2024 5th IEEE Global Conference for Advancement in Technology (GCAT)
Karnataka, India. Oct 4-6, 2024

2024 5th IEEE Global Conference for Advancement in Technology (GCAT) | 979-8-3503-7668-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/GCAT62922.2024.10924132

Q-Learning Assisted Efficient Collision Detection and Avoidance Framework for Autonomous Vehicles in ITS

Aditya Shah
Dept. of Comp. Science & Engg.
Institute of Technology, Nirma University, Ahmedabad, India
21bce008@nirmauni.ac.in

Vanera Vivek
Dept. of Comp. Science & Engg.
Institute of Technology, Nirma University, Ahmedabad, India
21bce313@nirmauni.ac.in

Denisha Tank
Dept. of Comp. Science & Engg.
Institute of Technology, Nirma University, Ahmedabad, India
22bce542@nirmauni.ac.in

Rajesh Gupta
Dept. of Comp. Science & Engg.
Institute of Technology, Nirma University, Ahmedabad, India
rajesh.gupta@nirmauni.ac.in

Sudeep Tanwar
Dept. of Comp. Science & Engg.
Institute of Technology, Nirma University, Ahmedabad, India
sudeep.tanwar@nirmauni.ac.in

Usha Patel
Dept. of Comp. Science & Engg.
Institute of Technology, Nirma University, Ahmedabad, India
ushapatel@nirmauni.ac.in

***Abstract*—In the evolving landscape of Intelligent Transportation Systems (ITS), autonomous vehicles (AVs) are crucial in enhancing road safety and traffic efficiency. This paper presents a novel framework for autonomous driving systems that integrates a reinforcement learning technique known as Q-Learning to enhance decision-making in dynamic environments. The proposed framework is divided into several layers, including the Autonomous Layer, Data Filtering Layer, and Q-Learning Layer, each contributing to a robust system that can effectively manage real-time data and make adaptive driving decisions. The Autonomous Layer leverages sensors like LIDAR, radar, and heat sensors to gather environmental data, which the Data Collection Layer processes to inform the Q-learning model. The Q-learning model is then used to optimize driving policies by considering various factors such as collision avoidance, obstacle detection, and relative distance measurement. The system's performance was evaluated through simulations, demonstrating its ability to improve safety and efficiency in autonomous driving scenarios. The results indicate that integrating reinforcement learning significantly enhances the system's adaptability and reliability, offering a promising approach for future autonomous vehicle technologies.**

***Index Terms*—Autonomous Vehicles, Q-learning, Reward Function, Reinforcement Learning.**

I. Introduction

Over the last few years, there has been a transition from conventional vehicle systems to smart vehicle systems that contain numerous functionalities, including sensors, LIDAR [1], radar, and environment sensors, which help to improve the system's performance for many smart vehicles but still, they depend on the driver inputs and decision-making. The shift towards autonomous, where no input from the driver is needed, is currently receiving traction globally, and many nations are investing in such research. Countries like the United States, Germany, China, Japan, and South Korea are at the forefront of autonomous vehicle (AV) development. The United States leads with extensive testing and innovation driven by companies like Waymo, Tesla, and Apple, supported by favourable regulations in states like California. Germany, home to automotive giants like BMW and Mercedes-Benz, focuses on integrating AVs with existing transportation infrastructure. China is rapidly advancing in AV technology with strong government backing and contributions from companies like Baidu and Alibaba. Japan leverages its expertise in robotics and electronics to develop AVs, aiming for widespread deployment ahead of the 2025 World Expo. South Korea, driven by companies like Hyundai and Samsung, emphasizes smart city integration and connectivity for AVs, making significant strides in both technology and legislation. These countries are leading the global race towards the future of transportation.

AVs also present multiple advantages, such as fewer crashes, less traffic congestion, and more opportunities for those unable to drive. However, there are a lot of obstacles that must be addressed before true autonomous cars become a reality. One of the major challenges in today's AVs is the threat of multiple obstacles present around them in the dynamic environment that leads to collisions. Although various AI techniques [2] have been proposed and implemented to address different challenges in AVs—such as detecting and mitigating attacks on vehicle systems, collision avoidance remains a critical and largely unresolved problem. Previous studies in papers like [3], [4], and [5] have explored different approaches to collision detection and avoidance, but addressing these challenges while trying to simulate different road conditions remains an important and unsolved factor in the field of AVs.

Motivated by this gap, we conducted a study to understand and propose techniques for collision avoidance of AVs in a dynamic environment using Reinforcement Learning. Our research involved considering a scenario for AVs that could

be evaluated in the real world. The Reinforcement Agent [6] was trained to replicate behaviour that could help avoid various obstacles present in the environment.

A. Research Contribution

- The aim of this paper is to explain the process of creating a dynamic environment for autonomous vehicles (AVs), with particular emphasis on avoiding different obstacles.
- The potential strategy focuses on developing an appropriate reward system for a complicated environment.
- The proposed work utilizes the Q-learning algorithm, which is a type of reinforcement learning strategy.
- Various parameters are used to measure the performance of this approach.
- The findings show how the method can improve the safety and dependability of AVs in dynamic environments.

B. Paper Organization

The remainder of this paper is organized as follows: *Section II* outlines the proposed framework used in the study, this section includes the design of dynamic environment and implementation of Q-learning algorithm. *Section III* presents performance evaluation, which highlights the pseudocode of our algorithm and also discusses all the reward metrics. Finally, *Section IV* concludes this paper with findings and potential directions for future research.

II. Proposed Framework

A. Autonomous Vehicle Layer

The Layer forms one of the base-layer components in our proposed framework and focuses on data collection and interpretation of the environment. There are n number of AVs in this layer, which are represented as $\{av_1, av_2, \ldots, av_n\} \in \mathcal{A}$. Each av_i is equipped with advanced sensors $\{s_1, s_2, \ldots, s_m\} \in \mathcal{S}$, where s_i can be represented as LIDAR, radar, heat sensors, and many more. AVs keep scanning their surroundings to monitor obstacles in every respect, including road conditions and other related factors. The resultant information the sensors provide $\mathcal{S}$ is pretty rich and includes data that affects driving decisions. In this layer, the AV operates as the agent within the reinforcement learning environment $\mathcal{E}$ [8] and collects all information available within the environment like pothole distance, wind speed and many more, which is represented by $I_{\mathcal{E}}$ and information about i$^{\text{th}}$ AV by $I_{c(av_i)}$. The total information captured by an agent is described as follows.

$$T_{\mathcal{E}_{info}} = I_{\mathcal{E}} + \sum_{i=0}^{n} I_{c(av_i)} \tag{1}$$

The gathered data [9], which includes detailed spatial and environmental information like vehicle speed, obstacle distance, surrounding temperature, and wind speed, is crucial for the agent to make informed decisions. However, not all of this information is relevant to our Reinforcement Agent. The details undergo transmission to the Data Filtering Layer for processing. The Autonomous Layer takes much responsibility in implementing all those decisions manufactured by the Q-learning model: lane changing, speed adjustment, and obstacle avoidance. This layer becomes highly crucial in terms of efficiency and safety in the framework of the autonomous driving system since this layer interacts directly with real elements and dynamic changes in the environment. After the Q Learning Layer has taken the action, it is communicated to the autonomous vehicle layer, which will take the action and calculate the reward accordingly with the environment. This cycle will continue with each action and reaction pair of the agent and the environment.

B. Data Filtering Layer

The information acquired by the Autonomous Layer is stored in the Data Filtering Layer. This layer is also responsible for data processing of the profuse data collected by the autonomous vehicle layer. Once the AVs, equipped with LIDAR, radar, heat sensors, and others, have scanned the environment and interpreted it, a large amount of data is sent to this layer for further processing [10]. The data collected includes spatial data, environmental conditions, and other crucial information, such as vehicle speed, distance to obstacles, temperature, etc. Let the raw data collected from the environment be represented by:

$$D_{raw} = \{d_1, d_2, \ldots, d_k\} \tag{2}$$

where d_i represents data points like speed, temperature, or obstacle distance. The data filtering layer aims to filter only the most significant features required for decision-making in the reinforcement learning model. Let the filtered data be represented as:

$$D_{filtered} = f(D_{raw}) \tag{3}$$

where $f(\cdot)$ represents the filtering function applied to the raw data to retain only relevant features. The processed data is forwarded to the Q-learning Layer, where the reinforcement learning agent uses the filtered information to calculate the next action. All actions are finally returned to the Autonomous Layer for execution, thus completing the loop and enabling the autonomous driving system to continue learning and adapting.

C. Q-Learning Layer

The last layer of our framework is the Q-Learning Layer, which handles the decision-making and learning through the interaction of the autonomous agent with the environment. Q-learning is a model-free reinforcement learning algorithm that helps an agent learn the optimal action-selection policy by iteratively updating Q-values [11] [12], representing the expected rewards of actions in specific states. This layer aims to develop a policy which will enable the system to reap the highest reward over a number of times, given a certain state and the best action to be taken. Specific action to be performed is determined from the Q-table derived from training the agent in the specified environment using the ε-greedy policy. The Q-value is updated using the following equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{4}$$

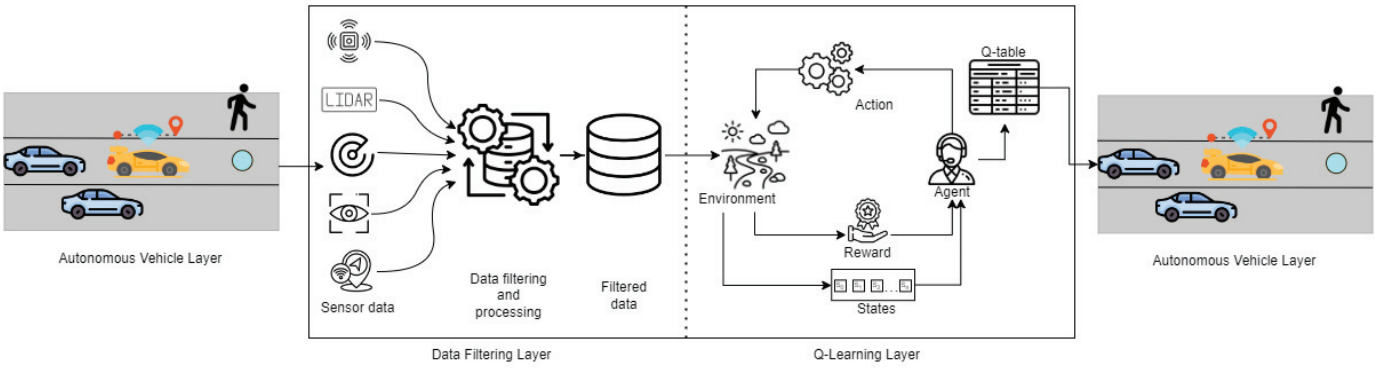

Fig. 1. Proposed Q-Learning framework [7].

where:

- $Q(s, a)$ is the current Q-value for the state s and action a.
- α is the learning rate, which determines the extent to which new information overrides old information.
- r is the reward received after taking action a in state s.
- γ is the discount factor, which determines the importance of future rewards.
- $\max_{a'} Q(s', a')$ is the maximum Q-value for the next state s' over all possible actions a'.

The agent determines the optimal action a_t for the given state s_t based on the policy π, which is updated continuously. The process can be represented as:

$$a_t = \pi(s_t) \tag{5}$$

The state information utilized by our model is categorized into the following components:

- **Agent Vehicle Information:**
  - *Lane:* The current lane in which the agent vehicle is travelling.
  - *Position:* The position of the agent vehicle along the lane.
  - *Speed:* The speed of the agent vehicle.
- **Environment Cars Information:**
  - *Lane:* The lane where each environment car travels.
  - *Position:* The position of each environment car along its lane.
  - *Speed:* The speed of each environment car.
- **Obstacle Information:**
  - *Lane:* The lane where the obstacle is located.
  - *Position:* The obstacle's position along the lane.
- **Pedestrian Information:**
  - *Lane:* The lane where the pedestrian is located.
  - *Position:* The pedestrian's position along the lane.

There are two actions that the agent is allowed to take. The first action is that it can stay in the same lane and move forward at a constant speed, and the second action is that it can change its lane and move forward at a constant speed. Based on the action decided by the Q-learning algorithm, the Q-learning layer will send this action information to the agent's car in the autonomous layer, which will behave and act accordingly. The state information of two environment cars will be stored and the reason for this is explained in the environmental setup section under performance evaluation.

Reinforcement Learning Configuration:

- **State Size:** 13
- **Action Size:** 2
- **Learning Rate (α):** 0.1
- **Discount Factor (γ):** 0.95
- **Initial Exploration Rate:** 1.0
- **Lane Change Probability:** 0.3
- **Training Episodes:** 1000
- **Testing Episodes:** 1000

One crucial aspect of reinforcement learning is the balance between exploration and exploitation, controlled by the exploration rate. In the initial episodes, the exploration rate is high, allowing the agent to explore various actions. As the training progresses, the exploration rate decays, encouraging the agent to exploit the knowledge it has gained.

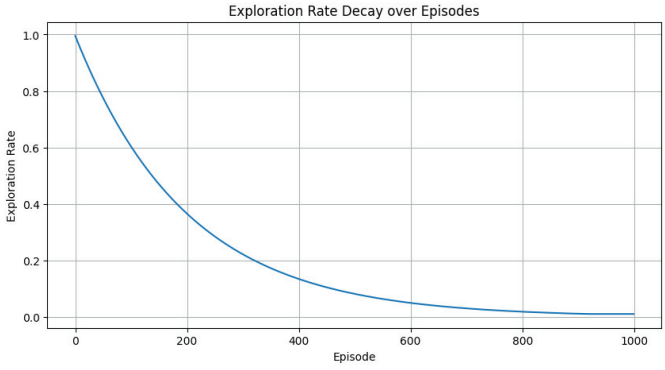

Fig. 2. Exploration decay of agent during training.

Fig. 2 illustrates the decay of the exploration rate over 1000 episodes. Initially, the exploration rate is set to 1, indicating complete exploration, but it decreases exponentially as training advances. By the end of the training, the exploration rate stabilizes near zero, allowing the agent to rely on its learned

policy. The approach begins with training, where the agent learns through 1000 episodes. When trained, the agent sends actions to the environment and receives corresponding rewards and updates the Q-table with a decreasing exploration rate to allow the exploitation of learnt values. The environment is also tested with the trained agent for 1000 episodes after training has been completed by rendering the environment state, rewards, and the number of collision events.

III. Performance Evaluation

A. Experimental Setup

Our proposed framework is implemented in Google Colab's IDE using Python 3.10.12. Our implementation uses various Python libraries like Numpy, Time, Random, and Matplotlib. We used Numpy to manipulate and store position data in the form of the 1-D array, which enables numerical computations with matrix operations. A random library provides random values needed to create randomness in the environment. At the same time, the Time library is used to measure the time taken for simulation and monitor delays. We used the Matplotlib library to display generated results in graphs. For evaluation, the total reward achieved by the agent and the number of collisions, potholes, and pedestrian hits are used as the performance measure matrices. Matplotlib is utilized to display the outcomes in relation to the agent's actions and episodes' trends.

B. Environmental Setup

In the environmental setup for our reinforcement learning-based autonomous driving simulation, we created a simulated environment consisting of a straight road with three lanes. The environment features two environment cars, each with its own position and speed, which can change lanes based on a probabilistic model. The agent, representing the autonomous vehicle, is tasked with navigating this environment while avoiding collisions with the environment cars, potholes, and pedestrians. Potholes and pedestrians are randomly placed ahead at a safe distance of the agent vehicle to introduce unpredictability and challenge. The environment cars are programmed to increase speed and can change lanes when a certain threshold distance from the agent car is maintained which is checked by collision distance function.

Additionally, these cars, potholes and pedestrians can reappear back into the environment, simulating continuous traffic without overloading computational resources, thereby allowing us to utilize the Q learning algorithm with maximum efficiency without the need for state information of more than two cars, one pothole and one pedestrian. The simulation ends after a set number of interactions that we kept as five iterations of the loop of reappearance of environmental cars. The state representation for the agent includes the discretized positions and speeds of the environment cars and agent cars and the locations of potholes and pedestrians, creating a complex, dynamic environment for testing and training the reinforcement learning model. Each episode lasted for an average of 34 steps till the 5^{th} iteration ended.

The reward distribution for the reinforcement learning agent is determined based on the type of collision or interaction. The following algorithm outlines the reward assignment based on different collision types:

Algorithm 1 Reward Distribution Algorithm for Agent Training

Input: Collision Type, Action
Output: Reward
if Collision == "Pedestrian Hit" **then**
    Reward $\leftarrow -100$
else if Collision == "Environment Car Collision" **then**
    Reward $\leftarrow -70$
else if Collision == "Pothole Hit" **then**
    Reward $\leftarrow -30$
else if Collision == "Passed Left of Pedestrian" **then**
    Reward $\leftarrow +10$
else if Collision == "Passed Right of Pedestrian" **then**
    Reward $\leftarrow -10$
else
    if Action == 0 **then**
        Lane Change Penalty $\leftarrow -1$
    else if Action == 1 **then**
        Lane Change Penalty $\leftarrow 0$
    end if
    Reward $\leftarrow$ *Lane Change Penalty*
end if
Return: Reward

C. Rewards

Fig. 3 shown above is a bar graph representation of the rewards the reinforcement learning agent obtained during the simulation phase for 1000 episodes. A bar graph, or bar chart, visualises data where rectangular bars compare individual categories. Each bar's length or height is proportional to the value it represents. From Fig. 3, we can conclude that the agent displays a large amount of variation in the rewards obtained by it. It ranges from a maximum reward of -100 to a minimum reward of -650, indicating that the agent has not fully converged to an optimal policy or that the environment's complexity is challenging for the agent to provide better rewards consistently. Overall, we can see that the agent tries several exploration approaches to improve its policy and reach an optimal policy in this dynamic environment. It suggests that further tuning of the Q-learning parameters or environment settings may be necessary to improve agent performance. Fig. 4 shows the reward values obtained by the agent during each iteration in Episode 200. This episode displays the maximum reward obtained by the agent in the 1000 episodes.

1) Initial Phase (Iterations 0-5): The agent starts with a reward of 0. We can see that it changes its lanes for a few steps and then collides with a pedestrian in its lane, thereby receiving a -10 reward to its total reward.

2) Middle Phase (Iterations 6-15): The rewards continue to decrease as the agent continuously changes lanes, avoiding

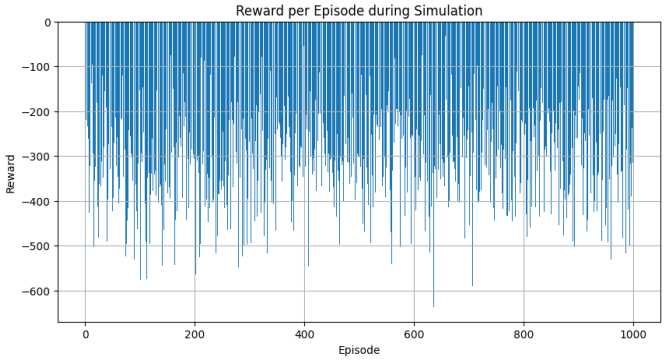

Fig. 3. Reward obtained per episode during simulation.

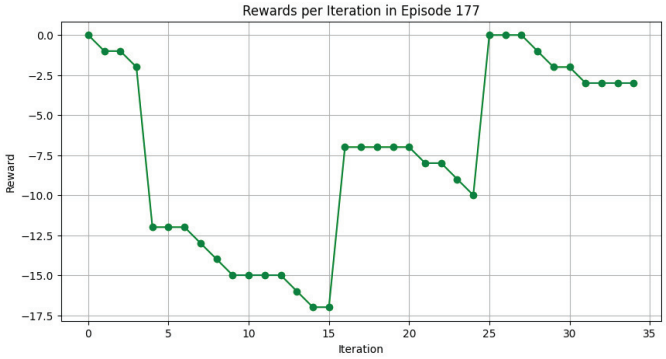

Fig. 4. Maximum reward obtained in the best simulation episode.

a collision. This is its optimum behaviour, and we want it to continue and avoid obstacles as much as possible.

3) Recovery Phase (Iterations 16-25): The agent sees its state information and concludes that a pedestrian will come in its way. It then decides to go through the left side of the pedestrian to gain a positive reward. After accomplishing this task, the agent continues avoiding obstacles, per its state information. The agent learns this policy and again gains a positive reward by going through the left side of a pedestrian at iteration 25.

4) Final Phase (Iterations 26-34): After the peak, the reward drops slightly and stabilizes around -3, suggesting the agent changed its lanes a few times while trying to avoid collisions. The reward maintains a consistent value, indicating that the agent has settled into a specific strategy, though it could still be suboptimal.

D. Environmental Car Collisions

Fig. 5 specifies the total amounts of collisions present in terms of the episode in several one thousand episodes throughout the simulation. The graph shows that collisions take place steadily all through the episodes, and maximum episodes depict a minimum of one collision. The magnitude of collision is between 1 and four for most of the episodes, although there are occasional surges which may perform up to five or six collisions. The agent collides with almost two environmental cars for most of the episodes in our simulation. This may give perception to the present-day functionality of the agent for reducing collisions to determine the place for improvements either in the agent's studying set of rules or in the layout of its surroundings

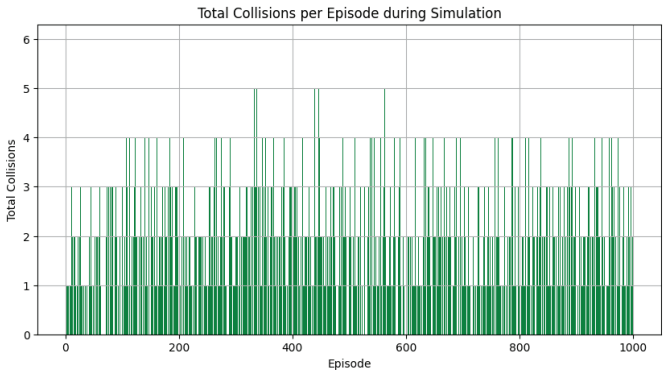

Fig. 5. Collisions encountered per episode during simulation.

E. Pedestrians

Fig. 6 shows the graph of the number of episodes versus the number of pedestrian hits that our Q-learning agent detected during each episode for 1000 episodes of the simulation. The graph shows the frequency of pedestrian hits, ranging from 0 to 4 hits. The results of the data analysis show that pedestrian hits occur often, with many episodes recording at least 1 hit. While the majority of episodes show 1 or 2 hits, there are very few occasional episodes with more hits, which shows the agent's difficulty in trying to avoid pedestrians throughout the simulation. However, we can see that the majority of pedestrians hit are 1, indicating that the model is trying to avoid pedestrians as many times as it can. Prioritizing it the most, the agent tries applying an optimum policy in a complex environment. This graphs shows that the agent has some deficiency, Specifically, the agent has to pay more attention to the safety aspect while training an autonomous system. The reason for this is because while the penalty for collision of car is less, the number of environmental cars are 2, thereby increasing the total penalty for hitting the environmental cars.

F. Potholes

Fig. 7 presents the distribution of pothole hits faced by the Q-learning agent at some point of each episode throughout one thousand episodes of the simulation. The graph captures the relative number of pothole hits accurately as a function of episodes ranging from zero to four hits. The graph proves that the agent often hits potholes, with all recorded episodes having at least one strike. As characteristic of most episodes with values of one to two pothole hits, there are random episodes where the agent hits more potholes, showing varied efficiency in avoiding the obstacles. These frequent and diverse pothole hits also tell us that the agent is not always in the best place to avoid potholes. It prioritizes hitting the potholes over hitting any other obstacle (in a situation when it has to avoid either

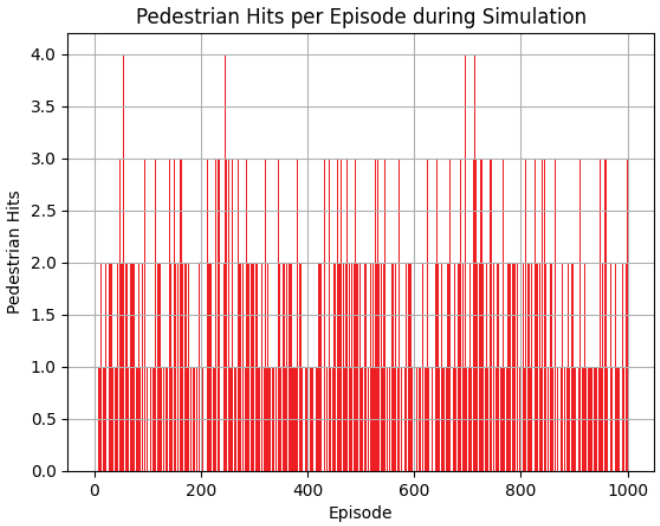

Fig. 6. Pedestrians Encountered per Episode during Simulation.

one) as it would receive less penalty in doing so. The agent may need additional education or, in addition, optimisation of the reward system to further avoid potholes and thus improve the effectiveness of the simulation. This graph justifies the application of similar adjustments inside the agent's learning interaction in order to minimize the counts of pothole hits and optimize environment navigation.

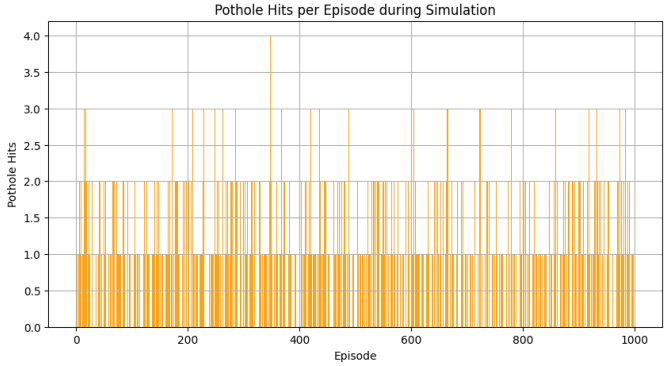

Fig. 7. Potholes encountered per episode during simulation.

Overall, based on the data observed, it can be seen that the agent's behaviour is inclined to the reward structure provided by us, which includes avoiding pedestrians having the most priority as one of the key factors. The agent would require more training to grasp the complex nature of the environment.

IV. Conclusion

In this paper, we construct a reinforcement learning scheme to automatically steer vehicles in a simulated setting by Q-learning. Our agent has been tested on various traffic lanes, potholes and other vehicles which are close to the real environment. The demonstration presented within the experiment exposed the degrees of roads' recognition, collision avoidance, and obstacles handling of the agent. The area of reinforcement learning for autonomous driving is benefited by the present work as it shows how decision can be made by agents in environments. The future work includes fine-tuning the proposed framework, trying with some of the more complex algorithms such as Deep Q-learning (DQN) or DDQN, and maximizing the efficiency.

References

[1] T. Asano and H. Toda, "Basic experiment of lidar sensor measurement directional instability for moving and vibrating object," in *2022 2nd International Conference on Robotics, Automation and Artificial Intelligence (RAAI)*, pp. 148–151, 2022.

[2] T. Islam, M. A. Sheakh, A. N. Jui, O. Sharif, and M. Z. Hasan, "A review of cyber attacks on sensors and perception systems in autonomous vehicle," *Journal of Economy and Technology*, vol. 1, pp. 242–258, 2023.

[3] Q. Deng, Y. Zhao, R. Li, Q. Hu, T. Zhang, H. Zhang, and R. Li, "A pseudo-hierarchical planning framework with dynamic-aware reinforcement learning for autonomous driving," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, pp. 2345–2352, 2024.

[4] B. Gangopadhyay, P. Dasgupta, and S. Dey, "Safe and stable rl (s2rl) driving policies using control barrier and control lyapunov functions," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 2, pp. 1889–1899, 2023.

[5] M. Hell, G. Hajgato, A. Bogar-Nemeth, and G. Bari, "A lidar-based approach to autonomous racing with model-free reinforcement learning," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, pp. 258–263, 2024.

[6] M. M. Abbas and M. N. Mladenović, "Agent-based control for adaptive high performance connected vehicle streams," in *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, pp. 947–948, 2013.

[7] Flaticon, "https://www.flaticon.com/." http://aiweb.techfak.uni-bielefeld.de/content/bworld-robot-control-software/, 2008. [Online; accessed 19-July-2008].

[8] E. Jebessa, K. Olana, K. Getachew, S. Isteefanos, and T. K. Mohd, "Analysis of reinforcement learning in autonomous vehicles," in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0087–0091, 2022.

[9] J. Gu, A. Lind, T. R. Chhetri, M. Bellone, and R. Sell, "End-to-end multimodal sensor dataset collection framework for autonomous vehicles," *Sensors*, vol. 23, no. 15, 2023.

[10] I. Sumalatha, P. Chaturvedi, G. R. R, S. Patil, H. P. Thethi, and A. A. Hameed, "Autonomous multi-sensor fusion techniques for environmental perception in self-driving vehicles," in *2024 International Conference on Communication, Computer Sciences and Engineering (IC3SE)*, pp. 1146–1151, 2024.

[11] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep q learnig," in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, pp. 201–205, 2018.

[12] H. Guo, M. K. Tan, K. G. Lim, H. S. Ee Chuo, B. Yang, and K. T. Kin Teo, "Improved q-learning algorithm for path planning of an automated guided vehicle (agv)," in *2023 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pp. 376–381, 2023.