Report

by Ashwin S H

Submission date: 19-May-2022 04:34PM (UTC+0800)
Submission ID: 1839703070
File name: ProjectReport_1.pdf (2.5M)
Word count: 5060
Character count: 24565

Chapter 1

Introduction

Autonomous vehicles are a class of vehicles that can perceive its surroundings and moving with scant or no human interaction. They can utilize multiple sensors such as LiDARs, RADARs, GPS, cameras etc. to achieve this. In the present world, lots of tasks are being automated to provide humans more convenience and safety. There can be scenarios where the drivers are not in a good state to drive, and it may lead to accidents. Instead, if the task of driving is given to an adequately trained machine, it will perform its task with maximum efficiency every single time.

Machine learning is defined as a process where a program or a model learns from its experience of carrying out the specified task to improve its performance. Usually, ML algorithms are classified into three broad categories: reinforcement, supervised and unsupervised learning. In supervised learning algorithms, the model is trained beforehand using labelled data to perform a certain task whereas in unsupervised learning the model is forced to learn from unlabeled data. The third category is Reinforcement Learning where an agent improves its performance in a task by constantly interacting with its surrounding.

For the process of autonomous driving, reinforcement learning is the optimal choice as the vehicle needs to actively interact with its environment to drive safely. It is impossible to train the vehicle for all possible scenarios beforehand thus ruling out supervised learning approach. There are a few challenges of using RL though, such as bridging the gap between simulation and reality, sample efficiency etc.

Autonomous driving systems consist of a few standard blocks as shown in Figure 1.1 [1]. Every autonomous driving vehicle consists of multiple cameras, RADARs, and LiDARs, and a GPS-GNSS system. Scene understanding, localization, mapping, planning, driving policy and control are a few problems addressed by these modules.

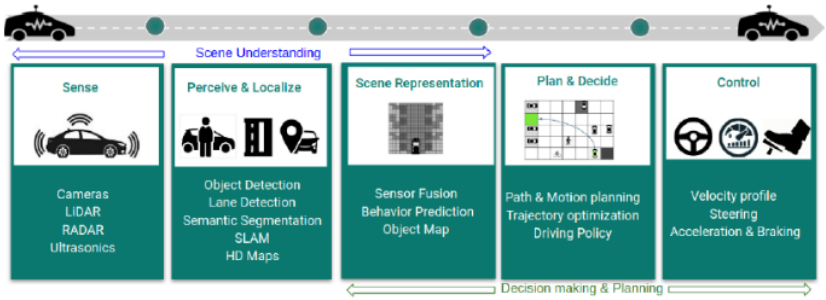

Figure 1.1: Standard blocks of Autonomous Driving System [1]

In the Reinforcement Learning model, an autonomous agent masters its performance by persistent interaction with its environment.Reward function is a performance criteria which is used to evaluate the RL agent. [2]. The general scenario of reinforcement learning is as shown in Figure 1.2. For each state encountered, the agent performs an action and obtains a reward(which can be positive or negative) from its environment based on the practicality of its decision. Agent's objective is to maximize the cumulative rewards obtained during its entire lifetime of the task. Eventually, by exploiting the knowledge

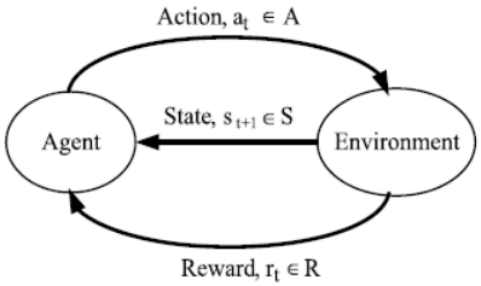

Figure 1.2: General scenario of Reinforcement Learning [2]

learnt the agent can increase the lifetime reward.

Chapter 2

Literature Survey

In the field of autonomous driving, a lot of progress has already been done. Recent papers have been reviewed and findings from a few of them are listed below.

The prime challenge in reinforcement learning is managing the balance between exploration and exploitation. To enhance the rewards it receives, an agent must exploit its knowledge by choosing actions that prove to be highly rewarding. In contrast, to ascertain such favourable actions, it has to make certain risky decisions. This may or may not result in higher rewards. This [3]. The strategy used by the agent to choose an action for a given state is known as a policy. If a particular policy gives maximum reward, such a policy is termed as optimal policy.

The work by Zhao et al., 2020 [4] aims to model the decision making and interactions between various vehicles which run on highways. Double Deep Q-Network (DDQN) is employed to train the host vehicle. An open source simulation platform called 'SUMO - Simulation of Urban MObility' is used to implement the work. The driving environment is created by having three driving lanes and randomly distributing 20 cars on the highway. While driving,

the host constantly measures the distance between itself and the car ahead of it. Incase the distance reduces between successive measurements, it begins to apply brakes to avoid collision. The speed of the host is altered accordingly by the algorithm.

Deep Deterministic Policy Gradient (DDPG) is one of the significant algorithm in the field of RL. It has been thoroughly explained in the work by Huang et al., 2019 [5]. In any autonomous driving system, mapping of driving state to driving action is an important criteria. This work establishes the same mapping using DDPG algirithm on TORCS simulation software. This gives an idea of driving the autonomous vehicle in the real world. Here, actor network and target actor are used for outputting the best action for a particular state. Q-value is estimated using critic and target critic network. The transitions obtained from exploration are stored in replay buffer.

The work done by Zhang et al., 2018 [6] focuses on using Double Deep Q Learning to build the vehicle speed control model. In Q Learning, an agent selects an action with best Q value which might not be an optimal choice. This is considered as over-estimation of action value which results in the complicated learning process. To resolve the over-estimation, two separate Q value estimators (Double Q) can be used. Unbiased Q-value estimation of the actions can be done by using these two separate estimators. The implementation of this concept using Deep Neural Networks is known as Double Deep Q Learning. The conventional supervised learning approach is not used so that the system is forced to approximate the values of the function by interacting with the environment. It also designes a reward network where each state is mapped to a reward expressing the expectation of the state. Double Deep Q Learning is efficient in both value accuracy and policy quality. The authors have reported that DDQN model's score is very high when compared with that of Deep Q-

Network (DQN). The future work would be focused more towards extensive testing with more real world data.

The work by Chopra et al., 2020 [7] is aimed at steering the vehicle in its path with the help of Deep Q-Learning algorithm. The model uses raw images, sensor inputs and calculated rewards to create a Q-value approximator which is used to steer the car. Authors were able to develop an algorithm to achieve the same in TORCS simulator. The drawback of this work is that it is very time consuming to train the model. This drawback provides a future scope of implementing Imitation Learning to make the algorithm faster by first training it using labelled data and then running RL on it. Table 2.1 shows the state space, action space and reward function of various existing work related to autonomous driving using reinforcement learning.

This survey paper by Elallid et al. 2022 [8] focuses on deep learning and reinforcement learning based approaches on autonomous vehicles with major functionalities such as scene understanding, decision making, motion planning, vehicle control etc. This paper reviews the literatures pertaining to autonomous vehicles that use DL/RL from 2016 to 2021. An extensive comparison is also made by the authors with respect to the functionalities listed above. It thoroughly explains the various sensors and their uses in autonomous vehicles. It also highlights various challenges which need to be addressed by the researchers in the days to come. One of the main challenges are behavious of AVs in different weather and lighting conditions. The authors are also unsure about the reliability of the Neural Network models that are trained using datasets.

The work by Rasheed Hussain et al. [9] focuses on the results that have been accomplished till now and the challenges that lie ahead for researchers working in the field The automobile industry attained significant milestones in designing reliable, safe, and efficient vehicles throughout the last century. Be-

cause of momentous progress in computation and communication technologies, autonomous vehicles are becoming a dream come true.

The work by Zhu et al. 2020 [10] is a car-following simulation model i.e. the agent learning to follow the lead vehicle that is being controlled by a human in front of it. Rather than utilizing a plethora of sensors, they have just used the distance between the agent and the lead vehicle to calculate time for collision, jerk in driving etc. The reward function is developed by observing human driving data captured real time from the lead car and later combining it with driving related features such as efficiency, comfort and safety. They have used 'Next Generation Simulation' software for the implementation.

This paper by Omeiza et al. 2021 [11] is about 'explainable autonomous driving'. 'Explainability' is an essential prerequisite for AVs. AVs must be able to explicate what they see, do, and might do in the environments in which they interact. This will build confidence and trust in AVs. Four main points are addressed in this paper. They are motivation for explanations, requirements of explanations for autonomous vehicles, review of previous work on explanations for AVs, and finally, authors have given a conceptual framework for autonomous vehicles explainability.

The paper by Grigorescu et al. 2020 [12] addresses the major areas which form open challenges for the budding researchers in the field of autonomous driving. They make a strong statement that Deep Learning and Artificial Intelligence will play a phenomenal role in overcoming these challenges. Among the major areas they addressed, functional safety and real time computing and communication prove to be really challenging.

The paper by Yang et al. 2020 [13] is a complete survey of significant works in the area of AVs. It aims to analyze the use of Artificial Intelligence in building the primary applications of autonomous driving. These applications are perception, localization & mapping, and decision making. It explores the recent approaches to comprehend how AI can be made use of and what are

the challenges associated with their implementation. Based on the survey of current practices and advancement of the technologies, this paper further provides acumens into impending opportunities regarding the use of AI in juxtaposition with other developing technologies.

This web article [14] speaks about the evolution of autonomous vehicles. It states that the cars became autonomous first in 1920s and were "Phantom Autos". They were called so because they were remote controlled. Later in the 1980s self-managed autonomous cars were introduced by Mercedes, which were not fully automatic. It was performing lane change with human intervention. Because of the advancements in automobiles, information technology and the increasing interest in designing the autonomous cars of many young researchers, this will result in low-cost, reliable, efficient cars on the road in the near future.

In the work done by Cao et al. 2022 [15] the authors have discussed about Trustworthy improvement Reinforcement Learning (TiRL). It has been mentioned that in spite of RL algorithms having the ability to constantly improve, there are instances where they are unreliable due to rule or model based algorithms. To get the best of both worlds the authors have designes a decision making framework that utilizes Reinforcement Learning and rule/model based algorithms.By doing so, their final framework has the potential to self-learn with better reliability. They have done simulations with more than 42000 kilometres of driving which was calibrated with human driving data. They have proved that TiRL can do better than any other arbitrary model based policy.

Table 2.1: Details of states, actions and rewards

Work by	**State space**	**Action space**	**Reward function**
Zhao et al., 2020 [4]	$S = \{pos_x, pos_y, v_x, v_y\}$	$A = \{acceleration\ value\}$	$R = 1 - \frac{V_{max} - V_x}{V_{max}}$
Huang et al., 2019 [5]	$S = \{v_{x,y,z} \epsilon R^3, \zeta \epsilon R^{19}, \delta, \theta\}$	$A = \{accelerate, brake, steer\}$	$R = vcos\theta - vsin\theta - v\delta$
Zhang et al., 2018 [6]	GPS position, acceleration, relative speed and position	$A = \{a \vert a\epsilon \{accelerate, decelerate, maintain\}\}$	R = {-2, -1, 0, 1, 2} depending upon the action taken
Chopra et al., 2020 [7]	Camera images	$A = \{left, half\_left, no\_action, half\_right, right\}$	$R = vcos\theta$

Chapter 3

Problem Statement and Objectives

3.1 Problem Statement

From the literature review it is noted that, in most of the work RL algorithms have been extensively used to control the autonomous vehicles. However most of these work concentrated on only one car in the track/road and always focusses on increasing the reward function depending on the speed of the car. The problem statement of this work is to detect other cars and obstacles present on road and drive the autonomous vehicle keeping in mind all those obstacls.

3.2 Objectives

Based on the problem statement defined in the previous section, the following objectives are formed:

1. To deploy a suitable RL algorithm to control the speed and facilitate lane keep operation of the car.
2. To avoid collisions by detecting obstacles using sensors.

Chapter 4

Methodology

After extensive literature survey, it is found that Carla, TORCS, Airsim, SUMO are a few simulators which are considered for autonomous driving applications. Each of them has its own pros and cons. Carla and Airsim, being relatively new softwares, are more suitable for vision based driving and are known for their photo-realistic driving scenes which require higher graphic computations. SUMO does not have its RL capabilities very well developed. On the other hand TORCS simulator satisfies the requirements to implement this project. Hence, TORCS-1.3.7 in Ubuntu 16.04 OS with Python 3.5 with tensorflow and keras deep learning framework is used. It is an open source, lightweight simulator software which is preferred due to its simplicity and variety of driving dynamics. The server code to control the car is installed with the software during installation. The client code to communicate with the server is written in python language as it offers immense libraries to code RL algorithms. The simulator offers a plethora of sensors and readings. Only the required sensors needs to be activated lest it throws too many unwanted values during the race. Once these sensor readings are interpreted, it needs to be fed into the client code to establish control over the car during execution.

4.1 DDPG Algorithm

DDPG is one of the RL algorithms which can be applied on tasks which deal with continuous state spaces. It is an off-policy model-free algorithm. It combines ideas from Deterministic Policy Gradient (DPG) and DQN. It uses Experience Replay and slow-learning target networks from DQN, and it is based on DPG, which can operate over continuous action spaces. The working of the algorithm is as illustrated in Figure 4.1.

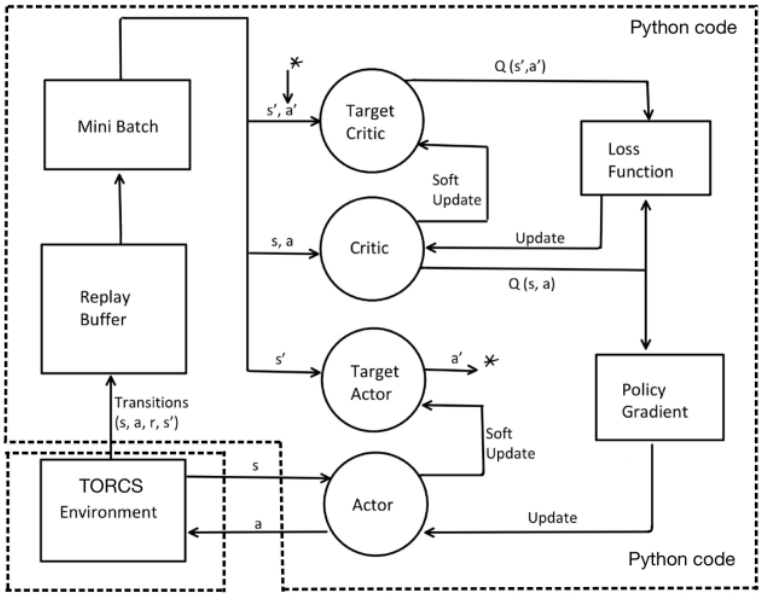

Figure 4.1: Deep Deterministic Policy Gradient block diagram

4.2 State Space

The state space consists of velocity of the car, tangential angle of the car with respect to road, horizontal distance between the centre of the lane and car, distance between lane edge and agent. According to the agent state, the actor

network encodes a state to action at time t. This action is sent to the simulator through shared memory to control the car and return the next state, reward for the same. In each iteration, the set $\langle state, action, reward, next\_state \rangle$ is collected and is stored into the replay buffer for network training. The velocity vector of the car is divided into three orientations x, y, z ϵR^3. The offset $\delta \epsilon R^1$ is the distance between the agent and centre of the track is normalized between (-1, 1) . The distance between the agent and the edge of the lane $\zeta \epsilon R^{19}$ is measured in (45, -19, -12, -7, -4, -2.5, -1.7, -1, -0.5, 0.5 ,1 , 1.7, 2.5 , 4 , 7, 12, 19, 45) degrees with respect to front of the agent. The parameter $\theta \epsilon R^1$ is angle between direction of the track and direction of the agent and it is measured in radians between ($-\pi, \pi$). These values are as shown in Figure 4.2 and Table 4.1.

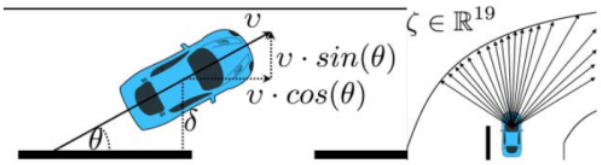

Figure 4.2: Driving Environment

4.3 Action Space

Agent action space a_t = ($accelerate, brake, steer$) represents agent acceleration, braking, and steering angle respectively as illustrated in Table 4.2 in detail. Acceleration value of 1 implies that the agent has applied full throttle i.e. in human driving terminology, it has pressed the accelerator pedal completely. If acceleration value is 0, it implies that the agent has completely let go of the accelerator pedal and it is moving only due to its momentum. Brake value of 0 states that the brake pedal is not at all pressed and a value of 1

Table 4.1: Table of States [5]

States Symbol	Description	Range
$v(\ km/h)$	agent's velocity	$(\ 0, 300)\ \epsilon R^3$
δ	offset between centre of the lane and the agent horizontally	$(\ -1, 1)\ \epsilon R^1$
$\theta(\ radian)$	departure angle between the lane and agent	$(\ -\pi, \pi)\ \epsilon R^1$
$\zeta(\ m)$	distance between edge of the lane and agent	$(\ 0, 200)\ \epsilon R^{19}$

imples that it is completely pressed. Steering value of -1 represents full right and +1 represents full left turn.

Table 4.2: Table of Actions [5]

Action Symbol	Description	Range
accelerate	current acceleration	$(\ 0, 1)\ \epsilon R^1$
brake	current braking	$(\ 0, 1)\ \epsilon R^1$
steer	current steering angle: negative for right, positive for left	$(\ -1, 1)\ \epsilon R^1$

4.4 Reward Function

One of the major tasks is designing an appropriate reward function. This function needs to be properly adjusted so that the car knows its priorities well and drives in a controlled manner. To briefly explain a scenario, consider an autonomous car X which is steadily going in its lane. It is getting some positive

reward. Suppose X detects some other car, named Y, which is seemingly rushing towards X and it might end up crashing onto the car X, then ego-car (X) must be getting higher reward to apply brakes and maneuver it away from Y. Inorder to achieve the objective of lane keep, the car needs to be trained with altered reward function wherein it must get a higher reward for staying in its lane and a lesser reward for swerving away from the required lane. Another possible approach here is to make use of the sensors to limit the car in one lane thereby creating a pseudo boundary for the car.

The current reward function is $R = vcos(\theta) - vsin(\theta) - v\delta$ where θ is the departure angle, v is the speed of the agent and δ is the offset between agent and centre of lane. If the departure angle is less, then $cos(\theta)$ will be high while $sin(\theta)$ will be a minute value. If the agent is near the centre of the lane, only a small value will be subtracted. In case it is near the edge of the lane, a higher value will be subtracted from the reward function. The second part of the work involves detection of obstacle using sensors. Two different sensors are used for this process, namely *opponents* and *track*. The *track* sensor is as represented by the ζ value which is shown in Table 4.1. It is a python-list comprising 19 range-finder sensors which contain the distance between the centre of the car and edge of the lane. The *opponents* sensor is also a python-list comprising 36 range-finder sensors. These are located throughout the car at every 10 degrees and they have a maximum range of 200 metres. In case no other cars are present, this sensor outputs the maximum value from each of the sensors. Otherwise it sends the distance between the agent and other cars present on track.

4.5 Front obstacle detection

The driving style implemented in this work is left hand drive(LHD). Four sensors out of the 36 *opponents* sensors are used for front obstacle detection.

Whenever the car enters a new state, it waits for a few millisecond to receive instructions from actor network on how to proceed. During this time interval, all the sensor readings are collected and analysed to see if any obstacles are present ahead of the vehicle. In case front *opponents* sensors detect a car within a range of 30-50 metres ahead, the agent stops accelerating i.e. sets its value to 0. After a few milliseconds, if the obstacle ahead is less than 30 metres, it applies brakes slightly i.e. it sets the brake value to 0.3. By doing so, it begins to slow down. If the obstacle is less than 20 metres away, it applies brakes a tad bit stronger. Now in case the obstacle is less than just 10 metres away, it applies full brakes so that it comes to a complete halt. If the obstacle is detected early enough, braking action is done in a step by step manner over a distance of 50 metres and it smoothly comes to a halt. In case the obstacle suddenly comes in the path of the agent with less distance between them, then the agent applies full brake inorder to avoid collision. In such a scenario, braking will not be smooth but that is secondary since it is more important to avoid collision.

4.6 Overtaking

This is interconnected with front obstacle detection. When driving in a lane and any obstacle appears suddenly, the only thing that comes to the mind of any driver is to apply the brakes and stop swiftly and that is exactly what the agent does too. On the other hand, if an obstacle is detected well in advance, the agent can plan to overtake the same. When the distance between the agent and the vehicle ahead is around 20 metres, agent takes another set of sensor readings to check if there are any vehicles that are currently on the right lane. If it is clear of any traffic, it steers to the right lane. After overtaking a vehicle, it checks if it can come back to its previous lane as sometimes there will be more than one vehicle going at a slow speed. If the path is again clear, it steers

back to its lane and continues driving.

The agent has a predefined speed limit of around 90 kilometre per hour. While overtaking, imagine a scenario where the agent has has completed half of the overtaking process by going to the right lane. At this point, if the vehicle which was being overtaken suddenly increases its speed and if this is equal to over more than the speed of the agent, the agent decides to abort the overtaking procedure. It slows down a little and checks if it can go back to its previous lane. If the road is clear, it steers left and follows the car which it had intended to overtake.

4.7 Side obstacle detection

In order to detect the obstacles at the sides of the agent, a total of 28 sensors - 14 for each side - are utilized. If all the vehicles that are plying over a stretch of road meticulously follow their lane, then agent takes no action. However if any vehicle deviates from its lane and enters the lane that belongs to the agent, at that time the problems arise. If the agent senses any vehicle approaching very close to it from sides, it slightly steers away to avoid collision. It moves away from the approaching vehicle only if there is space left on the road. In case it was already driving on the edge of the road and if some vehicle approaches from sides, then it slowls down letting the other vehicle pass. If such a situation occurs in the real world, drivers first press the horns of their vehicle to alert the driver who is deviating from their lane. Since the simulator does not have that privilege, it simply slows down as that is the best the agent can do to avoid a sideward collision.

Chapter 5

Result Analysis

Initially TORCS-1.3.7 was installed in the system along with all the python modules required to code RL algorithms. Then, a basic python client code is written to communicate with the TORCS server which enables the user establish control over the car. Controlling includes the speed variation, gear selection and steering control. The following are the screenshots (Figure 5, 6, 7, 8, 9) obtained during the execution of the program.

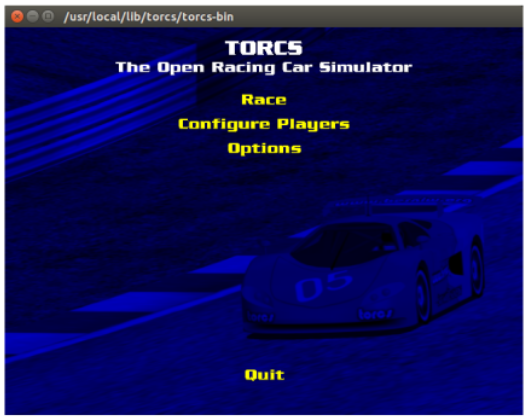

Figure 5.1: TORCS main screen

Figure 5.2: Track selection screen

Figure 5.3: Agent driving on the track

Figure 5.4: Agent goes off-track

Figure 5.5: Agent slowly getting back on track

Figure 5.1 shows the main menu of TORCS simulator . Figure 5.2 shows one of the numerous tracks which is available for selection. Figure 5.3 portrays a scenario where the agent is driving on the track. Since the agent is not perfectly trained, it sometimes deviates completely off the track as shown in Figure 5.4. However, as soon as it goes out of the track, its rewards begins to diminish. Hence it soon finds its way back to the track as shown in Figure 5.5.

After the initial training comprising of around 8000 iterations, the agent had learnt to drive on the track but it was not at all moving in a straight

line. It was swerving abruptly in its lane. As a solution to this, the learning rate was altered to give more time for the agent to learn. This resulted in the agent taking around 14000 iterations to completely re-learn driving from scratch. This is visualized in Figure 5.6. Reward obtained in each episode of training is plotted against the episode number.

Figure 5.6: Training of the agent

Figure 5.7 shows the variation of rewards with respect to speed(speedX), distance from lane centre(trackPos) and angle between lane and car(theta value). At instances where the speed is less, the rewards follow suit. At times, there are instances where in spite of having sufficiently good speed, the rewards are less. This is due to the fact that the other parameters collectively bring down the rewards as the vehicle may have deviated from the centre of the lane and may have not aligned itself properly with respect to the direction of the lane. In case the rewards are higher than the speed, it shows that the agent has received some extra reward for avoiding some form of probable collision.

Figure 5.7: Rewards vs speed, angle, and position (track 1)

Figure 5.8 further explains the dependency of speed, angle and distance from the centre for rewards. Despite the agent aligning itself properly on the track, it swerved left and right owing to high-speed cornering.

On careful observation of Figure 5.9, it can be seen that trackPos(distance of the car from the centre of the lane) is very large. In this case, it is larger than +1. This easily translates into the fact that the agent was not driving on the road but beside it. This scenario is created specifically for testing how the rewards would look if the agent is made to not drive on the road.

Before the agent was trained with an improved learning rate, it was moving in a zig-zag manner in spite of maintaining its lane. Figure 5.10 shows how the reward varies if the agent isnt properly trained to drive itself smoothly.

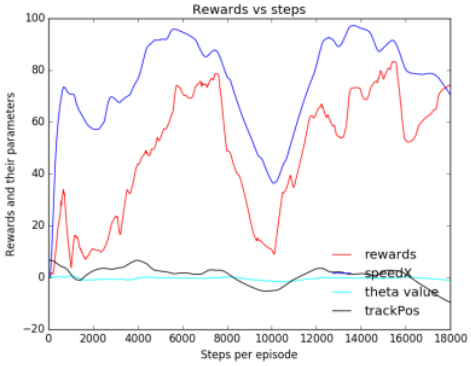

Figure 5.8: Rewards vs speed, angle, and position (track 2)

Figure 5.9: Rewards vs speed, angle, and position (track 3)

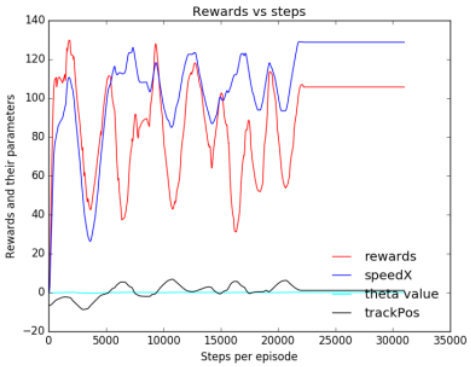

Figure 5.10: Rewards vs speed, angle, and position (track 4)

In all the simulations, the agent successfully managed to drive itself smoothly on the track by constantly interacting with the dynamic environment. It has learnt to detect the twists and turns in the track and applies brakes accordingly. It effeciently detects the obstacles and manages to take precautions or abrupt decisions to avoid collisions. Since all these cannot be depicted here, it is shown in the form of a video available through this link: Result Video Link

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

Today, self driving cars is the need of the hour because of the increasing traffic on the roads. Having self driven cars with highest possible efficiency and reliability are required very much for safe travelling. Even though many researchers work on this topic by making use of AI/ML tools, still 100% efficiency has not been achieved. In this work, one of the Reinforcement Learning algorithms, DDPG is used for controlled driving of a car. In the second phase, object detection and collision aviodance is achieved with the help of various sensors present on the car. Thus, the agent can handle situations like front, side obstacle detection, overtaking and can maneuver itself to avoid collision.

Vehicles have become an integral part of our life. Just a few decades ago, autonomous driving was just a distant dream. Over the past few years, there has been such a tremendous improvement in technology which is enabling us to bring such dreams into reality.

Elderly people, patients on medication, persons with disabilities who will have to depend on other persons for mobility can now move around independently. In addition, busy executives can attend to their work undisturbed while travelling and this saves their time. With more and more autonomous

vehicles on the road, it can be hoped that traffic accidents can be reduced to a minimum since all these vehicles can be designed to strictly follow the traffic rules.

6.2 Future Scope

This project does not take into account the changes in weather conditions, illumination levels etc. There can be some variations in the sensor values if one of more of these change and it can hamper the performance. It needs to be checked if all these sensors give accurate values after changes in weather and light. If not, it needs to be calibrated accordingly.

Since the simulator has no provision to include traffic signals, the same could not be implemented. Hence training the agent to sense the signals and act accordingly is a major for the future.

The agent heavily relies on lane markings to detect lanes. In the real world, if the lane markings are erased off in certain areas or if the lane markings are not at all there, it struggles to maintain its lane and drive efficiently. In the future, it can be trained to use a camera to detect the edge of the road and perform seamless driving.

Report

ORIGINALITY REPORT

3%	1%	2%	0%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Zhiqing Huang, Ji Zhang, Rui Tian, Yanxin Zhang. "End-to-End Autonomous Driving Decision Based on Deep Reinforcement Learning", 2019 5th International Conference on Control, Automation and Robotics (ICCAR), 2019 Publication	1%
2	huggingface.co Internet Source	1%
3	Akhil Hannegudda Ganesh, Bin Xu. "A review of reinforcement learning based energy management systems for electrified powertrains: Progress, challenge, and potential solution", Renewable and Sustainable Energy Reviews, 2022 Publication	<1%
4	mafiadoc.com Internet Source	<1%
5	Yali Yuan, Robert Tasik, Sripriya Srikant Adhatarao, Yachao Yuan, Zheli Liu, Xiaoming Fu. "RACE: Reinforced Cooperative	<1%

Autonomous Vehicle Collision Avoidance", IEEE Transactions on Vehicular Technology, 2020
Publication

6	deepai.org Internet Source	<1%
7	nbn-resolving.de Internet Source	<1%
8	Hao Li, Siddharth Misra. "Reinforcement learning based automated history matching for improved hydrocarbon production forecast", Applied Energy, 2020 Publication	<1%

Exclude quotes	On	Exclude matches	< 3 words
Exclude bibliography	On		