Proceedings of the 7th North American International Conference on Industrial Engineering and Operations Management, Orlando, Florida, USA, June 12-14, 2022

A Photo-Realistic Simulation and Test Platform for Autonomous Vehicles Research

Alexandra Dubs[1], Victoria Correa Andrade[1], Mark Ellis[2], Seth Ganley[2], Bayazit Karaman[2], and Onur Toker[1]

[1]Department of ECE, [2]Department of Computer Science
Florida Polytechnic University
Lakeland, Florida, USA

{adubs7221, vcorreaandrade7411, mellis6464, sganley6511, bkaraman, otoker}@floridapoly.edu

Abstract

In this paper, we present an autonomous vehicles (AV) research testbed with photo-realistic visualisation and simulation capability. Compared to Oak Ridge National Labs CAVE system, Microsoft's AirSim, and MATLAB's Vehicle Dynamics Blockset approach, the key difference is a decoupled and modular system architecture which allows developers to focus on only one aspect of the whole system and be productive with minimal training. Basically, the proposed system has a game engine, an external hardware accelerator for vehicle dynamical models, and a GPU equipped computer to run the AI algorithms. Our vision is to have all vehicle dynamical models to be built in MATLAB/Simulink and run on a hardware accelerator like Speedgoat, which is a leading software and hardware framework for scientific system modelling and hardware in the loop simulation. The proposed photo realistic visualization environment is Unreal Engine based, and all AI models are keras based. This decoupled system architecture allows all vehicle dynamical models and AI decision making to be implemented outside the Unreal Engine, while the game engine doing all collision detections and reporting. Similarly, all traffic and pedestrian movement stochastic models can be implemented outside the game engine, allowing a framework agnostic approach to be used throughout the whole system.

Keywords

Autonomous Vehicles, AI algorithms, Vehicle Dynamics, Photorealistic Simulation

1. Introduction

Autonomous vehicles (AVs) have potential to improve transportation efficiency and reduce traffic accidents. There have been very promising developments in AV research. For example, in (Bojarski et al. 2016), NVIDIA demonstrated a convolutional neural network (CNN) based on a single front-looking camera system for end-to-end steering. There are several AV research testbeds, for example (Gong et al. 2019), and (Baker et al. 2018). Authors would like to cite (El-Tawab et al. 2020), (Somogyi et al. 2018), and (Hafez et al. 2020) as alternative testbeds as well.

However, there is one fundamental difficulty in most of these reported research directions. Physically testing AVs for different scenarios is time consuming and may not be 100% safe for the vehicle team, nearby drivers, and pedestrians. Because of this difficulty in testing and verification of AVs, we propose a photo-realistic simulator which can be used to test any vehicle in almost any environment. The proposed system allows changes in light conditions, weather, and elevations. Furthermore, the vehicle dynamical model, i.e. the differential equations describing the motion of the vehicle, can easily be changed. Finally, different traffic and pedestrian models can be generated without changing anything in the game engine. All of these are also decoupled form the AI algorithm used for navigating the vehicle. The main goals of our research are (1) Simulate realistic vehicle dynamics defined in MATLAB/Simulink,

and (2) Simulate stochastic pedestrian and vehicle characteristics expressed as MATLAB/Python code or MATLAB/Simulink block diagrams.

D. Pomerleau's work, (Pomerleau 1989), is an important milestone for self-driving vehicles. In this paper, Pomerleau introduced the idea of using a neural network which outputs directly the steering angle from images captured by a front facing camera. Although the full model developed in (Pomerleau 1989) is a recurrent neural network with a camera and a range finder, it is possible to build a smaller feedforward network with

- 30x32 input layer representing the input retina/image sensor,
- a hidden layer, and an output layer of 30 nodes representing different steering angle positions,
- and almost 4000 trainable weights/parameters

Pomerleau used a real vehicle for AV research, however there is an alternative AV research direction based on game engines to create photo-realistic environments. Udacity's driving simulator, is a relatively basic but reasonably realistic driving simulator consisting of two different tracks. In Fig. 1, we present our test results for a pre-trained AI model. Udacity's driving simulator has a training mode and an autonomous mode. If there is no pre-trained model available, then one has to drive the vehicle manually, generate a dataset, and then use it for training a model. This approach is also called behavioral cloning.

Fig. 1. Udacity driving simulator: On the left, Unreal Engine based gaming software is used for photo-realistic simulation; On the top right, Python code running a pre-trained AI model generates steering angles; Finally on the bottom right, vehicle's front facing camera view is shown.

This test setup requires two separate executables running at the same time. The first one is the Unreal Engine based gaming software and the second one is a Python script running a `tensorflow.keras` model. These two executables "communicate" with each other by local interprocess communication techniques. The game engine generates a photo-realistic simulation environment and provides the front facing camera view to the Python script running the AI model. The Python script generates the steering angle using the AI model and sends this information to the gaming software. The game engine also provides current speed information, but the throttle and break are computed using a simple PID algorithm on the Python side and passed to the gaming software to close the loop. Effectively, the Python script drives the vehicle along the track and has resilience to external disturbances. Namely, the user can manually take the control and generate a disturbance and then leave the control to the AI engine, and the AI code restores the vehicle's position and continues to drive normally.

Carla, is another open-source simulator for autonomous driving research. Compared to the Udacity's approach, Carla platform allows more versatile AI algorithms to be implemented for self-driving vehicles research. More specifically, Carla can report collisions and therefore it can be used to implement reinforcement learning techniques for self-driving. To the best of authors' knowledge, there are a couple of other photo-realistic simulators used for self-driving vehicles research. These include AirSim, Euro Truck Simulator, and Deep Drive.

Authors would like to cite (Velasco-Hernandez et al. 2020) and (Saleem et al. 2021) for reviews of autonomous vehicle architectures and steering angle prediction techniques. In (Alghodhaifi and Lakshmanan 2021), a comprehensive review of modelling and simulation alternatives are presented. Other relevant work include (Ijaz and Wang 2021), (Islam et al. 2021), (Munir et al. 2020), (Zhang and Huang 2020) and references therein.

We have a research group at Florida Polytechnic University working on autonomous vehicles related problems. This work is basically a part of our long-term vision of becoming a AV research center. This paper is organized as follows: In Section II, we present a detailed summary of the proposed system, in Section III, which is the core of this work, Unreal Engine based campus model, a digital city model, and two-way communication using realtime UDP are presented. Section IV is mainly about vehicle dynamical models in MATLAB/Simunlink, running on an external hardware accelerator, and linking this to the game engine over UDP messaging. Finally, in Section V, we make some concluding remarks and outline future research directions.

2. Proposed System Architecture

Oak Ridge National Labs Connected and Automated Vehicle Environment Laboratory, a.k.a. CAVE, is a high-end D-Space based AV testbed with virtual traffic simulation capability (Smith 2022). Basically, it is a photo-realistic simulation environment to test AV algorithms under various traffic and environmental conditions. Microsoft's AirSim is a related software framework with certain similarities to CAVE as well as to the system designed in this work.

Our proposed solution is a low-cost alternative based on the Unreal Engine, an external hardware accelerator for vehicle dynamics, and an GPU equipped subsystem for AI based navigation algorithms. Further details of the proposed system are illustrated in Fig. 1.

Fig. 2. Unreal Engine Based System Architecture.

Basically, there are three different parts or subsystems: The first one is the Unreal Engine based photo-realistic environment. This subsystem will be used to model real or real-like physical environments, and to generate photo-realistic views. The vehicle dynamical models will run on an external hardware accelerator (Speedgoat), which will be the second subsystem. There will be a 1-GigE connection between these two, the Unreal Engine will be responsible for collision detection like tasks and reporting these to the external dynamical model, whereas the external dynamical model will be responsible for continuously updating the Unreal Engine about the vehicle's position and orientation. This decoupled system architecture allows highly complex MATLAB/Simulink based vehicle dynamical models to be simulated in the Unreal Engine without re-implementing these MATLAB/Simulink models in C++.

The third and the final subsystem is a GPU based computer running AI algorithms for lane detection, object recognition, steering decisions, and other related tasks. The Unreal Engine generates the driver's front view and

displays it on a large size high resolution display. There will be a camera on the GPU equipped computer, and this camera will basically see the large size high resolution display controlled by Unreal Engine. In summary, all AI algorithms are decoupled from the Unreal Engine, and can be implemented in Python using different AI frameworks. There will be a separate 1-GigE connection between the AI computer and the external dynamical model, which will be used to send steering and navigational decisions of the AI model to the vehicle dynamical model.

3. Photo-Realistic Visualization Subsystem

Our Photo-Realistic Visualization subsystem is developed using the Unreal Engine, which is an industry standard game engine that can be used for high performance photo-realistic simulation and visualization. Basically, we have built a model of the Florida Polytechnic Campus with an electric golf cart and simulated its movement with vehicle dynamics implemented completely outside the Unreal Engine. Addition of other vehicles, pedestrians, and their stochastic movement dynamics is planned as future work. Note that, in our proposed system dynamical models are completely outside the Unreal Engine, only collision detection type tasks are handled by the game engine and reported to the vehicle dynamical model over a high-speed link. The main advantage of this decoupled system architecture is ease of maintenance and development, plus support for multiple programming languages like MATLAB/Simulink and Python for traffic pattern generation type tasks.

3.1 Florida Polytechnic University Campus Model

The design of the Florida Polytechnic University campus is one of the interesting construction projects in the US. Testing an autonomous golf cart in the campus would take more time compared to a pure in-lab based testing. Furthermore, in-lab based tests are also much safer and more practical. Therefore, we have first modelled the campus in Unreal Engine 4.24, which will allow us to test our different autonomous vehicle dynamics and traffic patterns in a lab-environment with photo-realistic visualization accuracy.

The overview of the environment in our simulator is given in Fig. 2. While Fig. 3 shows the view of the IST building, the view of the dorms is presented in Fig. 4.Finally, Fig. 5 presents a snapshot of a demo video available on YouTube.

Fig. 2. The overview of the proposed model for Florida Polytechnic University.

Fig. 3. The view of the IST building at

Fig. 4. The view of the dorms at

Florida Polytechnic University. Florida Polytechnic University.

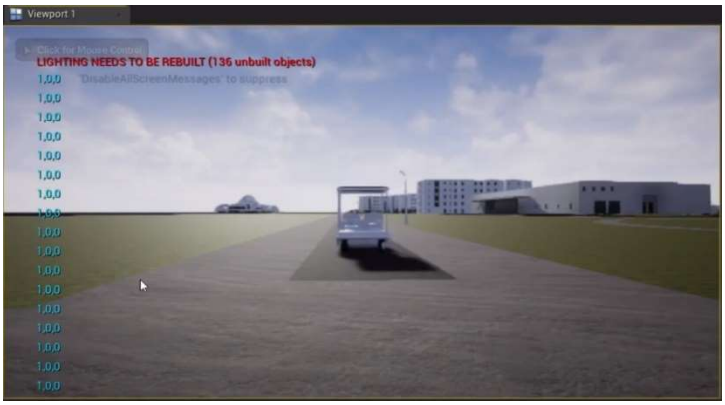

Fig. 5. A short YouTube demo video of our initial implementation, https://youtu.be/gJBJsJ8IZvk. Note that the vehicle dynamics is implemented completely outside the game engine.

3.2 Modular City: An Urban Environment Model

For the second phase of our research, we have used a digital asset available from the Epic store, see Fig. 6. This is an urban environment model with roads, intersections, buildings, traffic lights, and several other realistic details.

Fig. 6. Modular City is an urban environment model available from the Epic store.

3.3 Two-way UDP communication

The proposed system is based on the idea of a decoupled system with the game engine and the vehicle dynamical model running on different physical devices, and each system is developed using different software frameworks. We are using a 1 GigE connection for two-way real-time UDP communication between these two systems. ObjectDeliverer library is used to create a communication protocol between an autonomous vehicle model and our photo-realistic simulator. The ObjectDeliverer is freely available Unreal Engine plugin. The Blueprint shown in Fig. 7 is used to add UDP functionality to the Unreal Engine code. A string based simple packet format is adopted with vehicle x, y, z velocities separated by commas. A more efficient binary packet format is being planned for future versions.

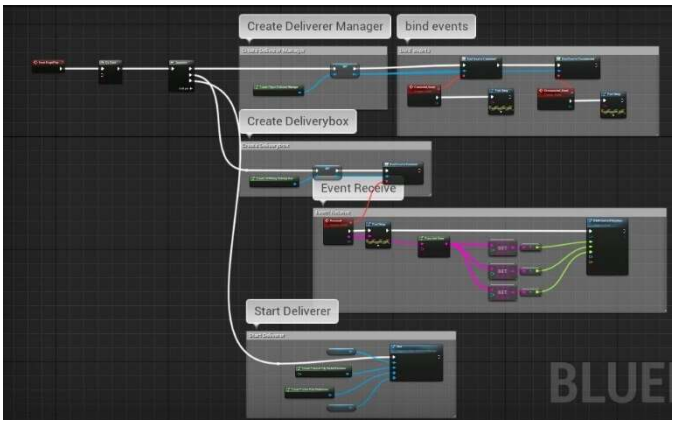

Fig. 7. ObjectDeliverer library based Blueprint used for UDP communication.

On the MATLAB/Simulink side, we are using Simulink Real- time IP blocks for UDP receive and send, see Fig. 8.

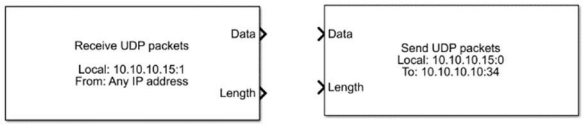

Fig. 8. Simulink Real-time IP blocks for UDP receive and send.

The vehicle dynamical model will send position, and orientation updates over the UDP link, whereas the game engine will send collision detection type reports to the MATLAB/Simulink model over the same high-speed link. As a proof of concept type test, we have used the following Python test code to send velocity data to the game engine continuously.

```
import socket

UDP_IP = "localhost" UDP_PORT = 8000
MESSAGE = "1,0,0" # velocity vector is np.array([1,0,0])
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) print("Starting")
while(True):
        #send the data through the socket to server
        sock.sendto(bytes(MESSAGE, "utf-8"), (UDP_IP, UDP_PORT))
        print("Sent")
```

This test has been done on an x64 PC with Intel i7 5th generation processor, NVIDIA GTX970, and 16 GB RAM. Currently, all messages are sent in text format, however MATLAB UDP blocks use IEEE-754 single or double precision formats. By using the `struct` package which supports float to bytes conversion for both little-endian and high-endian formats, it is possible to communicate with a MATLAB/Simulink block diagram.

The proposed system is designed so that while one software engineer is working on the game engine, the other developer can focus vehicle dynamical models in MATLAB/Simulink. These frameworks are quite different and require completely different skills and technical background to be productive. Adding traffic patterns or pedestrian movement models to our system is also relatively easy because of this decoupled system architecture. Basically, we can have multiple UDP ports in the game engine with some of them used for communication with vehicle dynamical models in MATLAB/Simulink, whereas the rest of the UDP port(s) are used to get pedestrian movement decisions from various AI models implemented in Python. In summary, the proposed system enables the simulation of different vehicle dynamical models, and traffic/pedestrian patterns in a more manageable way.

4. Vehicle Models

In this section, we present vehicle dynamical model built in MATLAB/Simulink. The output of the vehicle dynamical model can be instantaneous velocity connected to a UDP send block which will be used to report these to the game engine. The dynamical model also has driver input, which is read from another UDP block, this time UDP receive. This data comes from an AI model used for autonomous navigation.

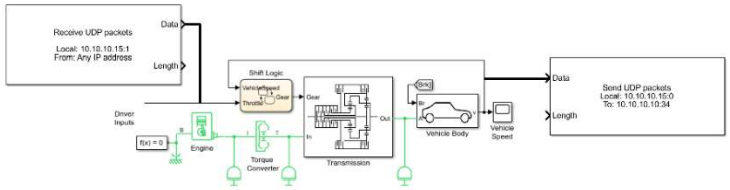

Fig. 9. Vehicle dynamical model with UDP blocks for communication with the game engine and the AI computer. UDP receive is from the AI computer running the navigation code, whereas UDP send is to the game engine for updates of the photo-realistic environment.

In our first design, the vehicle model can be executed on a Speedgoat external hardware accelerator. Our second vehicle model is shown in Fig. 10.

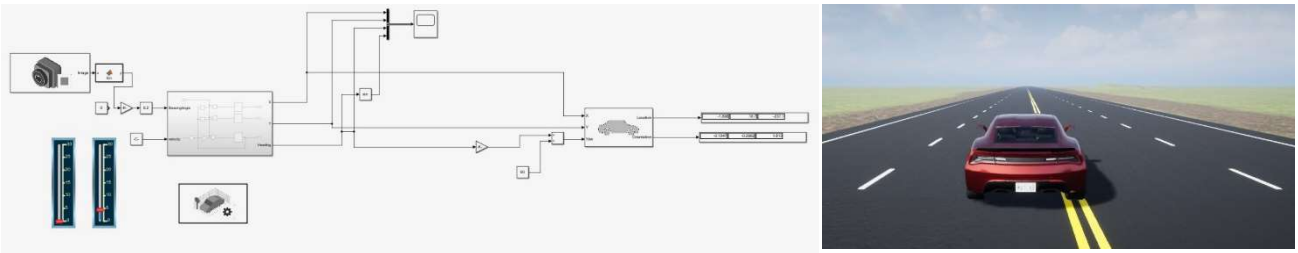

Fig. 10. Vehicle model (on the left), and photo-realistic environment (on the right).

For this second model, we are using MATLAB's Unreal integration, a.k.a. Unreal Engine Scenario Simulation. MATLAB/Simulink blocks are used to build a basic vehicle model with steering taken into account. We have tested this system without any AI integration, i.e. simply using the keyboard and mouse in manual driving model. Later, we have used an AI model to generate steering commands and tested it on this simulation platform.

5. Vehicle Modelling Based on Real Data

In this section, we will present a vehicle model that we developed using real data. Florida Polytechnic University has an Autonomous Vehicles laboratory, see Fig 11, which is set up by using both NSF funds and funds provided by the state of Florida. We have an electric golf-cart with drive-by-wire conversion done by our undergraduate and graduate students. We also have a Ford Fusion Plug-In Hybrid with drive by conversion done by Dataspeed, Rochester Hills MI.

On May 4th, 2022, the Ford Fusion is driven along the track shown in Fig 12. This track includes both city and highway traffic. Collected vehicle data consists of measured throttle (As % value), measured break (As % value), and measured speed (As km/h). Speed data is smoothed, and then differentiated numerically to obtain acceleration data.

Fig 13, and Fig 14, have details of the measured raw data. Between time values 890 sec and 960 sec, we see that steering angle is zero, and there are regions where break values are non-zero, and there are alternative regions where the throttle values are non-zero. Furthermore, during this time interval, vehicle speed is always positive, i.e. the vehicle never stops. In summary, during this interval we have simple linear vehicle dynamics.

Fig 11. Autonomous vehicles lab at Florida Polytechnic University. On the right, Ford Fusion Plug-in Hybrid and in the center electric golf-cart. Both vehicles are drive-by-wire capable and actively used for AV research.

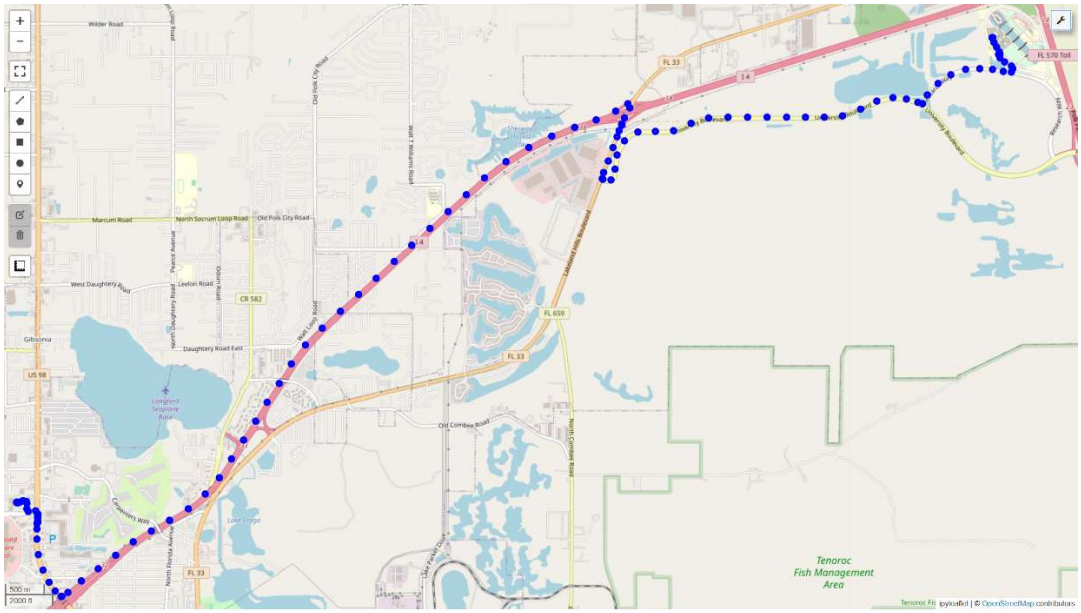

Fig 12. On May 4th, 2022, we drove the vehicle from Florida Polytechnic University campus (Top right) to Lakeland Square Mall (Bottom left). The selected track includes both city and highway traffic. Collected vehicle data is used for model development.

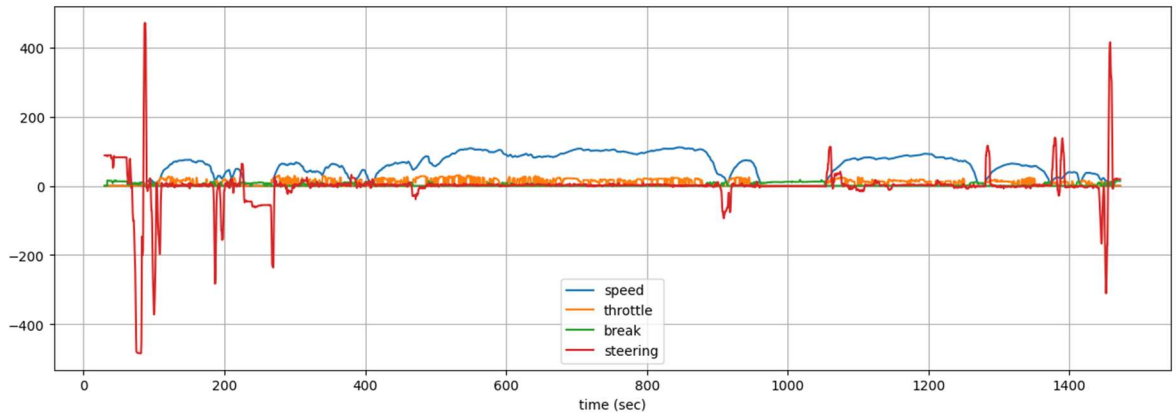

Fig 13. Measured raw data. Speed is in km/h, throttle and break are shown as percentage, and steering is in degrees.

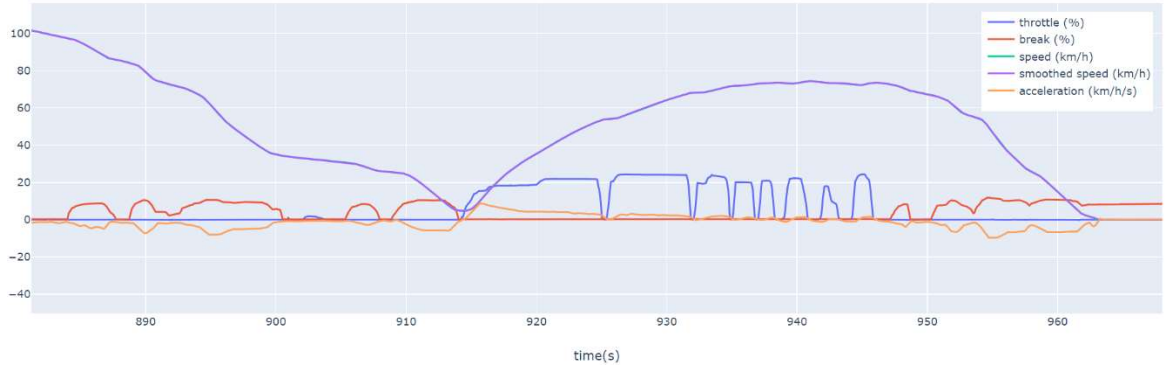

Fig 14. Measured raw data, and computed acceleration values. Speed is in km/h, throttle and break are shown as percentage, and acceleration is in km/h/s.

We developed a regression model by using the Python library `sklearn`. With a regression score of 80.12%, we get

$$A = 0.129\,\mathrm{TH} - 0.549\mathrm{B} - 0.190$$

where A represents the vehicle acceleration, TH is the throttle value as percentage and B is the break value as percentage. We are able to use this in-house developed model, as well as other vehicle models developed by different AV research groups in our HIL simulator. The constant term, -0.190, in our vehicle dynamics model represents the effect air friction. When there is no applied throttle or break, we still see a small but negative acceleration due to air fiction, see Fig 15. As a future research direction, we are considering velocity dependent air friction equations, which are expected to produce better modelling accuracy.

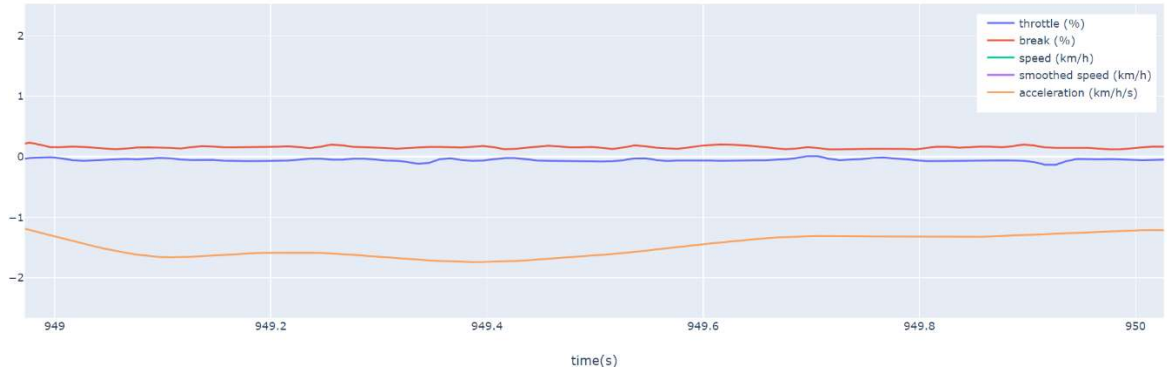

Fig 15. Negative acceleration when both throttle and break are zero. This is due to air-friction.

6. Conclusion

In this paper, we have summarized the design of a low-cost testbed for AV research. Similar to the CAVE system of Oak Ridge National labs (Smith 2022), we have a photo-realistic environment built using a game engine. However, we have a decoupled system architecture with vehicle dynamical models built in MATLAB/Simunlink, traffic/pedestrian patterns realized in Python, and autonomous navigation algorithms running on a physically separate GPU equipped computer. The main advantage of this modular approach is, three or more different developers having completely different skills and technical background can work in a quite productive way with minimal training. Overall, the proposed system is expected to ease and accelerate AV research.

Acknowledgements

This work has been supported in part by NSF grant 1919855, Advanced Mobility Institute grants GR-2000028, GR-2000029, and the Florida Polytechnic University grant GR-1900022. Authors would like to thank A. Sargolzaei, M. R. Khalghani, S. de Oliveira, M. Kalman, and B. Hicks for their collaboration and support.

References

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. *End to End Learning for Self-Driving Cars*, arXiv:1604.07316v1, 2016, https://arxiv.org/pdf/1604.07316v1.pdf

Gong, Z., Xue, W., Liu, Z., Zhao, Y., Miao, R., Ying, R., and Liu, P. *Design of a Reconfigurable Multi-Sensor Testbed for Autonomous Vehicles and Ground Robots*, Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, May 2019, https://doi.org/10.1109/ISCAS.2019.8702610

Barker, R., Hurst, A., Shrubsall, R., Hassan, G. M., and French, T. *A Low-Cost Hardware-in-the-Loop Agent-Based Simulation Testbed for Autonomous Vehicles*, Proceedings of the 2018 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), Auckland, New Zealand, Jul. 2018, https://doi.org/10.1109/AIM.2018.8452376.

Pisu, P., *Autonomous Golf Cart Testbed*, https://cecas.clemson.edu/pisugroup/autonomous-golf-cart.html.

El-Tawab, S., Sprague, N., and Mufti, A. *Autonomous vehicles: Building a test-bed prototype at a controlled environment*, 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), 2020.

H. Somogyi, Pup, D., Koros, P., Mihaly, A., and Soumelidis, A. *Research of required vehicle system parameters and sensor systems for autonomous vehicle control*, IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI), 2018.

Hafez, H., Maged, S. A., Osama, A., and Abdelaziz, M. *Platform modifications towards an autonomous multi-passenger golf cart*, 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES), 2020.

Smith D. E., *Connected and Automated Vehicle Environment Laboratory - CAVE*, 2022, https://www.ornl.gov/content/connected-and-automated-vehicle-environment-laboratory-cave.

Pomerleau, D. *ALVINN: An Autonomous Land Vehicle In a Neural Network*, Proceedings of (NeurIPS) Neural Information Processing Systems, 1989, pp. 305 – 313.

Velasco-Hernandez, G., Yeong, D. J., Barry J., and Walsh, J. *Autonomous Driving Architectures, Perception and Data Fusion: A Review*, 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing (ICCP), 2020, pp. 315-321, doi: 10.1109/ICCP51029.2020.9266268.

Saleem, H., Riaz, F., Mostarda, L., Niazi, M. A., Rafiq A., and Saeed, S. *Steering Angle Prediction Techniques for Autonomous Ground Vehicles: A Review*, IEEE Access, vol. 9, pp. 78567-78585, 2021, doi: 10.1109/ACCESS.2021.3083890.

Alghodhaifi, H. and Lakshmanan, S. *Autonomous Vehicle Evaluation: A Comprehensive Survey on Modeling and Simulation Approaches*, in IEEE Access, vol. 9, pp. 151531-151566, 2021, doi: 10.1109/ACCESS.2021.3125620.

Ijaz, N. and Wang, Y. *Automatic Steering Angle and Direction Prediction for Autonomous Driving Using Deep Learning*, 2021 International Symposium on Computer Science and Intelligent Controls (ISCSIC), 2021, pp. 280-283, doi: 10.1109/ISCSIC54682.2021.00058.

Islam, M. K., Yeasmin, M. N., Kaushal, C., Amin, M. A., Islam M. R., and Hossain Showrov, M. I. *Comparative Analysis of Steering Angle Prediction for Automated Object using Deep Neural Network*, 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2021, pp. 1-7, doi: 10.1109/ICRITO51393.2021.9596499.

Munir, F., Azam S., and Jeon, M. *Visuomotor Steering angle Prediction in Dynamic Perception Environment for Autonomous Vehicle*, 2020 IEEE International Conference on Consumer Electronics - Asia (ICCE-Asia), 2020, pp. 1-6, doi: 10.1109/ICCE-Asia49877.2020.9276907.

Zhang J. and Huang, H. *Steering Angle Prediction for Autonomous Cars Based on Deep Neural Network Method*, 2020 Australian and New Zealand Control Conference (ANZCC), 2020, pp. 205-208, doi: 10.1109/ANZCC50923.2020.9318380.