Development of Reinforcement Learning Based Autonomous Vehicle Platoon for Energy Efficient Operation

MAE 598 : Multi Robot Systems(Fall 2020)

Vivek Verma
Master of Science in Mechanical Engineering
Arizona State University
Tempe, Arizona
vverma14@asu.edu
1217211710

Nishanth Solomon
Master of Science in Robotics and Autonomous Systems
Arizona State University
Tempe, Arizona
nsolomo2@asu.edu
1217601723

Jayasurya Sevalur Mahendran
Master of Science in Robotics and Autonomous Systems
Arizona State University
Tempe, Arizona
jsevalur@asu.edu
1217399443

ABSTRACT

Recent developments in the field of Autonomous Vehicles have been groundbreaking. Autonomous vehicles offer higher potential to drastically reduce traffic accidents, increase productivity by saving time and energy and minimize the global-warming impact on the environment [17]. In this project, we have designed a Reinforcement Learning(RL) based Autonomous Agent[10] for transportation in CARLA [4].Despite reinforcement learning algorithms have achieved notable results in games and some robotic manipulations, this technique has not been widely scaled up to the more challenging real world applications like autonomous driving. The Multi-robot behaviours that we have implemented is platooning [14], characterized by the string of vehicles traveling with small separation distances. Only the first vehicle is driven by a Reinforcement Learning agent while the followers are assisted by the control system in our case we have designed a custom combination of the Proportional Integral Derivative (PID) controller [16] for the longitudinal control that is braking and acceleration and Stanley controller [12] for lateral control to control the steering of the vehicle. The main goal of this approach is to reduce fuel consumption and greenhouse gas emissions. This is achieved because the proximity of the vehicles in this configuration provides a more efficient airflow around the set of vehicles reducing the overall energy consumption. In our case we have two vehicles namely the leader vehicle which is an Autonomous agent driven by RL and a single follower agent. Though CARLA does not support platooning naturally like other simulators such as TORCS [5], we have implemented platooning using client-server communication functionality, the leader vehicle communicates the set of way points and the desired speed to the follower vehicle through a socket connection along with a time-stamp. In the follower vehicle the controller tries to adjust its steering and acceleration to closely follow the input way-points with the desired speed. The proposed Multi-Robot application area is transportation of goods keeping autonomous trucks in mind and the proposed Multi-Robot behaviour is energy efficient goal oriented motion using platooning.

Keywords: Autonomous Vehicle, Connected Vehicles, Reinforcement Learning, CARLA, platooning, Energy efficiency, PID, Stanley Controller

1 MATHEMATICAL MODEL

1.1 Reinforcement Learning

Reinforcement Learning is an important approach for goal-oriented optimization which is inspired from behaviorist psychology [22]. An RL agent learns through interactions with its environment, which in-turn receives feedback in the form of rewards. The returned reward reinforces the agent to select new actions improving learning process, hence the name of reinforcement learning [19].RL algorithms have achieved notable results in many Atari game domains [15] and go game [20]. RL algorithms have also achieved super human results [13]. Even after after the promising 2007 DARPA Urban Challenge, Autonomous driving is one of the current highly challenging tasks that is still an "unsolved problem".Standard RL algorithm randomly explore all the states and learn faced problems lose efficiency and become computationally intractable when dealing with high-dimensional and complex environments. In our project we built over an advantage actor critic approach with multi-step returns for autonomous driving. This type of RL has demonstrated good convergence performance and faster learning in several applications which make it among the preferred RL algorithms [9]. Actor-critic RL consolidates the robustness of the agent learning strategy by using a temporal difference (TD) update to control returns and guide exploration. The training and evaluation of the approach are conducted with the recent CARLA simulator. Designed as a server-client system, where the server runs the simulation commands and renders the scene readings in return, CARLA is an interesting tool since physical autonomous urban driving generates major infrastructure costs and logistical difficulties. It particularly offers a realistic driving environment with challenging properties variability as weather conditions, illumination, and density of cars and pedestrians.

1.1.1 ***CARLA SIMULATOR***.

CARLA has been developed from the ground up to support development, training, and validation of autonomous driving systems. In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles) that were created for this purpose and can be used freely. The simulation platform supports flexible specification of sensor suites, environmental conditions, full control of all static and dynamic actors, maps generation and much more.

Some of the key features of CARLA that was helpful for us were:

- Maps generation: users can easily create their own maps following the Open-Drive standard via tools like RoadRunner.
- Flexible API: CARLA exposes a powerful API that allows users to control all aspects related to the simulation, including traffic generation, pedestrian behaviors, weathers, sensors, and much more.
- Autonomous Driving baselines: we provide Autonomous Driving baselines as runnable agents in CARLA, including an Auto-Ware agent and a Conditional Imitation Learning agent.

Figure 1: CARLA simulator

1.1.2 *A3C algorithm*.

The Asynchronous Advantage Actor Critic (A3C) algorithm is a modern day algorithm developed by Google's DeepMind. This algorithm was first mentioned in 2016 in a research paper appropriately named Asynchronous Methods for Deep Learning.
The algorithm consists of majorly :

- **Asynchronous Property**: Unlike other popular Deep Reinforcement Learning algorithms like Deep Q-Learning which uses a single agent and a single environment, This algorithm uses multiple agents with each agent having its own network parameters and a copy of the environment. This agents interact with their respective environments Asynchronously, learning with each interaction. Each agent is controlled by a global network. As each agent gains more knowledge, it contributes to the total knowledge of the global network. The presence of a global network allows each agent to have more diversified training data. This setup mimics the real-life environment in which humans live as each human gains knowledge from the experiences of some other human thus allowing the whole "global network" to be better.
- **Actor-Critic**: Unlike some simpler techniques which are based on either Value-Iteration methods or Policy-Gradient methods, the A3C algorithm combines the best parts of both the methods, the algorithm predicts both the value function as well as the optimal policy function. The learning agent uses the value of the Value function (Critic) to update the optimal policy function (Actor). The policy function in this case is the probabilistic distribution of the action space. The learning agent determines the conditional probability of the state s.

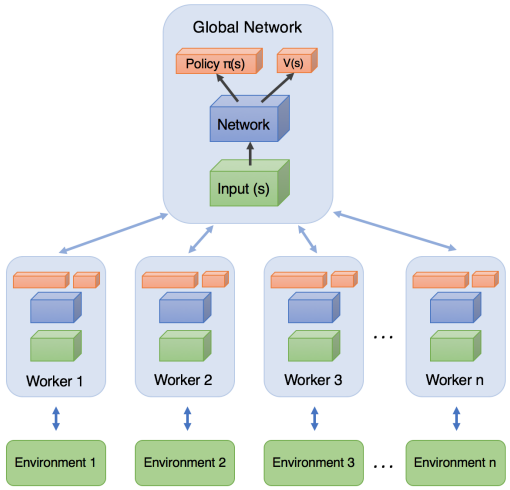


Figure 2: A3C algorithm

1.1.3 *CARLA Dataset*.

The model was pre-trained in the KITTI Vision Benchmark Suite [6] for the Town Map 01 and Town Map 02. Later we added the manually created dataset for Racetrack and trained the model over the top by adding a linear layer. The performance measures were analyzed and the experimental results below in the experimental analysis section.

Figure 3: Carla Dataset

1.1.4 *RL Model*.

Markov Decision Process (MDP) T_i is defined according to the tuple (S, A, p, r, , 0, H) where S is the total number of states of states, A is the action set, $p(s_{t+1}|s_t, a_t)$ is the state transition distribution predicting the probability to reach a state s_{t+1} in the next time step given current state and action, r is a reward function, is the discount factor, $_0$ is the initial state distribution and H the horizon. Consider the sum of expected rewards (return) from a trajectory $\tau_{(0,H1)} = (s_0, a_0, ..., s_{H-1}, a_{H-1}, s_H)$. The major objective of the RL is to learn a policy π of parameters θ that maps each state s to an optimal action a maximizing the reward R.

$$R_t = r_{t+1} + \gamma R_{t+1} = \sum_{i=t}^{t+H-1} \gamma^{i-t} r_{i+1} \tag{1}$$

The state value function V (s) can be defined including the reward as V (s): S → R and a state-action value function Q(s, a) can be

defined as Q(s,a) : A × S → R to measure the current state and state-action returns estimated under policy π:

$$V(s_t) = \mathbb{E}[R_t|s_t = s]$$
$$Q(s_t, a_t) = \mathbb{E}[R_t|s_t = s, a_t = a] \tag{2}$$

In policy-based methods, gradient descents are used to directly optimize a parameterized policy without using a value function.

$$\Delta\theta = \alpha\nabla_\theta \log \pi_\theta(s_t|a_t)R_t \tag{3}$$

In actor-critic framework is to replace the Reward function R_t in the policy gradient with the action value function that will enable the agent to learn the long-term value of a state and therefore enhance its prediction decision:

$$\Delta\theta = \alpha\nabla_\theta \log \pi_\theta(s_t|a_t)Q(s_t, a_t) \tag{4}$$

The Monte Carlo method and dynamic programming are combined to produce:

$$G_t = r_t + \gamma * V_t(s_{t+1}) \tag{5}$$

The one step TD update rule that allows the adjustment of the value function is given as:

$$V(s_t) = V(s_t) + \beta(\underbrace{r_t + \gamma V_t(s_{t+1}) - V(s_t)}_{\delta_t}) \tag{6}$$

The improvement in predicting an action compared to the average $V(s_t)$ is defined as the advantage function $A(s_t, a_t)$

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \tag{7}$$

$A(s_t, a_t)$ is defined as the difference between the expected future reward and the actual reward that the agent receives from the environment.

$$A(s_t, a_t) = R(s_t, a_t) - V(s_t) \tag{8}$$

The advantage of the actor policy gradient is got by using the above equation in policy gradient.

$$\Delta\theta = \alpha\nabla_\theta \log \pi_\theta(s_t|a_t)(G_t - V(s_t)) \tag{9}$$

The final actor policy gradient is deduced by assuming T D error as a good candidate to estimate the advantage function.

$$\Delta\theta = \alpha\nabla_\theta \log \pi_\theta(s_t|a_t)\delta_t \tag{10}$$

By defining the configurable multi-step returns within the TD Target algorithmically, we are able to deduce the TD error as:

$$\delta_t = [\sum_{i=t}^{t+H-1} \gamma^{i-t} r_i] + \gamma^H V(s_{t+H}) - V(s_t) \tag{11}$$

1.2 PID and Stanley Controller

A PID controller is a control loop mechanism which employs feedback. It is widely used in control systems and a plethora of other applications that requires continuous modulated control. The basic idea behind it is to read a sensor value, then solve for the desired output by calculating the proportional, integral, and derivative responses and then summing the three components to find the output.[18] In a control system, process variable is a system parameter that needs to be controlled. It can be any variable based on specific systems. To measure that process variable a sensor is used and it also provides a feedback to the control system.In our project we used PID for longitudinal control of the vehicle. Below is an example of a closed loop system.[8]

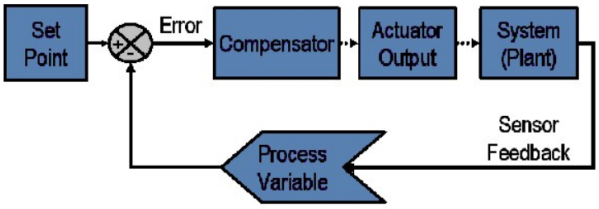


Figure 4: Closed Loop

At any given moment in time, the difference between the process variable and the reference value is used by the control system to determine the desired controller output to drive the system. This is called a closed loop control system, since the process of reading the sensor's output to provide constant feedback and calculating for the desired output is a continuous process and takes place at a fixed loop rate.[3] The proportional aspect is dependant only on the difference between the reference value and the current value of the state of the system. The gain associated with it is called proportional gain *Kp*. The integral component sums the error term over time, which results in even a small error term causing the integral to slowly increase. Integral response slowly increases with time if the error is not zero, hence the effect is to drive the steady-state error to zero. So the aim is to drive that error to zero. The gain associated with it is called integral gain *Ki*. The derivative aspect is responsible for the output to decrease if there is a rapid increase in the process variable. The derivative component is proportional to the rate of

change of the process variable. Changes in the derivative time *Td* is responsible for the system's response to change in the error term. For example using a higher derivative time will cause the system to reach more strongly to changes in the error. This parameter is highly sensitive to the noise in the signal in process variable. If the sensor feedback signal or if the rate of control loop is slow, the response of the derivative can make the control system unstable. The gain associated is called derivative gain *Kd*[11].

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{12}$$

Equation (1) represents the PID control law.

Stanley controller is used in our project for lateral control of the vehicle. The control law associated can be defined as shown below

$$\phi(t) = \theta_p(t) + tan^{-1}\left(\frac{d_f(t)}{l_d}\right) \tag{13}$$

Equation (2) represents the Stanley control law.

The control law consists of two terms. The first term is to compensate for the angular error θp while the second term is to compensate for the front lateral distance d_f which is measured from the center of the front axle to the closet point on the path. This very error is also considered as the angular error θt. l_d is the headway distance and is generally chosen proportional to the speed of the vehicle.[12] Below is an example of the representation of the parameters discussed above

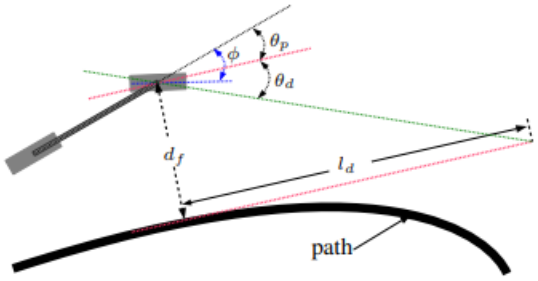


Fig. 3. Stanley control

Figure 5: Stanley Parameters Representation

Many advanced versions of the Stanley controller have also been developed on top of the existing one like Stanley controller with yaw compensation, modified Stanley controller and have been used in vehicles for different racing competitions and researches[12].

2 THEORETICAL ANALYSIS

2.1 Inter-Vehicle distance

The platoon consists of N vehicles moving at the same speed v_d with a desired inter distance L between each set of two vehicles. The leader of the platoon is driven by the RL agent. The followers are controlled to maintain the desired inter-distance[2].

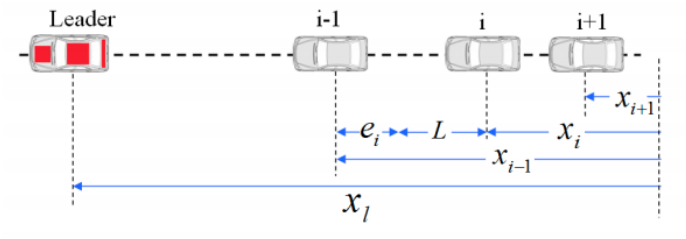


Figure 6: Platoon

The constant spacing error is given by

$$e_i = \Delta X_i - L \tag{14}$$

where,

$\Delta X_i = x_{i-1} - x_i$ real spacing between i^{th} car and its predecessor $(i-1^{th} car)$

x_i is the position of i^{th} vehicle

L is the desired inter-vehicle distance

$$\dot{e_i} = \dot{x_{i-1}} - \dot{x_i} = v_{i-1} - v_i \tag{15}$$

The main objectives of the control law is to keep the inter-vehicle distance equal to L, and to make all vehicles move at the same speed so $\dot{e_i} = 0$. Figure 7 shows the theoretical Inter-Vehicle distance vs time graph which is got through MATLAB simulation. As optimal behavior of the platoon, the inter-vehicle distance must be constant after reaching the steady-state. This is an important behavior in order to get the benefits of platooning. The fuel efficiency increases only when the air drag reduces, and the air drag reduces only when the vehicles are traveling with constant inter-vehicle distance. We have analyzed the steady-state inter-vehicle distance in our experiments.

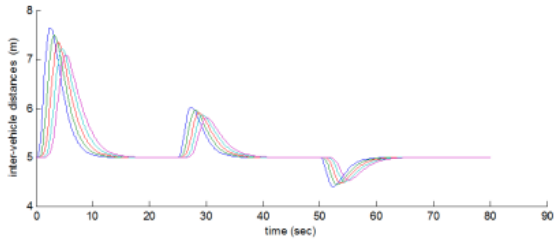
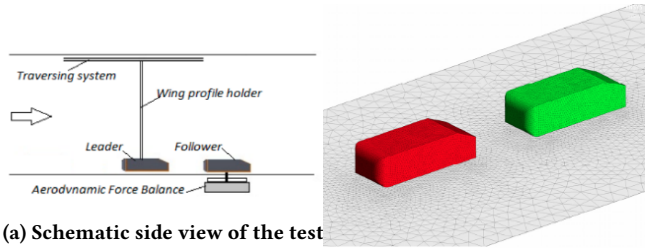


Figure 7: Inter-Vehicle distance vs time graph

2.2 Aerodynamic Parameters

From the paper *The influence of distance between vehicles in platoon on aerodynamic parameters*[7], the reduction of fuel consumption of vehicles driving in platoon arrangement is proved. The research



(a) Schematic side view of the test section

(b) Computational domain with Ahmed bodies

Figure 8: Desired Trajectory vs Actual Trajectory

concerns numerical and experimental investigations of the pressure and velocity fields as well as other aerodynamics parameters between two vehicles in platoon arrangement. CFD simulations are performed based on the steady-state Reynolds-Averaged Navier–Stokes equations with the standard k– model and standard wall function. The simulations are validated based on a series of wind tunnel measurements. Both CFD simulations and wind tunnel measurements allowed ascertaining and quantifying the upstream effect by a downstream car on the drag of the first one. Complex flow fields have been numerically visualized showing various velocity and pressure fields formed at different regions. The flow structures in the wake region behind the downstream Ahmed vehicle is more or less the same for all the analyzed cases (x/L = 0.2-1.2). Comparison of various flow characteristics revealed that a significant change in ow structure between the vehicles in the platoon with 0.3<x/L<1.2 is the underlying reason for the increase in the drag coefficient.

We have used the result in this paper to determine the inter-vehicle distance. The x/L value selected was 1 for the experiments. But, Carla doesn't support analyzing of aerodynamics parameter. Hence we theoretically prove that the platooning with 0.3<x/L<1.2 significantly reduces the air drag and hence improving the fuel efficiency of the vehicles in the platoon.

3 EXPERIMENTAL ANALYSIS

Now we have an RL model that was re-trained on the KITTI dataset. We have formulated an experimental setup to test the RL model and platooning behavior.

(1) Straight: Destination is straight ahead of the starting point
(2) One turn: Destination is one turn away from the starting point

The RL agent is deployed in the above scenarios, and the way-points of the RL agent is given as the output. The RL agent is considered as the Lead vehicle. This set of way-points is simultaneously communicated to the follower agent using a socket connection. The follower agent is a combination of a PID controller and a Stanley controller which takes the way-points as input and generates controls for the follower agent in order to follow the lead vehicle. With the lead agent moving in the deployed scenario the follower agent should be able to follow with the following parameters. The distance between the two cars must be constant, the follower vehicle must be able to move in the same trajectory which the lead vehicle has taken. Hence the speed, steering angle, braking, and throttle must be almost the same. Here we analyze the steady-state distance between the two agents, the comparison between the trajectory, the speed, steering response, braking amount, and throttle position.

3.1 Steady-state distance

The normal distance between two agents is suggested to in the range of 0<(x/L)<1.2[7], where x is the distance between two vehicles and L is the length of the vehicle. Considering the average vehicle length of 4.5m and selecting the x/L ratio as 1, we fixed the desired steady-state distance to be 4.5m. To get the steady-state distances, we took the trajectory of the lead and follower vehicle indexed over time. Figure 9 shows that the initial distance between the two vehicles is about 10m and when the vehicles started moving the follower vehicle moved closer to the lead vehicle and started to maintain at a steady-state distance.

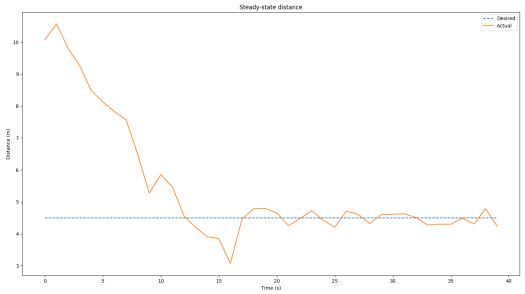


Figure 9: Steady-state Distance for Race Track map

Hence we can conclude that the inter-vehicle distance is maintained at a constant value (in our case its 4.5m) and hence the platooning has worked properly.

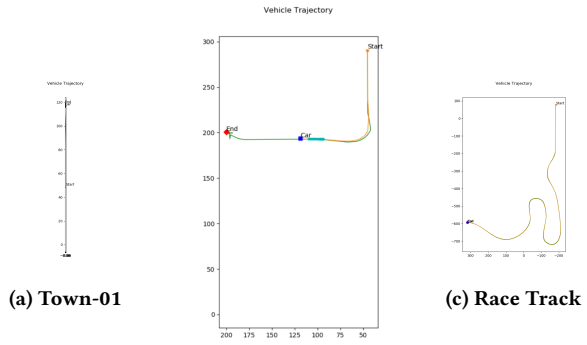
3.2 Trajectory

The controls should have been designed for the follower vehicle in such a way that the vehicle follows a time parameterized reference i.e., a geometric path with an associated timing law[1]. The Lead vehicle sends the geometric path information as waypoints which are indexed over time. Now the follower vehicle having this information should be able to follow the same trajectory. We plot the trajectory taken by the lead vehicle and the trajectory taken by the follower vehicle in figure 10.

Hence the follower vehicle follows the lead vehicle in the same waypoints that the lead vehicle has taken.

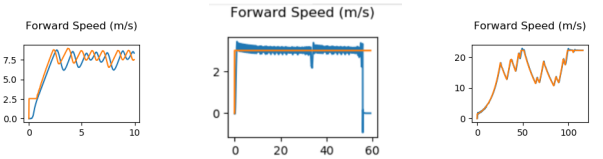
3.3 Speed

All the vehicles in the platoon move at the same speed. The follower vehicle must have the same speed as the lead vehicle. If the speed of the follower vehicle increases more than the speed of the lead vehicle, then the desired distance between the two vehicles decreases ending up in accidents. And if the speed of the follower vehicle decreases, the distance between the vehicles would be high resulting in more air drag, and hence less fuel efficiency.



(a) Town-01 (b) Town-02 (c) Race Track

Figure 10: Desired Trajectory vs Actual Trajectory

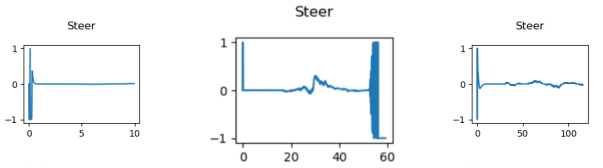


(a) Town-01 (b) Town-02 (c) Race Track

Figure 11: Speed vs. Time Graph (Orange:Lead Vehicle; Blue:Follower Vehicle)

3.4 Steering

The Stanley controller takes the waypoints as input and it generates the steering angle. The steering angle is fed to the vehicle controller. We have plotted the steering angle vs. the time graph.

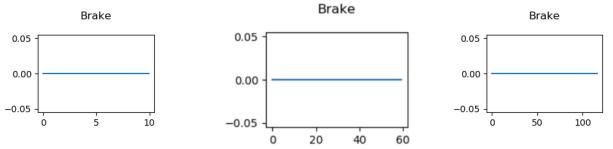


(a) Town-01 (b) Town-02 (c) Race Track

Figure 12: Steering Angle vs. Time

3.5 Braking

One of the important features of platooning is that the vehicles move very close to each other. When vehicles move very close to each other the distance required to brake also needs to be reduced. The brakes cannot be applied once the agent behind senses the lead vehicle is braking through perception. Hence we implement vehicle to vehicle (V2V) communication[21]. The lead vehicle communicates the braking information to the follower vehicle and the follower vehicle starts braking as required. We have plotted the Braking profile graph of the follower vehicle.

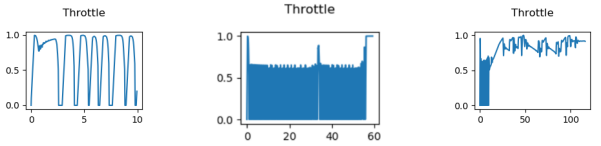


(a) Town-01 (b) Town-02 (c) Race Track

Figure 13: Braking vs. Time

3.6 Throttle

The PID controller takes the waypoints and the desired speed as input and it generates the Throttle response required. The throttle is fed to the vehicle controller. We have plotted the required throttle vs. the time graph.



(a) Town-01 (b) Town-02 (c) Race Track

Figure 14: Throttle vs. Time

4 RESULT DISCUSSION AND FUTURE WORK

We were able to complete whatever we had mentioned in the project proposal. We successfully implemented the A3C algorithm based Reinforcement Learning agent, which was trained by us on the race track dataset. The RL agent performed quite well in the Racetrack map and also gave reasonable performance in the Town Map 01 and Town Map 02. The follower agent for the platooning was designed using the PID and Stanley controller namely for longitudinal and lateral control respectively, which performed optimally to follow the set of way-points communicated by the leader agent through socket connection in CARLA. The results were as expected, however when training the model we had to eliminate the traffic and other agents such as pedestrians to simplify the model. The follower agent followed the trajectory of the leader agent quite closely in the scenarios of Racetrack and Town Map 01, however it got crashed in Town Map 02. Upon achieving the stability the follower agent followed the leader agent maintaining a constant distance between them also meeting the expected braking and steering characteristics. Platooning would be able to increase the fuel efficiency and hence reduce the carbon emission making it more efficient and eco-friendly.

For future work we would like to implement the Multi client behaviour of CARLA so that the leader agent and the follower agent can be placed in the single simulation window, also training the model for complex experimental setup so that the RL agent will be robust enough to run when deployed in any setup. We would also like to explore the fuel consumption analysis, air drag analysis of the vehicle which was not currently possible in Carla.

ACKNOWLEDGMENTS

We would like thank professor Spring Berman, for her constant support and valuable inputs during the entire semester, that helped us to complete the project on time. We would also like to thank Shenbagaraj Kannapiran (shenbagaraj@asu.edu), Graduate Research Associate, SEMTE ASU for taking time and resources for guiding us when we were stuck at different instances during the project.

REFERENCES

[1] A. P. Aguiar and J. P. Hespanha. 2007. Trajectory-Tracking and Path-Following of Underactuated Autonomous Vehicles With Parametric Modeling Uncertainty. *IEEE Trans. Automat. Control* 52, 8 (2007), 1362–1379. https://doi.org/10.1109/TAC.2007.902731

[2] Alan Ali, Gaëtan Garcia, and Philippe Martinet. 2013. Minimizing the Inter-vehicle Distances of the Time Headway Policy for Platoon Control on Highways., Vol. 2. 417–424.

[3] P. Cominos and N. Munro. 2002. PID controllers: recent tuning methods and design to specification. *IEE Proceedings - Control Theory and Applications* 149, 1 (2002), 46–53. https://doi.org/10.1049/ip-cta:20020103

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*. 1–16.

[5] E. Espié, Christophe Guionneau, Bernhard Wymann, Christos Dimitrakakis, Rémi Coulom, and A. Sumner. 2005. TORCS, The Open Racing Car Simulator.

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[7] Renata Gnatowska and Marcin Sosnowski. 2017. The influence of distance between vehicles in platoon on aerodynamic parameters. *Experimental Fluid Mechanics 2017* 180 (01 2017), 194–198. https://doi.org/10.1051/epjconf/201818002030

[8] P. V. Gopi Krishna Rao, M. V. Subramanyam, and K. Satyaprasad. 2014. Study on PID controller design and performance based on tuning techniques. In *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*. 1411–1417. https://doi.org/10.1109/ICCICCT.2014.6993183

[9] Ivo Grondman, Lucian Busoniu, Gabriel Lopes, and Robert Babuska. 2012. A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics* 42 (11 2012), 1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595

[10] Zhaoyuan Gu, Zhenzhong Jia, and Howie Choset. 2019. Adversary A3C for Robust Reinforcement Learning. *CoRR* abs/1912.00330 (2019). arXiv:1912.00330 http://arxiv.org/abs/1912.00330

[11] O. Hanif and V. Kedia. 2018. Evolution of Proportional Integral Derivative Controller. In *2018 International Conference on Recent Innovations in Electrical, Electronics Communication Engineering (ICRIEECE)*. 2655–2659. https://doi.org/10.1109/ICRIEECE44171.2018.9008628

[12] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun. 2007. Autonomous Automobile Trajectory Tracking for Off-Road Driving: Controller Design, Experimental Validation and Racing. In *2007 American Control Conference*. 2296–2301. https://doi.org/10.1109/ACC.2007.4282788

[13] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2015. End-to-End Training of Deep Visuomotor Policies. *CoRR* abs/1504.00702 (2015). arXiv:1504.00702 http://arxiv.org/abs/1504.00702

[14] André Mendes, A.T. Fleury, Marko Ackermann, and Fabrizio Leonardi. 2017. Heavy-duty Truck Platooning: A Review. https://doi.org/10.26678/ABCM.COBEM2017.COB17-0843

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (02 2015), 529–33. https://doi.org/10.1038/nature14236

[16] Robert Paz. 2001. The Design of the PID Controller. (01 2001).

[17] M. Ryan. 2020. The Future of Transportation: Ethical, Legal, Social and Economic Impacts of Self-driving Vehicles in the Year 2025. *Science and Engineering Ethics* 26 (2020), 1185 – 1208.

[18] Axel Sauer, Nikolay Savinov, and Andreas Geiger. 2018. Conditional Affordance Learning for Driving in Urban Environments. arXiv:cs.RO/1806.06498

[19] Joni Schwartz and Reem Jaafar. 2018. Jaafar, R. Schwartz, J. (2018). Applying holistic adult learning theory to the study of calculus. Journal of University Teaching Learning Practice.15:3. (11 2018).

[20] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature* 529, 7587 (jan 2016), 484–489. https://doi.org/10.1038/nature16961

[21] Dongliang Su and Sanghyun Ahn. 2017. In-vehicle sensor-assisted platoon formation by utilizing vehicular communications. *International Journal of Distributed Sensor Networks* 13, 7 (01 Jul 2017), 1550147717718756. https://doi.org/10.1177/1550147717718756

[22] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

A APPENDIX- DIVISION OF LABOR

Team Member	**Project Assigned Task**	**Report Assigned Sections**
Vivek	1.Creating a CARLA environment for training the agent, 2. collecting characteristics data, 3. Running the A3C Reinforcement Learning Agent	Equal contribution for all the sections
Nishanth	1.Pre-training the RL model for the Racetrack dataset, 2. Implementing the socket connection in CARLA for Leader follower communication 3. Implementation of platooning	Equal contribution for all the sections
Jayasurya	1.Creating the controller based Agent (PID controller for Longitudinal Control and Stanley Controller for Lateral Control), 2.Collecting characteristics graphs for different scenarios 3. Implementation of platooning	Equal contribution for all the sections

B LINK TO THE DRIVE FOLDER CONTAINING ALL THE FILES

Google Drive link
https://drive.google.com/drive/folders/1Yh-rBjR6SlGski2nbfIERRWL-8su9pCK?usp=sharing

The readme file is provided inside the drive folder.