Title:
A Virtual Driving Test Environment for Autonomous Vehicles

Authors:
Jacob Hasenau, jhasenau@umich.edu, University of Michigan-Dearborn
Kenneth Rogale, krogale@umich.edu, University of Michigan-Dearborn
Brandon Lackowski, blackows@umich.edu, University of Michigan-Dearborn
Bruce Maxim, bmaxim@umich.edu, University of Michigan-Dearborn
Kiumi Akingbehin, kiumi@umich.edu, University of Michigan-Dearborn
Jie Shen, shen@umich.edu, University of Michigan-Dearborn

Keywords:
Autonomous Driving, Machine Learning, Virtual Environment, Virtual Vehicle, Virtual Engineering

DOI: 10.14733/cadconfP.2019.153-157

Introduction:
Autonomous vehicles are self-driving cars with the capability of sensing their environment and making control decisions without human inputs. The sensors of such vehicles may include radar, cameras, sonar, GPS, odometry and inertial measurement units. The potential benefits of autonomous vehicles are reduction of traffic accidents [11],[14], increase in fuel efficiency [5], improvement in travel comfort [2], and enhanced mobility for seniors and disabled people [6].

Because of the early stages of the autonomous driving technology, testing such vehicles is a crucial task to secure their safety before those cars are extensively used by the public. Field test is a predominant way to validate and verify the safety of autonomous vehicles. But, the field test is limited by the difficulty in constructing or reproducing problematic traffic situations that cause real traffic accidents. Each field accident occurs only after thousands of thousand miles of driving. This renders an inefficient testing methodology. A virtual testing allows scientists and engineers to assess new control strategies and data fusion schemes of autonomous vehicles under various virtual road conditions and traffic patterns. This provides a complementary way of testing that potentially saves money and manpower in the entire development cycle of autonomous vehicles.

Virtual testing of autonomous vehicles is possibly related to the modeling of vehicles with various sensors [1],[4],[13],[15], perception [12], neighboring traffic [8-9], road [16], weather [18], path planning [3], and control modules [7],[10],[17],[19]. There are several commercial software tools available in the market for general-purpose vehicle simulations. CarMaker is a product from IPG Automotive (https://ipg-automotive.com/products-services/simulation-software/carmaker-release-70). It is designed for testing passenger cars, light-duty vehicles, trucks and two-wheelers with an entire environment defined by real test scenarios. CarSim is another popular product from Mechanical Simulation (https://www.carsim.com). It aims at the simulation of advanced driving assistance system (ADAS) technologies. VIRES Virtual Test Drive is a product from VIRES Simulationtechnologie GmbH (https://vires.com) with an aim on open simulation standards and a focus on contents (road and scenario editors) and customization.

Very few, if any, past studies were focused on a virtual testing system of autonomous vehicles with respect to machine learning and computer gaming. The main objective of the study in this paper is to design and implement a computer gaming software system for testing the machine learning of autonomous vehicles.

Object-Oriented Design of a Virtual Testing System for Autonomous Vehicles:
Six main classes are designed, as shown by a UML class diagram in Fig. 1. The primary functionalities of these six classes are

(1) **TrackManager** – It spawns a set of CarControllers and puts them the inside of a class RaceCar; it checks every 0.02 seconds that every car is in the correct position, that cars are still racing, and what type of game mode is currently occurring; it also initiates evolution once generation is over.

(2) **Checkpoint** - Each Track Piece has a checkpoint, and it measures how far the specific RaceCar has gone versus how much distance the car must go.

(3) **RaceCar** - Private class of TrackManager. This class gives CarController more information that is only needed for the TrackManager.

(4) **CarController** – It holds the brain and handles the physical movement for a car.

(5) **Brain** - The neural network for each car. This class determines weights on each input and output.

(6) **CarMovement** – By using two inputs, it handles how the car moves.

A data dictionary is stored in a separate file in order to make it easier to update as well as reference.

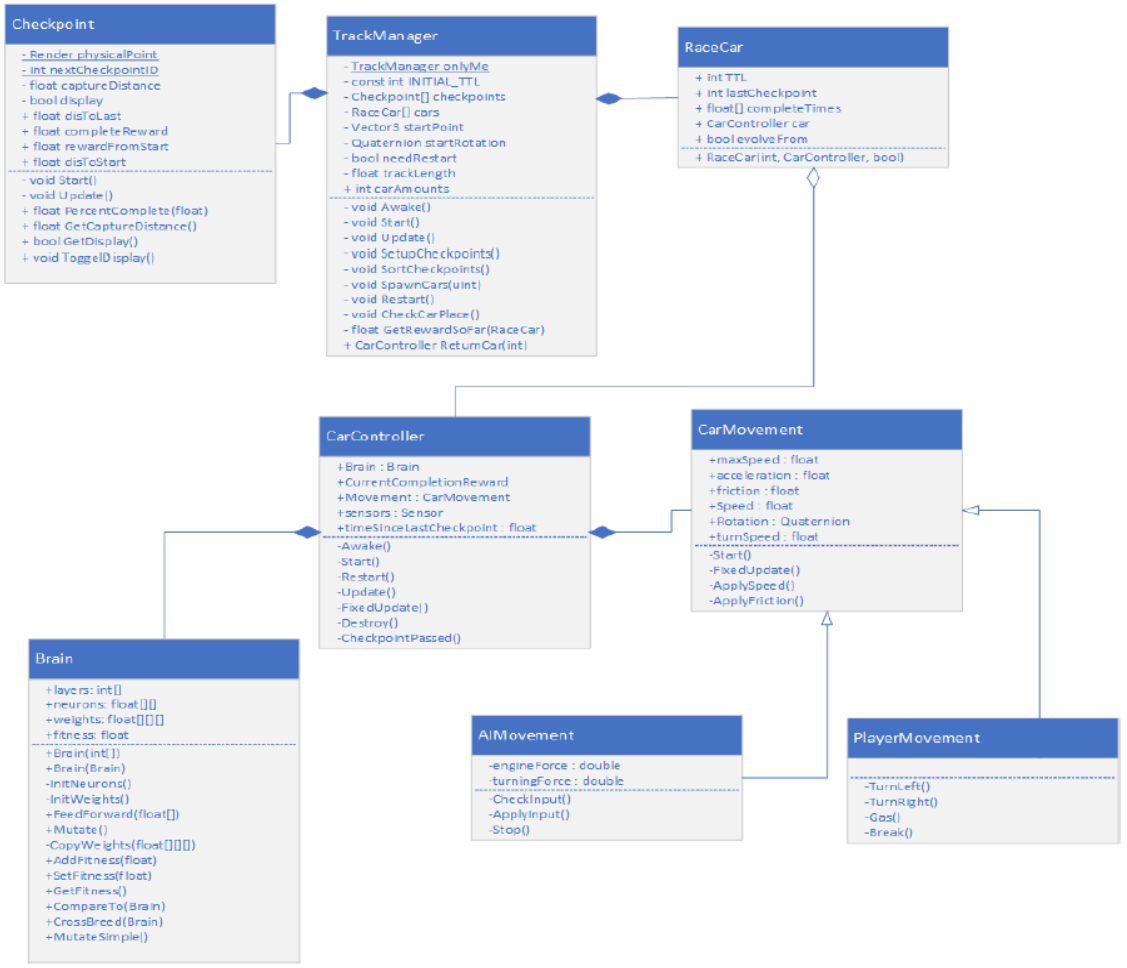

Fig. 1: A UML class diagram of our virtual testing system.

Software Interface and Behavior:

Our system uses a Unity interface and associated libraries heavily, to the point that the majority of our classes inherit from the mono-behavior class defined in Unity's libraries. The system is interacted with a user interface coded within Unity. This interface is the only way to interact directly with the system.

Fig. 2 is a focused view on one car and an overhead view of a full track view. Track Manager is begun each and every time a game mode is selected, initiates the subcomponents Track Spawner and Car Spawner, and then tracks the progress of all spawned cars across the track. Once all the cars have either crashed or completed the track, track manager initiates a restart of the track, interfacing with Brain in order to have all other cars "learn" from the best two cars and increments the generation counter. Finally, Track Manager continues until the player hits a quit condition that track manager monitors (primarily, pausing and quitting).

Track Spawner attempts to load in the map found at location MapName; if it does not exist, it returns an error and spawns a default map. It should return a set of sorted CheckPoint objects.
Track Spawner attempts to save the current map to location MapName. If it cannot be saved to MapName, Track Spawner should throw a permission error. If Track Spawner is unable to save the map due to some error with the map, an invalid map error should be thrown. If successful, SaveTrack should return true.

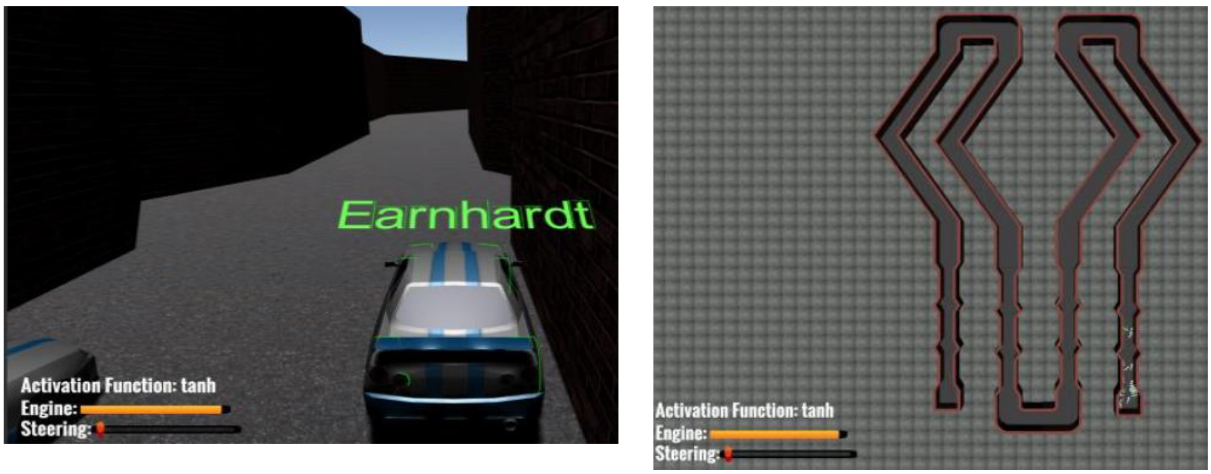

Fig. 2: An example of focused camera on one car and an overhead view.

Car Controller and Machine Learning:

Car Controller is responsible for creating a shell of a vehicle that is used by all generated, simulated vehicles as well as the user's vehicle, if the race mode has been selected. This shell includes values for the vehicle's motor force, braking force and four-wheel colliders as well as a Brain object. The two back wheels apply the motor torque and the two front wheels are capable of steering left and right. Car Controller also handles vehicles that have collided with a wall. It causes these vehicles to be disabled until the next generation. Its subcomponents are AIMovement and PlayerMovement which control the AI and player vehicle movement, respectively. The player vehicle movement is handled by user inputs while the AI vehicle movements are handled by five sensors attached to the front of the vehicles and the Brain object.

There are three main processing modes for Brain. The first is initialization, where the class either inherits weights from another brain or the weights are randomized if one is not provided. The second mode is mutation, where upon calling the function mutate the instance's weights are randomly mutated at a rate defined in the preferences. The third and final mode of operation is the feed forward

mode, where the given inputs are fired into the neural network. Next, the nodes use each connection's weight, and then the next node decides to fire or not to fire based on the sum of fired values, until the output layer is reached. From there, the function returns the values of the output nodes. Fig. 8 is an example of a neural network used in this study.

Validation Test:

There are three categories of tests in this study. The first one is vehicle-related tests:

1. Vehicle sensors should detect walls within range.
2. Vehicles should be able to speed up, slow down, turn right and turn left.
3. Vehicles should stop moving immediately after colliding with a wall.
4. Vehicles should not be able to collide with each other.

Another category is neural network-related tests:

1. The neural network should fire according to which sensors are activated.
2. The neural networks of the two top performing cars should be used as a base for the next generation.

The third category is track-related tests:

1. Track should spawn vehicles at the beginning of each simulation.
2. Track should track distance traveled of each vehicle.
3. Track should track time taken to complete a lap for each vehicle.

In total, there were 40 tests that were performed by the first three authors of this paper. All the tests were passed via a black box test or a unit test or a white-box performance test. Due to page limitation, the detailed test results are not given here.

Conclusions:

Neural Burnout is a fully customizable interactable driving test system in 3D space that applies an evolutionary neural network to a series of cars and has them run simulated races against each other until eventually, almost every car can finish the race without crashing. Users are allowed to change how many cars each generation should have and how fast the cars can travel. Forty different tests were performed with a satisfactory result. Fig. 9 shows two screenshots of the main GUIs. The system thereby provides a virtual facility for testing autonomous vehicles.

As future work, we will add the following features: (1) testing different learning algorithms, (2) automatic generation of test track based on Google map, and (3) testing different control and path planning algorithms.

References:

[1] Baer, M.; Bouzouraa, M.E.; Demiral, C.; Hofmann, U.; Gies, S.; Diepold, K.: Egomaster: a central ego motion estimation for the driver assist systems, IEEE International Conference on Control and Automation, Christchurch, New Zealand, December 9-11, 2009, 1708-1715. https://doi.org/10.1109/ICCA.2009.5410518

[2] Dang, R.; Wang, J.; Li S.E.; Li, K.: Coordinated adaptive cruise control system with lane-change assistance, IEEE Transactions on Intelligent Transportation Systems, 16(5), 2015, 2373-2383. https://doi.org/10.1109/TITS.2015.2389527

[3] Gonzalez, D.; Perez, J.; Milanes, V.; Nashashibi, F.: A review of motion planning techniques for automated vehicles, IEEE Transactions on Intelligent Transportation Systems, 17(4), 2016, 1135-1145. https://doi.org/10.1109/TITS.2015.2498841

[4] Gruyer, D.; Grapinet, M.; De Souza, P.: Modeling and validation of a new generic virtual optical sensor for ADAS prototyping, IEEE Intelligent Vehicle Symposium, Alcala de Henares, Spain, 2012, 969-974. https://doi.org/10.1109/IVS.2012.6232260

[5] Hamid, U.Z.A.; Pushkin, K.; Zamzuri, H.; Gueraiche, D.; Rahman, M.A.A.: Current collision mitigation technologies for advanced driver assistance systems --- a survey, PERINTIS eJournal, 6(2), 2016, 78-90.
[6] Haper, C.D.; Hendrickson, C.T.; Mangones, S.; Samaras, C.: Estimating potential increases in travel with autonomous vehicles for the non-driving, elderly and people with travel-restrictive medical conditions, Transportation Research Part C: Emerging Technologies, 72(November), 2016, 1-9. https://doi.org/10.1016/j.trc.2016.09.003
[7] He, Y.; Lu, C.; Shen, J.; Yuan, C.: A second-order slip model for constraint backstepping control of antilock braking system based on Burckhardt's model, International Journal of Modelling and Simulation, 2019, online first. https://doi.org/10.1080/02286203.2019.1570449
[8] Krajzewicz, D.; Hertkorn, G.: SUMO (Simulation of Urban MObility) an open-source traffic simulation, 4th Middle East Symposium on Simulation and Modelling, Sharjah, United Arab Emirates, September, 2002, 63-68.
[9] Li, P.Y.; Shrivastava, A.: Traffic flow stability induced by constant time headway policy for adaptive cruise control vehicles, Transportation Research Part C: Emerging Technologies, 10(4), 2002, 275-301. https://doi.org/10.1016/S0968-090X(02)00004-9
[10] Park, M.; Lee, S.; Han, W.: Development of steering control system for autonomous vehicle using geometry-based path tracking algorithm, ETRI Journal, 37(3), 2015, 617-625. https://doi.org/10.4218/etrij.15.0114.0123
[11] Peden, M.; Toroyan, T.; Krug, E.; Iaych, K.: The status of global road safety: The agenda for sustainable development encourages urgent action, Australasian College of Road Safety, 27(2), 2016, 37-39.
[12] Roth, E.; Drindorfer, T.J.; Knoll, A.; von Neuman-Gosel, K.; Thomas, G.; Andreas, K.; Marc-Oliver, F.: Analysis and validation of perception sensor models in an integrated vehicle and environment simulation, Proceedings Of 22nd International Technical Conference on the Enhanced Safety of Vehicles, Washington, D.C., U.S.A., June 13-16, 2011.
[13] Thrun, S.: Learning occupancy grid maps with forward sensor models, Autonomous Robots, 15(2), 2003, 111-127. https://doi.org/10.1023/A:1025584807625
[14] Toroyan, T.; Peden, M.; Iaych, K.: Global status on road report, Management of Noncommunicable Diseases, Disability, Violence and Injury Prevention, Tech. Rep., 2015.
[15] Viandier, N.; Nahimana, D.; Marais, J.: GNSS performance enhancement in urban environment based on pseudo-range error model, IEEE/ION Position, Location and Navigation Symposium (PLANS), Monterey, U.S.A., 2008, 377-382. https://doi.org/10.1109/PLANS.2008.4570093
[16] Wang, L.; Groves, P.D.; Ziebart, M.K.: GNSS shadow matching: improving urban positioning accuracy using a 3D city model with optimized visibility scoring scheme, Journal of the Institute of Navigation, 60(3), 2014, 195-207. https://doi.org/10.1002/navi.38
[17] Xu, L.; Wang, Y.; Sun, H.; Xin, J.; Zheng, N.: Integrated longitudinal and lateral control for Kuafu-II autonomous vehicle, IEEE Transactions on Intelligent Transportation Systems, 17(7), 2016, 2032-2041. https://doi.org/10.1109/TITS.2015.2498170
[18] Yamauchi, B.: All-weather perception for small autonomous UGVs, Proceedings Of SPIE Unmanned Systems Technology X, 696203, April 4, 2008, 1-10. https://doi.org/10.1117/12.776792
[19] Yuan, C.; Liu, H.; Shen, J.; Chen, L.; Jiang, H.: Design and analysis of an auto-braking system controller for autonomous vehicles under the influence of perturbation, IEEE Transactions on Vehicular Technology, 67(3), 2017, 1923-1931. https://doi.org/10.1109/TVT.2017.2771781