ADVANCING AUTONOMOUS DRIVING: TAILORED VIRTUAL PLATFORM FOR RIGOROUS ALGORITHM EVALUATION AND COMPLIANCE

Cătălin-Mihai COSTESCU [1], Marius-Dany BIOLARU-STĂNCULESCU [1], Irina GRĂMESCU [1], Dorin-Mihail DINULESCU [1], George-Stelian PETRAN [1]

[1] Faculty of Mechanical and Mechatronics Engineering,
e-mail: dorin.dinulescu@stud.mec.upb.ro

.

Abstract: *This article introduces a tailored virtual platform for meticulous evaluation of autonomous driving algorithm performance. The simulator rigorously assesses algorithms within an established infrastructure, focusing on identifying and categorizing road markings using a programmer-developed algorithm. It evaluates the algorithm's object recognition via a neural network, providing a holistic assessment of its perceptual acumen. The interactive platform scrutinizes algorithm responses in a dynamic simulated landscape, ensuring adherence to legislative guidelines for autonomous driving. This advanced testing ground validates algorithm robustness and efficacy in recognizing markings, object classification, and simulated navigation while complying with regulatory frameworks. The platform offers vital insights for advancing autonomous driving technology.*

Keywords: *simulator, artificial intelligence, autonomous driving, game engine.*

1. Introduction

Nowadays, autonomous mobile robots play a crucial role in modern logistics, revolutionizing the way goods and people are transported. These robots are extensively utilized in various industries, such as e-commerce, manufacturing, and warehousing, to streamline operations and enhance efficiency. By leveraging advanced technologies like artificial intelligence, machine learning, and computer vision, these robots can navigate complex environments, avoid obstacles, and adapt to dynamic surroundings.

The virtual platform for autonomous driving algorithm performance evaluation is a crucial tool in modern logistics. Simulating real-world scenarios, it allows comprehensive testing and iterative refinement of algorithms without risking physical robots. By assessing adaptability to diverse conditions and vehicle types, it ensures robust and reliable autonomous systems. Valuable data insights optimize efficiency and safety, while fostering collaboration drives innovation. Continuously updating with real-world data keeps the platform relevant and future-proof. This tool accelerates the development of autonomous mobile robots, enabling safer and more efficient transportation of packages, subassemblies, and even people in areas commonly frequented by individuals or marked spaces.

2. Current status

Building upon our prior research, we have introduced a supplementary aspect by incorporating a comprehensive survey of virtual testing environments. This inclusion serves to facilitate the implementation of closed-loop testing protocols throughout the algorithm development phase for autonomous vehicles. The ensuing section provides a compendium of 22 virtual testing

environments, meticulously arranged in alphabetical order, and enriched with corresponding references, web links, and notable features. Additionally, instances where applicable, the names of the providers are duly indicated in parentheses to provide further contextual clarity.

Euro Truck Simulator [1] is a PC video simulation game developed and published by the Czech company SCS Software. The game is available on Windows, Linux, and macOS platforms, allowing players to become truck drivers and drive across Europe, transporting various goods from one point to another.

In general, Euro Truck Simulator is a fun and engaging simulation game for truck and driving enthusiasts. From a technical perspective, it is a driver's perspective driving simulator, which can help users improve their driving skills. However, it is limited in its scope as it cannot test autonomous driving algorithms and includes many elements primarily for player amusement.

RDS_v0810 [2] is a drone simulator developed by the Romanian company RDS Drone Systems. It is designed to assist users in learning and practicing drone flight without the risk of damaging or losing real drones.

The RDS_v0810 simulator can simulate different types of drones, including popular models like DJI Phantom and Mavic. It offers a variety of flight environments, including various weather conditions, terrains, and obstacles, making the flying experience much more realistic and interactive.

Advantages of the RDS_v0810 simulator include:

1. Reduced risks: Users can practice drone flight without worrying about damaging or losing real drones.
2. Lower costs: The RDS_v0810 simulator is much more affordable than purchasing a real drone and can be used anywhere with a computer and an internet connection.
3. Safe learning: Users can practice drone flight in a safe and controlled environment, allowing them to learn techniques and gain experience without endangering people or property.

Disadvantages of the RDS_v0810 simulator include:

1. Reality limitations: Drone simulators are generally more limited than real drones in terms of performance and functionality.
2. Lack of real sensations: The simulator flying experience may differ from flying a real drone since it doesn't provide physical sensations like wind or other external factors that can affect the flight.

Project Cars is a racing video game released in 2015, developed by Slightly Mad Studios and published by Bandai Namco Entertainment. The game is available on PC, PlayStation, and Xbox platforms and offers a racing simulation experience with a wide range of vehicles, tracks, and customization options.

Advantages of Project Cars include superior graphics, a high level of realism and fidelity in racing simulation, a wide variety of available tracks and vehicles, and the ability to customize the car, including mechanical settings, tire selection, and color/design customization.

Disadvantages of the game include a steep learning curve, especially for those unfamiliar with simulation games, and a relatively short campaign mode. Additionally, some reviews have mentioned stability issues and problems with the game's car physics.

3. Solution

In order to enable rapid and cost-effective development, we have created a simulator using the popular Unity [3] video game engine, capable of real-time physics simulation for a car (fig.1). This is achieved by abstracting the car to its fundamental physical elements, described by simple and

computationally efficient mathematical formulas.

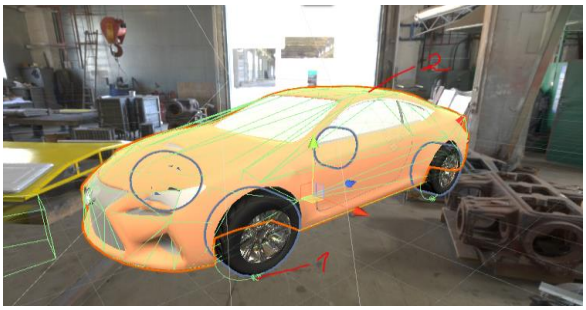

Figure 1: *The car in the simulator*

The car is composed by 3 components:

1. Four wheels defined by a spring and a telescopic mechanism to simulate the suspension, including the point of contact with the road. The wheels have no mass of their own, so the suspension calculation does not consider inertia, but also does not account for friction losses. The contact points continuously check for intersections with the road surface and, in a favorable case, correlate the wheel's tangential velocity with the relative velocity of the rolling material. They influence the suspension and also counteract lateral sliding with a force equal to the product of the coefficient of friction (μ) and the normal force (N_wheel, F_l). The point of contact also serves as the application point for forces generated by the car's engine and brakes. Additionally, it opposes forward movement proportionally to its own velocity to simulate friction losses.
2. The car itself, represented as a point mass and a convex surface to efficiently calculate collisions with the surrounding environment.
3. Control systems. These are responsible for simulating the brakes, engine, and power steering.

The simulator sends data from a video camera, GPS, distance sensor, inertial measurement unit, and rotation encoder to an external program through simulated files in RAM and global mutexes to avoid simultaneous read and write operations that could lead to message degradation.

It employs a deterministic algorithm that utilizes a directed graph with finite states, where each node represents a subprogram, and each edge represents a transition conditioned by the presence of specific stimuli (e.g., markings, objects, or indicators) near the vehicle. This algorithm primarily influences the car's speed, while a local mapping and critical path-finding algorithm ensure adherence to road markings.

At the same time, a neural network is responsible for detecting complex objects that cannot be identified through classical methods.

After analyzing the data, the program sends commands and parameters for controlling the car back to the simulator using the same method. There are several control methods:

1. Direct method: The program directly sends the steering angle and the target speed (maintained by a PID algorithm) to the simulator.
2. Waypoints method (fig.2): The program sends a series of target points that represent a path for the car to follow one by one.

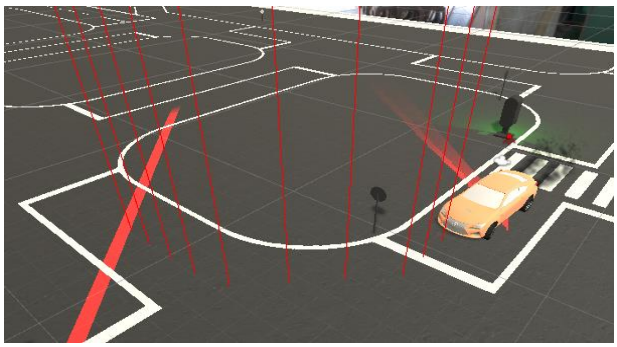

Figure 2: *Exemplification of the waypoint method*

3. Circular Arcs method (fig.3): The program sends a destination point and a direction. The car will then calculate and execute a maneuver consisting of two mutually tangent circular arcs with equal radii. This maneuver will transition the car from its current position and orientation to the specified position and orientation.

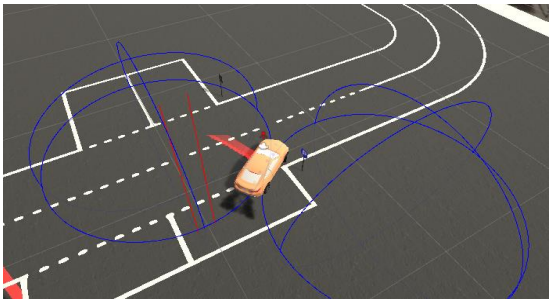

Figure 3: *The arc of circles method.*

4. Local mapping

We chose this method because, for any navigation algorithm, knowing the surrounding environment is essential for planning a collision-free route. This method provides a clear and continuous perspective of the operating environment and is fast and robust. After capturing the image from the overhead-mounted video camera, the program will project it onto a plane perpendicular to the road, providing a clear view of the nearby path (fig.4). This transforms the image into a satellite or bird's-eye view using matrix transformation.

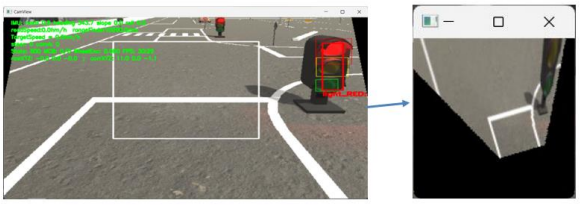

Figure 4: *Bird eye view*

After this, the color white (defined by high brightness and relatively low saturation) is extracted, and then the road markings are classified using a set of criteria such as area, length, thickness, straightness of the line, orientation, etc. Subsequently, the markings are thickened by half of the car's width.

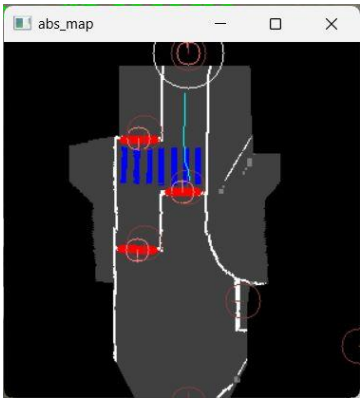

Figure 5: *The line classifier*

Meaning of Colors (fig.5):

1. Blue: pedestrian crossing
2. Red: stop line
3. White: continuous line
4. Light gray: dashed line
5. Dark gray: asphalt with potholes or without
6. Black: outside the field of view.

The algorithm is applied to both road markings and asphalt areas, making it capable of recognizing potential parking spaces as well. In the final stage, the resulting raster is placed on a local map, considering the orientation and global position to align the frames (fig.6). This map is accessed using the formula:

$$Map[(y + 0x01000000) \,\&\, (H - 1)][(x + 0x01000000) \,\&\, (W - 1)]. \quad (1)$$

where W and H are powers of 2, and x, y are global indices of the pixel. This formula acts similarly to a modul operation, but utilizes bitwise AND operations to reduce the processing time to a single cycle .

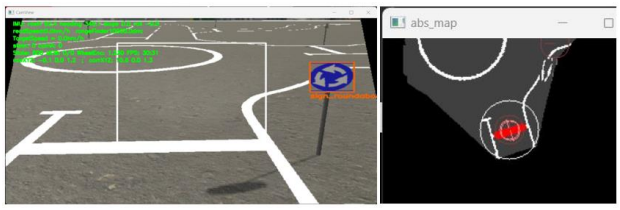

Figure 6: *Car localization based on stop lines*

5. Object Detection

Due to the fact that traffic signs, pedestrian attire, and car models present on the road highly depend on the geographic context and also because classical algorithms that were tested earlier cannot consistently and repeatably recognize signs or obstacles, We have decided to use a YOLOv4Tiny [4] convolutional neural network (fig.7). This choice was made due to its fast runtime, remarkable generalization capacity, and relatively easy and quick training through stochastic gradient descent, requiring only 2000 training cycles (with a batch of 64 annotated images) for each object.

Although the network is relatively easy to train compared to other alternatives, manually creating a dataset for training is very time-consuming.

After balancing the dataset using example censorship and performing 10,200 training cycles through stochastic gradient descent with Darknet library (64 images per cycle), usable results were obtained

.

Figure 7: *YOLOV4TINY detections*

As evident, the network can effectively generalize pedestrians and cars. The only reason we did not proceed with this approach is the need for a powerful, potentially expensive graphics card to achieve real-time processing. However, as we aimed to implement the system on an accessible development board like Raspberry Pi, we decided to create our own neural network [5] with similar runtime and training speed using the gradient descent method with the ADAM optimizer.

Therefore, we opted for training based on our own examples, using a simple yet efficient method of procedural data augmentation.

We developed a program that takes images with transparent backgrounds of the desired objects and randomly places them over backgrounds from a set of images. The objects undergo a composite matrix transformation that randomly rotates them (around each axis), scales, and translates the images. Additionally, they undergo chromatic adjustments to increase object diversity (fig.8).

For three-dimensional objects, multiple images from various angles and lighting conditions (~200 per object) are required, as transparent images are treated as planes (fig.9). However, for signs, being flat objects, only a small number of examples (between 1 and 5) are necessary.

Figure 8: *Input: images with transparency*

Figure 9: *Output: 10,000 images with the given objects and text files describing the center and size of the bounding box for each object.*

We noticed that by introducing objects that should not be detected (such as trees, poles, false indicators like red circles, blue squares, etc.), the number of false detections significantly decreases.

As seen in the above images, pedestrians do not have natural colors. We introduced this condition following a problematic test using the BDD100K [6] dataset, which mainly contains images from Europe. It led to the network's inability to recognize people of different colors or with different clothing styles (both in terms of color and shape). As a

result, we modified the algorithm to generate multi-colored pedestrians. This change guided the network to focus on identifying the main shape, disregarding the color of the pedestrian, thereby reducing the number of required examples.

Once trained, the network is executed on a graphics card or a Tensor [7] unit due to the architecture suitable for running parallelizable subroutines.

6. Calculating the Critical Path Locally

With the raster of the surrounding environment available, we need to navigate the vehicle from point A to point B without colliding with obstacles and minimizing the crossing of restrictive markings. To achieve this, we utilize the Lee algorithm.

Each type of surface is assigned a cost, and then we recursively calculate the gradient of minimum costs for a path from the evaluated point to point B. Finally, starting from point A, we choose and store the neighbor with the minimum cost, thus obtaining a path described by a series of points to be communicated to the car's control system.

7. Calculating the Critical Path Globally

The method mentioned above is not sufficient in most cases, as the destination must be within the vehicle's visual range. Therefore, a directed graph is stored in memory, containing possible routes. The mentioned algorithm is also used here to find the critical path between the nearest node in the graph and the destination. Global points are provided one by one as destinations for the local algorithm. The vehicle moves towards the local target until the distance to it falls below a certain limit, at which point the local target is changed to the next global target (fig.10).

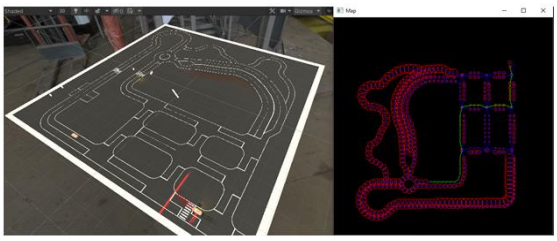

Figure 10: *Local mapping*

8. Conclusion

In the ever-evolving landscape of autonomous driving, our study provides a foundational framework for deeper explorations. The intricacies of these algorithms remain a captivating enigma, urging further investigation to unlock their full potential. Affirmatively, the elimination of human fallibility in driving holds far-reaching implications. It not only promises to extend the golden years of our aging populace but also sets the stage for a paradigm shift in transportation efficiency.

The combination of synchronized vehicles, orchestrated through intersection collision avoidance and convoy-like coordination, heralds a new era of mobility. The canvas of possibilities stretches far beyond highways and byways. The versatile palette of this simulator's applications extends its strokes into domains as diverse as factory logistics, where precision choreography ensures seamless production flows, and mining operations, where safety meets optimization.

Furthermore, the brushstrokes of innovation find resonance in construction and agriculture, rendering new dimensions of potential. Productivity gains flourish alongside diminished operating costs, underpinned by the technological scaffold of the simulator. It is conceivable that the tapestry of safety may be rewoven, as accidents potentially recede in the presence of these augmented systems. As this research catapults us into uncharted terrain, it beckons researchers, industries, and policymakers to unite in nurturing the seeds of progress, redefining the contours of multiple sectors, and shaping a future underpinned by intelligent automation.

References

1. Euro Truck Simulator from https://en.wikipedia.org/wiki/Euro_Truck_Simulator_2
2. RDS drone simulator from https://www.realdronesimulator.com/downloads
3. Unity documentation from https://docs.unity.com/
4. [Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, 2004] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, *"YOLOv4: Optimal Speed and Accuracy of Object Detection,"* arXiv:2004.10934v1 [cs.CV].
5. [Joseph Redmon, 2016] Joseph Redmon, *"Darknet: Open Source Neural Networks in C,"* URL: http://pjreddie.com/darknet/, 2013-2016.
6. [Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, Trevor Darrell, 2020] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, Trevor Darrell, *"BDD100K: A Diverse Driving Dataset for Heterogeneous Multitask Learning,"* in Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2020
7. Tensorflow documentation from https://www.tensorflow.org/api_docs