AWS DeepRacer

Ryhan Sunny N01654285
Sharmi Das N01639206
Noah Nkalubo Nsimbe N01645322
Murtuza Salman Mohammed N01622777
Amaka Genevieve Jane Awa N01613636

The Lightning McQueen

Parameters used

waypoints:A list of coordinates representing waypoints along the track.
closest_waypoints: Indices of the two closest waypoints to the vehicle's current position.
heading: The current heading (direction) of the vehicle.

Initialized the reward value to a typical value of **1.0**. Calculate the direction of the track (centre line) based on the coordinates of the next and previous waypoints.

The track direction is converted from radians to degrees.

Direction difference calculates the absolute difference between the track direction and the heading direction of the vehicle.

If the absolute difference is greater than 180 degrees, adjust it to the smaller angle between the two directions.
Define a threshold (DIRECTION_THRESHOLD) for the acceptable difference between the track direction and the heading direction.

If the direction difference exceeds the threshold (it indicates the vehicle is not aligned well with the track direction), the reward is penalized by halving it (reward *= 0.5).

Training configuration

Race type
Time trial

Environment simulation
re:Invent 2018 - Counterclockwise

Reward function
Show

Sensor(s)
Camera

Action space type
Continuous

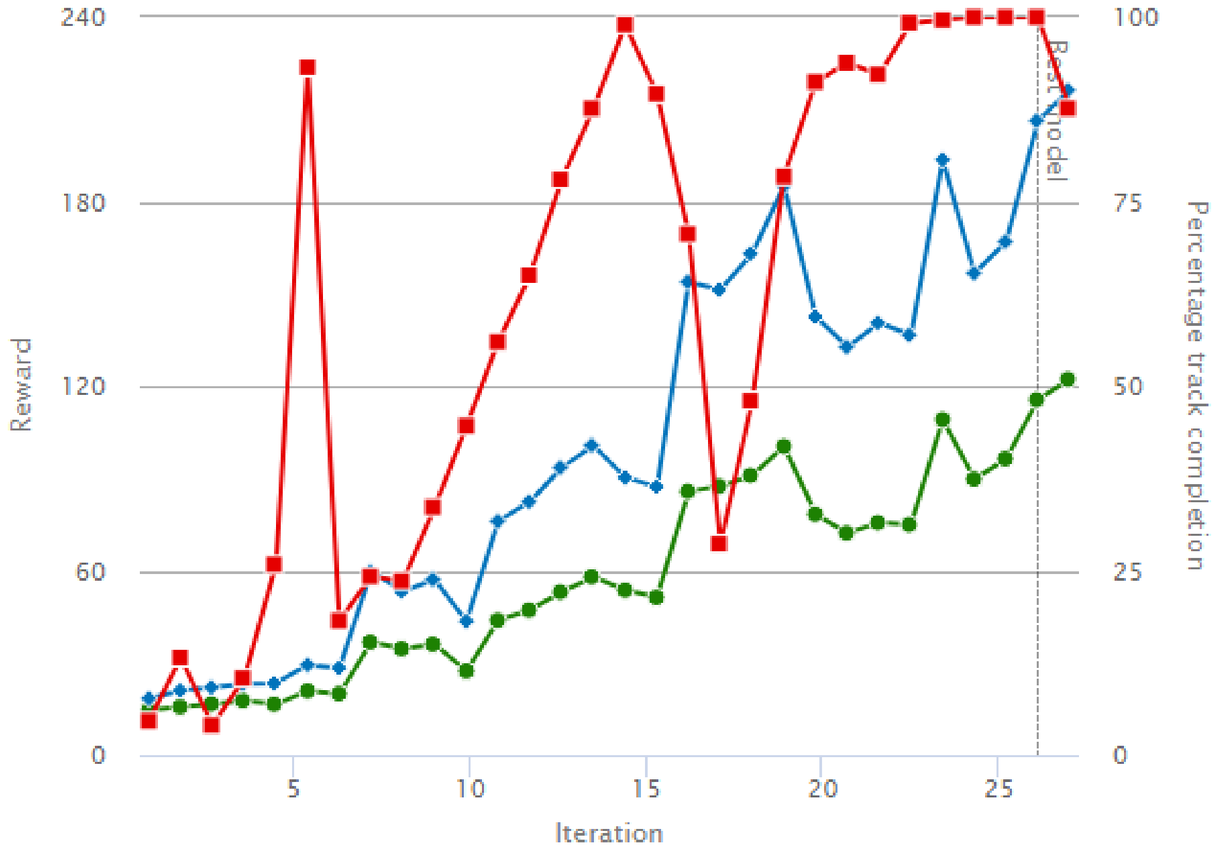
Action space
Speed: [1.1 : 2] m/s
Steering angle: [-30 : 30] °

Framework
Tensorflow

Reinforcement learning algorithm
PPO

Hyperparameter	Value
Gradient descent batch size	64
Entropy	0.01
Discount factor	0.88
Loss type	Huber
Learning rate	0.0003
Number of experience episodes between each policy-updating iteration	18
Number of epochs	4

Reward graph Info



Average reward (Training)

Average percentage completion (Training)

Average percentage completion (Evaluating)

Out best time: 10.5s !!

Evaluation results

Trial	Time (MM:SS.mmm)	Trial results (% track completed)	Status
1	00:10.545	100%	Lap com
2	00:10.695	100%	Lap com
3	00:10.606	100%	Lap com

Using multiple reward functions

```python
class RewardClass:
    def __init__(self, params) -> None:
        self.__params = params
        self.__weights = {
            "follow_center_line": 0.25,
            "stay_inside_borders": 0.25,
            "avoid_zig_zag": 0.15,
            "all_wheels_on_track_and_speed": 0.2,
            "steering_angle": 0.15,
        }

    def __follow_center_line(self):⟺

    def __stay_inside_borders(self):⟺

    def __avoid_zig_zag(self):⟺

    def __all_wheels_on_track_and_speed(self):⟺

    def __steering_angle(self):⟺

    def get_reward(self):⟺


def reward_function(params):
    reward_class = RewardClass(params)
    reward = reward_class.get_reward()
    return float(reward)
```

Trial	Time (MM:SS.mmm)	Trial results (% track completed)
1	00:14.653	100%
2	00:14.530	100%
3	00:14.156	100%