Survey on Multi-agent Driving Game Simulator

Saurabh Pahwa[#1], Dhruv Patel[#2], Harish Nair[#3], Yash Sheth[#4],Prof. Payel Thakur[#5]

Computer Department , Pillai College of Engineering
Maharashtra, India

***Abstract*— The Multi-Agent Driving Game Simulator will consist of a real-time driving environment with accurate physics simulation. A population of AI drivers is trained using reinforcement learning. At each step the progress and live simulation is demonstrated. The environment can be made more complex as the AI gets better at traversing it. After sufficient training the efficiency and proficiency of the AI drivers can be demonstrated whereby they compete for driveable space and co-operate with other agents to minimize crashes. Later human decision makers can be introduced to the environment to further test the proficiency of AI in the presence of unpredictable scenarios. This type of dynamic and adaptive environment is a powerful test-bed for demonstration of reinforcement learning algorithms and their proficiency in solving hard problems by exploration in tandem with exploitation. This project is aimed at aiding the Autonomous Vehicle industry and provides a safe and quick assessment of the self-learning algorithms.**

I. Introduction

There are numerous techniques currently deployed in the self-driving space to reach autonomy. The two most prevalent techniques are Localization by LIDAR and end-to-end deep learning by computer vision. The former lacks in the task of accounting for the unpredictability that occurs in the driving task whereas the latter is limited by the amount and the quality of data available for training an end-to-end deep learning model. In this project, reinforcement learning techniques are used in a dynamic environment which makes the learning rate much more representative as the agent has more degree of freedoms in contrast to the aforementioned techniques

II. Literature Survey

Driver Modeling through Deep Reinforcement Learning and Behavioral Game Theory. Berat Mert Albaba,Yildiray Yildiz. Advantages: Deep Q-Learning Algorithm gives excellent predictive power. Disadvantages: Complexity increases with the size of the environment.

Emergent Tool Use From Multi-Agent Autocurricula Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glen Powell, Bob McGrew, Igor Mordatch. Advantages: Demonstration of multi-agent Co-operation. Disadvantages: environment can be exploited by the model.

Deep Reinforcement Learning using genetic algorithms for Parameter Optimization Adarsh Sehgal, Hung Manh La, Sushil J. Louis, Hai Nguyen. Advantages: Combination of DDPG and Genetic algorithm is a powerful repertoire for a complete self-driving algorithm. Disadvantages: need for an optimizing tool for Reinforcement Learning Performance.

Exploration by Random Network Distillation.Yuri Burda ,Harrison Edwards ,Amos Storkey ,Oleg Klimov. Advantages: Parameter optimization and no bias. Disadvantages: Cannot be applied to high level exploration problems.

PPO DASH Improving generalization in deep reinforcement learning: Improving Generalization in Deep Reinforcement Learning Joe Booth. Advantages: Can be efficiently applied to a gamified environment with a large population of agents. Disadvantages: Large Number of iterations on a large population increases complexity.

III. Algorithms and Techniques

A. Existing System:

Independent driving is not a single technology but rather a sophisticated system that incorporates many technologies.A standalone automation technology consists of three main systems: algorithms, including hearing, sight, and resolution; client, including application and hardware platform; and a cloud platform, which includes data storage, simulation, high-resolution (HD) mapping, and in-depth learning model training. Meaningful information is extracted from raw sensor data and decisions about actions performed by an algorithm. In addition, to meet real-time requirements and reliability, the customer system integrates with these algorithms. The client-supported system ensures that the longest phase of the processing pipe takes less than 16 milliseconds (ms) to complete when the sensor camera produces at 60Hz. Offline computer features are provided by the cloud platform for independent vehicles. We are able to explore new algorithms and update HD map-plus, train better recognition, tracking and decision models using cloud platforms.

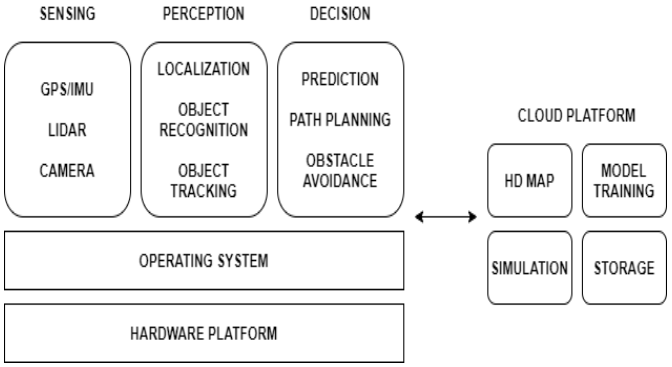

Fig. 3.2 Existing system architecture used for autonomous driving technologies [6]

The development of a high-quality content-based recommendation system is shown in Figure 3.2. The recommendation process is done in three steps, each with a separate component:

Sensing:

Various sensors are used in autonomous driving such as GNSS receivers, LiDAR and Cameras. GNSS receivers help private vehicles to do their own thing by updating world positions with an accuracy of at least meters. LiDAR is commonly used to create High-Definition, real-time maps, and to avoid obstacles Cameras are widely used for object recognition and tracking functions, such as route detection, traffic detection, and pedestrian detection.

Perception:

It is about understanding the environment, including the current location, object recognition, and tracking etc. Several can be utilized to acquire accurate vehicle position updates in real-time. The natural choice for localization is to use GNSS directly.

Decision Making:

In the decision-making phase, action forecasting and planning methods are combined to produce a workable application in real time. A major challenge in organizing driving autonomy is to ensure that private vehicles travel safely in complex road environments. The decision-making unit makes the predictions of nearby vehicles before making an app decision based on these predictions. Planning an autonomous vehicle in a dynamic area is a complex problem, especially when the vehicle needs to use its full mobility.

B. Proposed System:

Proposed System Architecture:

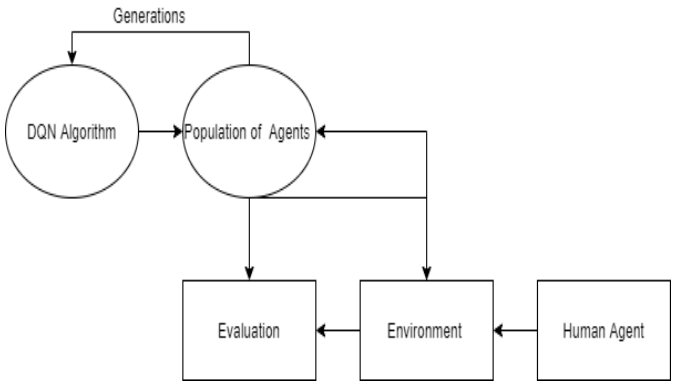

Fig. 3.3 Proposed system architecture

A. DDPG Algorithm:

Deep Deterministic Policy Gradient or commonly known as DDPG is basically an off-policy method that learns a Q-function and a policy to iterate over actions. It employs the use of off-policy data and the Bellman equation to learn the Q function which is in turn used to derive and learn the policy.

The learning process is very closely related to Q-learning where if you know the optimal-action-value function Q*(s, a), the best and optimal action to taken in that state can be found out using a*(s) which is [3].

$$a^*(s) = \arg\max_a Q^*(s,a)$$

Q-learning based algorithms, specifically DDPG employs the use of the following to deal with a continuous action space: Make use of the Bellman equation to obtain the optimal action for a given state using its state-action/Q-value[3]

$$Q^*(s,a) = E_{s'\sim P}[r(s,a)+\gamma \max_{a'} Q^*(s',a')]$$

DDPG employs the use of mean-squared Bellman error (MSBE) function which estimates how close Q* comes close to satisfying the Bellman equation as shown in the equation[3]:

$$L(\phi,D)=E_{(s,a,r,s',d)\sim D}[(Q\phi(s,a)-(r+\gamma(1-d)\max_{a'}Q\phi(s',a')))^2]$$

B. Deep Q Learning Algorithm:

1: Initially, D(memory) is set to N capacity.

2: Main network must be initialized along with the target network. QN and QN^T. with weights sampled from a uniform distribution of range $[-\sqrt{\frac{6}{n_{input}+n_{output}}}, \sqrt{\frac{6}{n_{input}+n_{output}}}]$ where n_{input} and n_{output} are the number of input and output neurons, respectively.

3: T = 50 is set

4: For episodes 1 to M perform:

5: For t from 1 to K perform:

6: Action at , taking the probability values Pt(ai) = eQNt (ai)/Tj=0n-1eQNt (aj)/T, i= 1,2,3.....size(ActionSpace)
7: ExecuteAction(a)t , ObserveReward(rt), transitioned state st+1
8: cache the experience (st , rt , at , st+1) in D
9: if size(D) ≥ n, then
10: Sample an odd batch of experiences. consisting of P four-tuples (sj , rj, aj, sj+1)
11: for j = 1 to P do
12: Set yj =rj + γmax, QNT (sj+1 , a; W)
13: if sj+1 , is terminal, i.e. ego driver crashes. then
14: Set yj = rj
15: end if
16: Carry out a gradient descent step using the cost function yj-NN(sj+1, aj;W))2 with respect to weight matrix W
17: end for
18: end if
19: if sj+1 , is terminal, i.e. ego driver crashes, then
20: break
21: end if
22: end for
23: if T>1 then
24: Update Boltzmann Temperature T = T*c, c < 1
25: end if
26: end for

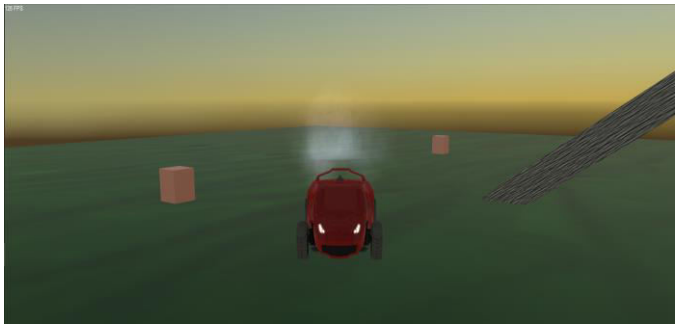

Fig. 3.4 Proposed system simulated environment

Implementation Details:

The Reinforcement learning algorithms with optimizations can be inferred from the open-ai baselines paper. all the dependencies to train the agent are installed via the open source github repo openai/baselines.

The environment is made using unity and unreal engines to provide an in depth and high quality textures. The assets can be loaded in to the environment and their inputs and outputs can be controlled via C#.

The performance of the agents on the environment is consistently evaluated and the data generated is stored in a database to further analyze the behavior and learning patterns in order to optimize the agents.

Evaluation Metrics:

The Performance of the agent can be measured in two ways: MBSE Loss metric :

DQN algorithms and other reinforcement learning algorithms can be assessed based on MSBE Loss which is calculated based on the aggregate of all the loss values.

Reward Function Changes : Since the behavior of the agent depends on the updation of the reward function, which indicates either a positive or a negative reward based on the action taken by the agent.

The agent's action selection is modeled as a map called policy[7]:

$$\pi: A \; x \; S \; -> \; [0,1]$$

$$\pi(a,s) \; = \; Pr(at \; = \; a \; | \; st \; = \; s)$$

The policy map gives the probability of taking action a when in state s.

The frequency of these actions determine the performance of the ai. Since there can be multiple agents in the environment, the collective performance of these populations of agents can be modelled in a reward function vector and the minimum and maximum values can be inferred.

IV. APPLICATIONS

Automated cars:

A fully automated vehicle's commercial availability appears to be adjusted around the corner. The final obstacles to commercial realization may be the necessary changes to road safety regulations and perhaps more difficult to define, the overcoming of individual and community fears of relying on the wheels of an automated machine on our roads. Tests on current automated vehicles, however are proving their reliability, with test results better than human judgments made on the road.

Vehicle automation in different fields:

We can use it in farming vehicles like tractors, irrigators and many more, it can also be used in mining vehicles like drilling rigs and it can also be implemented on industrial vehicles like forklift, robots and car crash testing vehicles.

GPS acquisition and processing:

GPS is used to obtain the absolute position of an object in an open space environment. In our simulation, our proposed design is able to navigate a route based on GPS rather than pre saved maps that are not frequently updated and do not include

all roads of all countries. In addition to gps there are gyroscope and accelerometers which help to contain the xyz axis coordinates of the ai based agent.

Landmark assistance in local positioning systems:

The car detects forms that are meant to be there as natural, such as trees, barriers, posts, et cetera. Usually, it may be difficult to locate natural landmarks, reducing system effectiveness. A tree, for instance, may change form, or obstacles emerge from various angles.

Artificial landmarks, in contrast, provide the autonomous vehicle with a simple, reliable way to define its location.

Control of the automated vehicle:

The information acquired by the automated vehicle is used to drive. The vehicle controls the speed and direction of the vehicle. The first and foremost task of the vehicle is to control the motion along a specific line. So, it uses Proportional Integral Derivative to get the orientation difference and the distance. Mainly Proportional Integral Derivative checks where the automated vehicle should be and where it is. Reason behind this is to reduce the difference between it. For this automated vehicle applies a theoretical model to get the final result. This can help to minimize the human input in driving vehicles thus reducing the errors while controlling the agent.

Automated vehicle path planning:

When an in line motion is achieved during autonomous vehicle control, the results can be extrapolated from one line to another. The vehicle agent can create paths of any desired shape which is achieved after determining its own path using decision rules. This can be helpful to normal, disabled and elderly people. It can be used on roads, facilities like factories, airports and campuses to transport passengers or loads while aiming to reduce overhead cost and energy for traditional path planning

.

Obstacle avoidance:

Obstacle avoidance For successful independent driving, automated vehicles are required to navigate a number of path modifications. If an obstacle is on the way to the initial route, an autonomous vehicle also needs to decide one of the path modifications by itself. An obstacle, if it is in motion, could be stationary or in a collision path. The autonomous vehicle has to take into account the vehicle model and the environmental map to decide the shortest route to regain the original path, taking into account the minimum clearance of the nearby barriers, in order to eliminate the barrier. Decision algorithms are used to segment the map into a grid to do so.

V. SUMMARY

Reinforcement learning in real world autonomous driving applications is still an evolving field. Although a few commercial applications are successful, very little literature or large-scale public datasets are available. Therefore, we were inspired to formalize and coordinate RL autonomous driving applications. Interacting agents are involved in autonomous driving scenarios which include negotiation and complex decision making that fits RL. However in order to provide mature solutions that we address in depth, there are several problems to be solved.

VI. ACKNOWLEDGMENT

We are also grateful to Dr. Sandeep Murlidhar Joshi, principal of Pillai College of Engineering for giving us the opportunity to work with them and providing us the necessary resources for the project.

We would like to express our deep gratitude to Dr. Sharvari S. Govilkar, our Head of Department for Computer Engineering, for their patient guidance, enthusiastic encouragement and useful critiques of this research work.

We would like to convey our sincere gratitude to Prof Payel Thakur for her useful and positive feedback during the preparation and production of this Project. Her willingness to give her time so generously was very much appreciated.

REFERENCES

[1] B. Albaba and Y. Yildiz, "*Driver Modeling through Deep Reinforcement Learning and Behavioral Game Theory*", Arxiv.org, 2020. Available: https://arxiv.org/pdf/2003.11071.pdf. [Submitted on 24 Mar 2020]

[2] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever and I. Mordatch, "*Emergent Complexity via Multi-Agent Competition*", arXiv.org, 2020. Available: https://arxiv.org/abs/1710.03748. [Submitted on 10 Oct 2017 (v1), last revised 14 Mar 2018 (this version, v3)]

[3] T. Salimans, J. Ho, X. Chen, S. Sidor and I. Sutskever,"*Evolution Strategies as a Scalable Alternative to Reinforcement Learning*" Arxiv.org, 2020. Available: https://arxiv.org/pdf/1703.03864. [Submitted on 10 Mar 2017 (v1), last revised 7 Sep 2017 (this version, v2)]

[4] Y. Burda, H. Edwards, A. Storkey and O. Klimov, "*Exploration by Random Network Distillation*", arXiv.org, 2020. Available: https://arxiv.org/abs/1810.12894. [Submitted on 30 Oct 2018]

[5] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell and A. A. Efros, Pathak22.github.io, 2020. Available:

https://pathak22.github.io/large-scale-curiosity/resources/largeScaleCuriosity 2018. [Submitted on 13 Aug 2018]

[6] https://www.oreilly.com/radar/creating-autonomous-vehicle-systems

[7] Kaelbling, Leslie P.; Littman, Michael L. Moore, Andrew W. (1996). "*Reinforcement Learning: A Survey*". Journal of Artificial Intelligence Research. 4: 237–285. arXiv:cs/9605103. doi:10.1613/jair.301. S2CID 1708582. Archived from the original on 2001-11-20.