_to_vpit` function). If there is a pit in the vertical direction in the vicinity, we drive right. Otherwise, we drive vertically up or down straight to the area within 30 units of the X axis. Thus, we can reach the road this way.

2.2 Code explanation and subtleties

In the following underscore has been replaced by hyphen for the function names. The threshold for angles is set to 3 throughout in the `self.threshold` variable.

close-to-vpit function

We set the `self.pits_list` variable to hold the `ran_cen_list`. We then iterate through these centers and check if the current position of the car is within a given value of the centers. Even if one pit falls under this vicinity, True is returned.

The basic check is that : return True if $x_{car} \in (x_{center} - \delta, x_{center} + \delta)$ and its y coordinate between the car and x-axis. Suitable δ is chosen.

close-to-center function

This function checks if the current absolute y coordinate is within 30 units of the X axis and returns True if it indeed is.

drive-right function

Here we follow a strategy similar to Task 1. We initially rotate anticlockwise or clockwise, depending on what is faster and our required angle should be 0. Hence we check if the current angle becomes $\leq 0 + threshold$ or $\geq 360 - threshold$. While rotating we set the acceleration to -5 (effectively braking). Then we drive straight by setting steer to 0, and acceleration to +5.

drive-vertical function

Similar to the above function our required angle is 270 or 90 depending on the y coordinate being positive or negative. We rotate till required angle is achieved within the set threshold and then drive straight without steering.

next-action function

This function brings the overall strategy together, as discussed previously. It calls `close_to_center` and drives right, if it is true. Otherwise we return `drive_right` if there are vpits nearby (checked by the function `close_to_vpit`), or return `drive_vertical` if there aren't any vpits.

2.3 Guarantee of reaching the road

Given that each quadrant has a pit, and the way the locations of the pits have been set, it is guaranteed that there exists some place where there isn't any pit in the vicinity in the vertical direction of the car. Hence at this position the car moves straight up or down, and reaches the point where it is within 30 units of the X-axis. As there can't be any pit here(since the center of the pit are at least 110 units away from both the axes), driving right continuously without steering guarantees that we reach the road. Again the variance is very small and can't cause significant drift of the car.

3 Video for demo

I have attached the link for the video of my simulation, which will aid in understanding the exact technique used. Click here for demo of simulation.