SoftwareX 26 (2024) 101702

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

Behavior metrics: An open-source assessment tool for autonomous driving tasks

Sergio Paniego *, Roberto Calvo-Palomino, JoséMaría Cañas

Dep. Teoria de la Señal y de las Comunicaciones y Sistemas Telemáticos y Computación, Universidad Rey Juan Carlos, Madrid, Spain

ARTICLE INFO

Keywords:
Evaluation tool
Autonomous driving
Imitation learning

ABSTRACT

The development and validation of autonomous driving solutions require testing broadly in simulation. Addressing this requirement, we present Behavior Metrics (BM) for the quantitative and qualitative assessment and comparison of solutions for the main autonomous driving tasks. This software provides two evaluation pipelines, one with a graphical user interface used for qualitative assessment and the other headless for massive and unattended tests and benchmarks. It generates a series of quantitative metrics complementary to the simulator's, including fine-grained metrics for each particular driving task (lane following, driving in traffic, route navigation, etc.). It provides a deeper and broader understanding of the solutions' performance and allows their comparison and improvement. It uses and supports state-of-the-art open software such as the reference CARLA simulator, the ROS robotics middleware, PyTorch, and TensorFlow deep learning frameworks. BehaviorMetrics is available open-source for the community.

Code metadata

Current code version	v3.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-24-00029
Permanent link to Reproducible Capsule	https://github.com/JdeRobot/BehaviorMetrics/tree/3.0.0
Legal Code License	GPL-3.0 license
Code versioning system used	git
Software code languages, tools, and services used	Python, PyQt5, TensorFlow, PyTorch
Compilation requirements, operating environments & dependencies	GNU/Linux/Windows/OSX, Docker
If available Link to developer documentation/manual	https://jderobot.github.io/BehaviorMetrics/
Support email for questions	sergio.paniego@urjc.es

1. Motivation and significance

Autonomous driving (AD) has gained popularity incrementally in recent years. The adoption of this robotics and AI technology will significantly impact the future [1], emphasizing the critical need for reliable and secure solutions. This popularity has its origins partly in the recent developments in deep learning (DL), helped by the introduction of easy access to high-quality supervised datasets and powerful GPUs.

SAE International's J3016 standard [2] categorizes the vehicle autonomy into six levels ranging from no automation (level 0) to full automation (level 5) where all the driving tasks are performed by the AD system, without any intervention of a driver. Although there are already some examples of level 4 technology, there are still many research advances to be accomplished in this field.

Technically, an AD system can be divided into distinct functions like perception, control, or planning. These functions can be solved individually and then combined into a modular system [3,4] with internal communication for driving autonomously or they can be addressed as an end-to-end problem [5–7], generating directly control commands from the raw input sensory data.

In robotics research and development, it is common to use simulators for validating solutions instead of developing them directly on real robots or vehicles. The simulators allow easier and cheaper iterative development and testing, validating models of solutions in a large number of simulated scenarios before selecting a good one and implementing it in a real scenario (Sim2Real). Final validation there

* Corresponding author.
E-mail addresses: sergio.paniego@urjc.es (Sergio Paniego), roberto.calvo@urjc.es (Roberto Calvo-Palomino), josemaria.plaza@urjc.es (JoséMaría Cañas).

https://doi.org/10.1016/j.softx.2024.101702
Received 15 January 2024; Received in revised form 11 March 2024; Accepted 21 March 2024
Available online 28 March 2024
2352-7110/© 2024 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

can be done much faster and for a largely reduced number of scenarios. The simulated scenarios can be predefined in a digital environment model or they can be imported from real driving tests (Real2Sim).

Various open-source simulators are available, playing a pivotal role in AD research [8,9]. SUMO [10] is specialized in traffic and pursues simulating efficiency and management strategies. TORCS [11] specializes in racing. Gazebo [12] is a general-purpose robotics simulator that has also been employed for AD research [13]. CARLA [14] offers realistic urban driving simulation and is widely used and customizable, offering a wide range of common day-to-day vehicles, sensors, and maps for validating AD solutions. DeepDrive [15], Baidu Apollo [16], Autoware [17], or Udacity's Self-Driving Car Simulator [18] are other possible options, with their strengths and limitations. Gazebo and CARLA are compatible with ROS [19,20], the open-source global standard for robotics middleware. ROS streamlines application development and facilitates seamless transition from simulation to real-world robots.

Besides simulators, publicly available datasets significantly contribute to the advancement of autonomous driving systems by supporting deep learning training for various tasks. For visual perception, we have nuScenes [21], BDD100K [22] or Cityscapes [23], for planning nuPlan [24] or for lane following (lane keeping) commaAI [25].

Considering the wide range of traffic scenarios and driving tasks in AD and the security standards needed [26], robust evaluation metrics are compulsory. While common deep learning (DL) metrics like mean squared error (MSE) or accuracy suffice for static supervised data, they may prove inadequate for fully assessing AD systems. These offline metrics may not capture the dynamic performance across various driving tasks over time intervals (e.g., lane following, obstacle avoidance, intersection navigation, auto parking), necessitating a supplementary evaluation framework. For example, a failure in a single control iteration of an autonomous vehicle could lead to a collision later, despite only occurring in one frame.

This question has already been addressed in the literature and a vast range of metrics and evaluation strategies can be found for different parts of the autonomous driving systems and different situations [27–30].

CARLA simulator already generates metrics that can be used for validating solutions. CARLA Autonomous Driving Leaderboard[1] is an assessment framework and challenge built atop the simulator for evaluating and ranking solutions for AD in urban scenarios and routes. This framework is designed for broadly testing and validating fully AD solutions in a variety of traffic scenarios simultaneously. But it can be overly challenging or even unsuitable when considering developing solutions for specific driving tasks or researchers starting in AD as it lacks fine grain detailed or specific metrics.

In this paper, we introduce Behavior Metrics (BM), a multi-platform open-source software for the assessment of AD solutions in simulation for different driving tasks (currently lane following, driving in traffic, and navigation between points). It assists both everyday users and researchers in developing and validating AD solutions by augmenting simulator-generated metrics with enhanced evaluation metrics.

It supports both CARLA and Gazebo simulators using ROS as communication middleware. It can be used with different sensory inputs for the vehicle like a camera, LIDAR, or any other type of sensory data input supported by the simulators, like the bird-eye-view. They all are managed and added to the simulation using the configuration file. The tool conducts comprehensive online evaluation across driving tasks, yielding objective, fine-grained metrics superior to those provided by simulators. It offers both GUI-based interaction and headless batch processing for large-scale testing.

Designed for the research and advancement of autonomous driving (AD), it establishes a unified framework for evaluation and facilitates the creation and automated execution of extensive benchmarks across various vehicle types, dynamics, lighting conditions, and scenarios. This enables fair comparison among different approaches, including deep learning (DL), reinforcement learning (RL), or explicit programming, providing valuable insights for enhancing each method.

2. Software description

Behavior Metrics' software architecture (Fig. 1) is based on a Model-View-Controller (MVC) design pattern implemented in Python. The evaluation configuration is described in a dynamic YAML configuration file, including the scenario, vehicle, driving task, sensors, and vehicle robot controller. Using this configuration, BM conducts the experimental evaluation, initiating the simulator with the ego vehicle, utilizing the vehicle's robot controller for driving, and ultimately generating comprehensive evaluation metrics for performance insights. The user may change or include any part of the experimental setup like scenario, vehicle (e.g. model), sensor... modifying the configuration file.

The tool supports evaluation in two simulators, CARLA and Gazebo, through integration with ROS 1 Noetic. ROS manages communication between the application and the simulators, allowing for reusable code between the simulators' handlers.

BM communicates with the simulators using the publish/subscribe design pattern of ROS (details in Fig. 2). For example, the application subscribes to the sensor nodes of the ego vehicle to extract the raw data that are then processed by the robot controller and it publishes messages to control the vehicle that are translated to the actual movement of the vehicle in the simulation. BM enables actions like playing or pausing the simulation and controlling simulator processing steps (simulation speed). The raw sensory and simulator data undergo processing to generate evaluation metrics for assessment.

The vehicle controller (Fig. 3) is responsible for the ego vehicle motion. It reads the sensory input provided by the sensors attached to the vehicle, like the camera, bird-eye-view images, ground-truth segmentation camera, or odometry, and processes them. Based on the knowledge extracted from the input, it iteratively generates the control outputs that are commanded to the actuators. Control commands may be generated using a DL model, an RL policy, or even an explicitly programmed algorithm. This abstraction layer facilitates the use of different types of vehicles without any code modification. BM supports the most common DL frameworks (TensorFlow and PyTorch).

It provides control over the simulation time speed, allowing for the selection of either asynchronous or synchronous time modes, and even managing the simulation iteration time-step. By default, the simulator operates asynchronously, making simulator time independent of Behavior Metrics and its vehicle robot controllers. Simulated time becomes crucial for low-resource systems requiring more time for controller iterations. Spending excessive time in this process could result in vehicle control malfunctions, even with correct decisions. Considering the vehicle speed and safety standards needed for AD, the time spent per iteration is crucial. This flexibility enables researchers to test solutions in a broader range of conditions, including simulating systems with limited resources.

BM is compatible with all CARLA towns and allows the management of traffic conditions (traffic lights, traffic signs), simulation start and end points, and weather conditions. With this approach, Behavior Metrics can be used for a full AD agent evaluation or an evaluation of a specific driving task like lane following or driving in traffic. This precise level of control is particularly valuable for researchers focusing on specific driving tasks, where detailed control is essential.

The software is designed to be highly versatile and cross-platform. It achieves this using Docker [31], which facilitates effortless sharing across various operating systems. It is encapsulated within a Docker image, enabling deployment on the most prevalent operating systems. Moreover, the software's core functionality is native to Linux, enabling direct usage on Linux computers without relying on the Docker image. This approach enhances user experience and flexibility.

[1] https://leaderboard.carla.org/

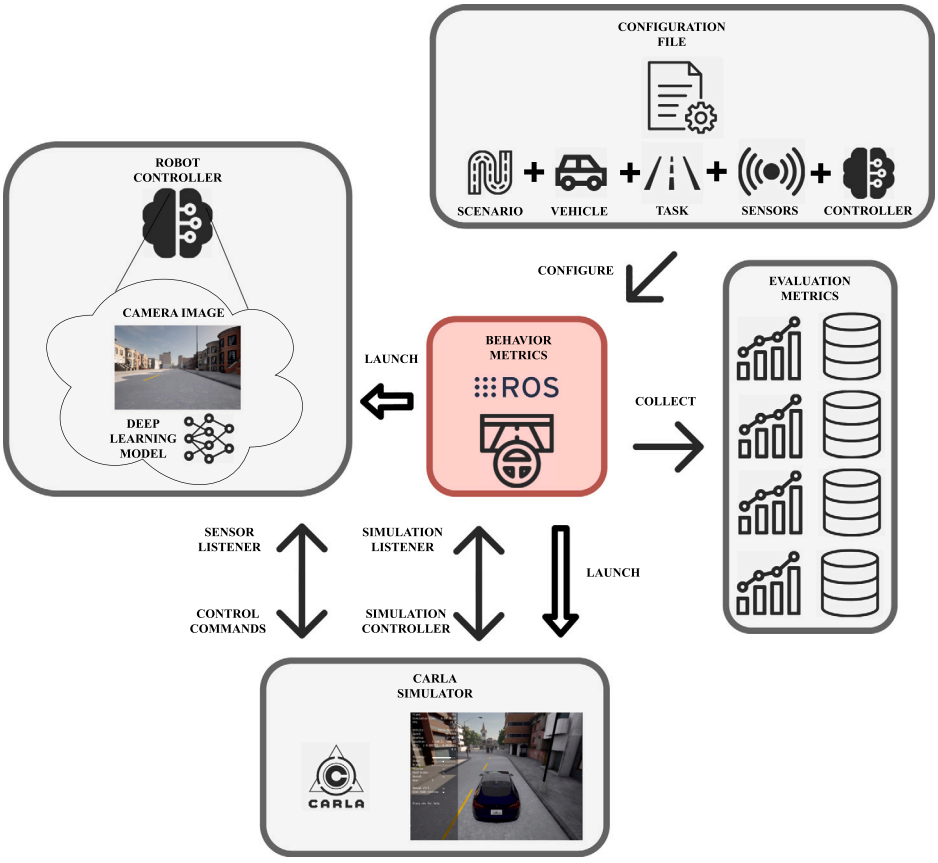

Fig. 1. Behavior Metrics tool architecture. The configuration file describes the setup of the evaluated experiment.

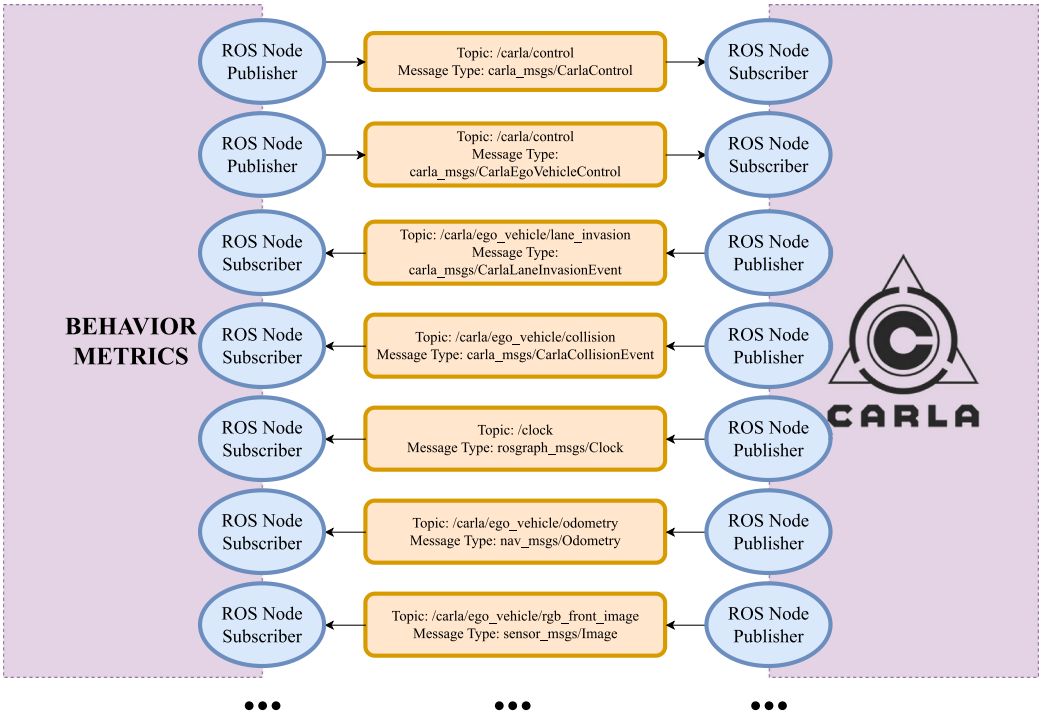

Fig. 2. Some of the connections between BM and CARLA.

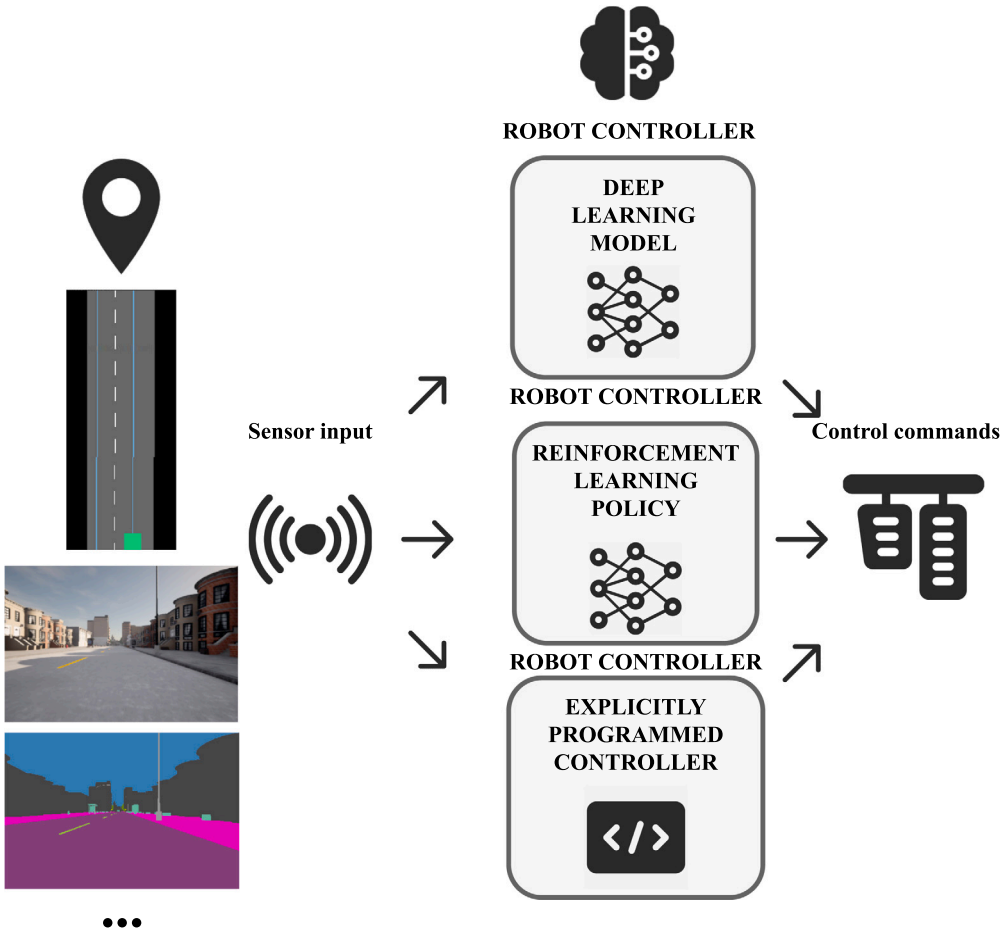

Fig. 3. Details of the robot controller, three types are supported.

2.1. Supported driving tasks

BM supports three distinct driving tasks with varying difficulty levels, contributing to research advancements across different aspects of an AD application: lane following, driving in traffic, and route navigation. The first involves accurately staying within the lane without encroaching on adjacent lanes. The second consists of an ego vehicle and a series of vehicles dispersed along the route, requiring the vehicle to correctly follow the lane while maintaining a safe distance from potential front vehicles. The third combines the previous ones adding a starting and ending point to the experiment setup. The vehicle will drive between goal points while following the lane and maintaining a safe distance from front vehicles. This task also considers traffic lights, traffic signs, and intersections. Other driving tasks can be easily defined along with their metrics since the software tool is provided open-source and its design is modular.

2.2. GUI and headless evaluation modes

The application provides two evaluation modes, the graphical user interface (GUI) mode and the scripted mode (headless). The GUI mode generates a user interface, implemented using the PyQt5 framework, based on a configuration file describing the simulation environment (scenario, ego vehicle, traffic, ...). Using this mode, users can seamlessly execute experiments while visually monitoring the ongoing evaluation process (Fig. 4). Alongside the classic simulator view, this interface displays sensor information and provides convenient buttons for starting, stopping, or restarting the evaluation. This mode is typically employed for qualitative AD solution evaluation. After completing the experiment, quantitative evaluation results are graphically displayed in a separate window and saved in files for future analysis.

In headless mode (Fig. 5) the user defines a configuration file with the evaluated task, all the scenarios, robot controllers, and models to be evaluated, as well as the number of experiment repetitions. Behavior Metrics conducts the experiments as a batch, without user intervention. No graphical part is displayed during the evaluation, and the results are directly saved to files for subsequent analysis. In addition to results from each of the experiments, this mode generates combined results for all the run experiments together, making it easier for the researcher to compare directly how a specific controller behaves. Setting up configuration files in this manner enables BM to support the creation of extensive benchmarks that can be automatically executed.

2.3. Autonomous driving evaluation metrics

BM generates a set of quantitative evaluation metrics that complement those directly provided by the simulator and other evaluation frameworks like the CARLA Leaderboard, offering a more informative and complementary perspective on the behavior of a specific controller. The supplied metrics have been selected as they have been required in several research works. Adding more metrics, such as the CARLA Driving Score, is pretty straightforward as long as the raw data are generated by the simulator.

- **Mean position deviation per km (MPD)**: average deviation, in meters, of the ego vehicle from the center of the lane that it

Fig. 4. Behavior Metrics GUI architecture using CARLA simulator. It displays two separate windows: the application GUI and the simulator.

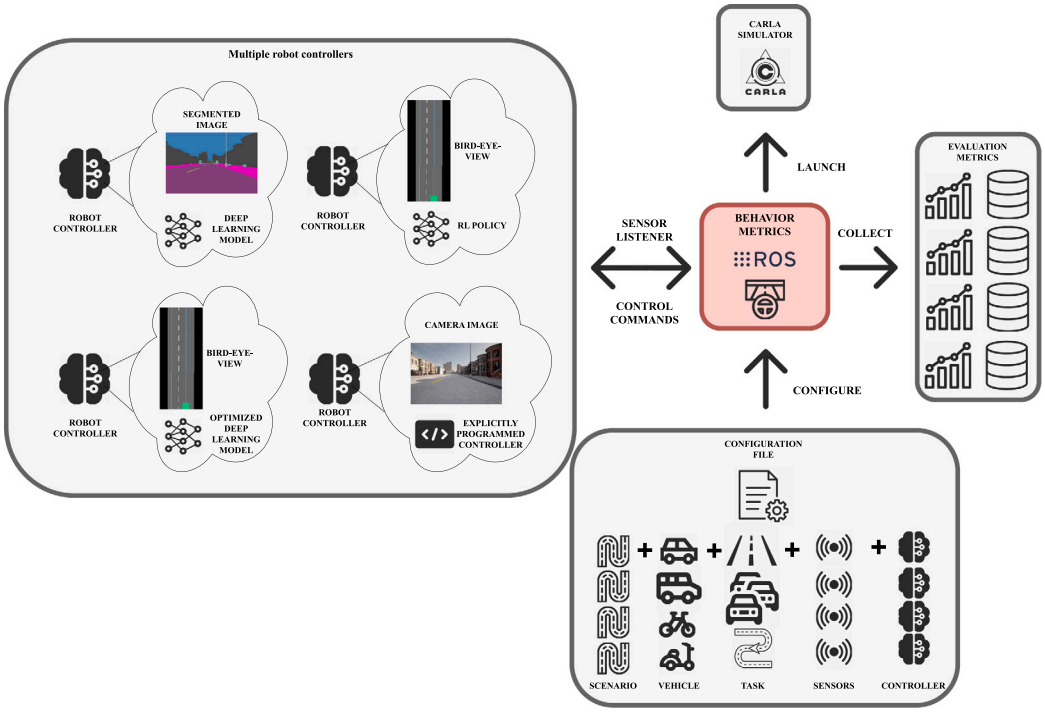

Fig. 5. Behavior Metrics headless evaluation mode.

is traversing. Calculated using the mean of all the points obtained using the minimum Euclidean distance (*MinED*) of each traversed position (*EgoVehiclePosition*) to the center of the lane (*centerOfLane*).

$$\text{MPD} = \frac{1}{N}\sum_{i} \text{MinED}(\text{EgoVehiclePosition}_i, \text{centerOfLane}) \quad (1)$$

- **Effectively completed distance**: distance completed during the experiment in meters that passes through checkpoints that bind its starting and end points. These checkpoints are centered on the lane followed during the experiment, so it is an indicator of how consistently the vehicle drives on the lane.
- **Vehicle longitudinal jerk per km (VJ)**: metric that defines if the vehicle drives smoothly or makes jerks in velocity during the experiment. It indicates whether the conduction is aggressive or smooth. Approximated by calculating the mean of the differences between the current and previous speeds(*VehicleSpeed*).

$$\text{VJ} = \frac{1}{N}\sum_{i} (\text{VehicleSpeed}_i - \text{VehicleSpeed}_{i-1}) \quad (2)$$

- **Robot controller iteration frequency.**
- **GPU inference frequency**: number of GPU iterations per second when the robot controller has a DL model that uses the GPU as the core computational element.
- **Collisions per km**: number of collisions of the ego vehicle during the experiment per kilometer.
- **Lane invasions per km**: number of lane invasions infractions committed by the ego vehicle per kilometer.
- **Distance to the front vehicle**: this metric indicates how close the ego vehicle circulates to other front vehicles and gives insight into whether it follows safety standards or not. The distance is divided into four categories: great distance (20–50 m), medium distance (15–20 m), short distance (6–15 m), and dangerous distance (0-6 m), and it is provided as the percentage of experiment time that the ego vehicle spends on each category.
- **Route completion percentage**: for route navigation task, percentage of route completed for each of the conducted experiments.
- **Average speed**: achieved by the ego vehicle during the experiment.
- **Successful experiments**: this metric is tuned depending on the task. In general, an experiment is considered successful when the safety drive conditions are met but for each specific task, this metric is slightly tuned. For a lane following, for example, BM considers that the vehicle drives at a constant speed and that it does not deviate above a threshold from the middle of the lane. For driving in traffic, BM also considers that the car distance to the front vehicle is not dangerous. For route navigation, BM measures whether the vehicle has reached the goal position following the user's directions.

3. Illustrative examples

The project repository contains example files for each component required for running example evaluations, including the configuration files, robot controllers, and imitation learning-based DL models for supported driving tasks in different simulators. It includes examples for TensorFlow and PyTorch frameworks.

3.1. GUI application example

Using the configuration file edited by the researcher, BM generates the experiment visual setup, including the BM and the simulator windows [32] (Fig. 4). The configuration file is editable to customize the scenario, vehicle, task, included sensors, or vehicle controller. The BM GUI incorporates functionality for initiating the evaluation, watching the sensor state, and the simulated ego vehicle performance. Researchers can then commence the simulation and experiment recording. Upon completion, researchers stop the recording and the simulator, and Behavior Metrics visually presents the results and saves them in log files for subsequent detailed analysis.

3.2. Headless application example

In this case, the configuration file comprises a list of vehicle controllers, scenarios, and experiment repetitions to evaluate. BM evaluates each of the combinations in an unattended manner, without graphical information while evaluating [33] (Fig. 5). Once the experiments conclude, Behavior Metrics furnishes results for each case and the aggregated outcomes for all combinations.

4. Impact

Our contribution impacts the field of AD, which is a relevant research topic as proved by the cited papers and the CARLA Leaderboard challenge. It provides a common framework for testing AD solutions (DL models, RL algorithms, etc.), and supports different driving tasks.

The two evaluation modes, GUI and headless, streamline the process of evaluating and rapidly testing ideas, as well as supporting iterative development of AD solutions. Furthermore, it facilitates a comprehensive evaluation for comparison of different solutions and models. It leverages CARLA, a highly regarded AD simulator, thereby amplifying its impact.

Behavior Metrics has been effectively employed to evaluate end-to-end solutions exploring various concepts. These include assessing the significance of utilizing vehicle controllers with memory and kinematic input, studying the impact of model optimization on controller iteration speed, and analyzing the resulting behavior of the vehicle (published in [34]). It has proved its capabilities in the evaluation of solutions for the driving in traffic task [35], with specific metrics, such as distance to the front vehicle (in peer-review process). And has served as an assessment tool for Google Summer of Code open-source AD projects [36] on route navigation capabilities through metrics like route completion.

5. Conclusions

This paper introduces Behavior Metrics, an open-source evaluation software designed for assessing autonomous driving solutions. We highlight the software's potential utility for researchers in assessing AD solutions across various driving tasks. The included metrics, complementing simulator-provided metrics and unique to our solution, provide a more comprehensive understanding of vehicle performance, contributing to the development of improved solutions for diverse driving scenarios.

Our software offers two distinct pipelines, setting it apart from other solutions: GUI mode and headless mode (benchmarks). They empower researchers to qualitatively and quantitatively test their solutions, facilitating comparisons with alternative approaches. Additionally, our software supports state-of-the-art simulators and DL frameworks broadening its accessibility within the research community.

In future work, we intend to support a wider range of autonomous driving tasks, each one with its corresponding evaluation metrics, such as auto-parking, lane-changing, overtaking, or crossing negotiation.

CRediT authorship contribution statement

Sergio Paniego: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Roberto Calvo-Palomino:** Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **JoséMaría Cañas:** Writing – review & editing, Writing – original draft, Supervision, Resources, Methodology, Investigation, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was partly supported by GAIA project (Ref PLEC2023-010303) "Gestión integral para la prevención, extinción y reforestación debido a incendios forestales" from Spanish Research Agency (AEI) and partly supported by the Google Open Source Programs Office through Google Summer of Code 2022 and 2023.

References

[1] Litman T. Autonomous vehicle implementation predictions implications for transport planning. Tech. rep., Victoria Transport Policy Institute; 2022.
[2] SAE International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. 2021, URL https://www.sae.org/standards/content/j3016_202104/.
[3] Yurtsever E, Lambert J, Carballo A, Takeda K. A survey of autonomous driving: Common practices and emerging technologies. IEEE Access 2020;8:58443–69. http://dx.doi.org/10.1109/ACCESS.2020.2983149.
[4] Gómez-Huélamo C, Diaz-Diaz A, Araluce J, Ortiz ME, Gutiérrez R, Arango F, et al. How to build and validate a safe and reliable autonomous driving stack? A ROS based software modular architecture baseline. In: 2022 IEEE intelligent vehicles symposium. 2022, p. 1282–9. http://dx.doi.org/10.1109/IV51971.2022.9827271.
[5] Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, et al. End to end learning for self-driving cars. 2016, http://dx.doi.org/10.48550/ARXIV.1604.07316, arXiv preprint arXiv:1604.07316.
[6] Toromanoff M, Wirbel E, Moutarde F. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. CVPR, 2020.
[7] Chen L, Wu P, Chitta K, Jaeger B, Geiger A, Li H. End-to-end autonomous driving: Challenges and frontiers. 2023, arXiv preprint arXiv:2306.16927, arXiv:2306.16927.
[8] Shao H, Wang L, Chen R, Waslander SL, Li H, Liu Y. ReasonNet: End-to-end driving with temporal and global reasoning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2023, p. 13723–33.
[9] Wu P, Jia X, Chen L, Yan J, Li H, Qiao Y. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. 2022, http://dx.doi.org/10.48550/ARXIV.2206.08129, arXiv preprint arXiv:2206.08129, arXiv:2206.08129.
[10] Lopez PA, Behrisch M, Bieker-Walz L, Erdmann J, Flötteröd Y-P, Hilbrich R, et al. Microscopic traffic simulation using SUMO. In: The 21st IEEE international conference on intelligent transportation systems. IEEE; 2018, URL https://elib.dlr.de/124092/.
[11] Espié E, Guionneau C, Wymann B, Dimitrakakis C, Coulom R, Sumner A. TORCS, the open racing car simulator. 2005.
[12] Koenig N, Howard A. Design and use paradigms for azebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ international conference on intelligent robots and systems (IROS) (IEEE cat. no.04CH37566), vol. 3, 2004, p. 2149–54. http://dx.doi.org/10.1109/IROS.2004.1389727.
[13] Costa V, Rossetti RJ, Sousa A. Autonomous driving simulator for educational purposes. In: 2016 11th Iberian conference on information systems and technologies. CISTI, 2016, p. 1–5. http://dx.doi.org/10.1109/CISTI.2016.7521461.
[14] Dosovitskiy A, Ros G, Codevilla F, Lopez A, Koltun V. CARLA: An open urban driving simulator. In: Conference on robot learning. PMLR; 2017, p. 1–16.
[15] DeepDrive contributors. Deepdrive. 2022, https://github.com/deepdrive/deepdrive-sim. [Online; Accessed 22 September 2023].
[16] Fan H, Zhu F, Liu C, Zhang L, Zhuang L, Li D, et al. Baidu Apollo EM motion planner. 2018, http://dx.doi.org/10.48550/ARXIV.1807.08048, arXiv preprint arXiv:1807.08048.
[17] Autoware contributors. Autoware. 2022, https://github.com/autowarefoundation/autoware. [Online; accessed 10 January 2024].
[18] Udacity's Self-Driving Car Simulator contributors. Udacity's self-driving car simulator. 2022, https://github.com/udacity/self-driving-car-sim. [Online; accessed 10 January 2024].
[19] Quigley M, Conley K, Gerkey B, Faust J, Foote T, Leibs J, et al. ROS: An open-source robot operating system. In: ICRA workshop on open source software, vol. 3, 2009.
[20] Macenski S, Foote T, Gerkey B, Lalancette C, Woodall W. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics 2022;7(66):eabm6074.
[21] Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, et al. Nuscenes: A multimodal dataset for autonomous driving. In: CVPR. 2020.
[22] Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, et al. BDD100K: A diverse driving dataset for heterogeneous multitask learning. 2018, http://dx.doi.org/10.48550/ARXIV.1805.04687, arXiv preprint arXiv:1805.04687.
[23] Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, et al. The cityscapes dataset for semantic urban scene understanding. 2016, http://dx.doi.org/10.48550/ARXIV.1604.01685, arXiv preprint arXiv:1604.01685, arXiv:1604.01685.
[24] Caesar H, Kabzan J, Tan KS, Fong WK, Wolff E, Lang A, et al. NuPlan: A closed-loop ML-based planning benchmark for autonomous vehicles. 2022, http://dx.doi.org/10.48550/ARXIV.2106.11810, arXiv preprint arXiv:2106.11810, arXiv:2106.11810.
[25] Santana E, Hotz G. Learning a driving simulator. 2016, http://dx.doi.org/10.48550/ARXIV.1608.01230, arXiv preprint arXiv:1608.01230.
[26] Najm WG, Smith JD, Yanagisawa M. Pre-crash scenario typology for crash avoidance research. Tech. rep. DOT-VNTSC-NHTSA-06-02; DOT HS 810 767, John A. Volpe National Transportation Systems Center (U.S.); 2007, URL https://rosap.ntl.bts.gov/view/dot/6281.
[27] Sharath MN, Mehran B. A literature review of performance metrics of automated driving systems for on-road vehicles. Front Future Transp 2021;2. http://dx.doi.org/10.3389/ffutr.2021.759125, URL https://www.frontiersin.org/articles/10.3389/ffutr.2021.759125.
[28] Westhofen L, Neurohr C, Koopmann T, Butz M, Schütt B, Utesch F, et al. Criticality metrics for automated driving: A review and suitability analysis of the state of the art. Arch Comput Methods Eng 2023;30(1):1–35. http://dx.doi.org/10.1007/s11831-022-09788-7.
[29] Paz D, jung Lai P, Chan N, Jiang Y, Christensen HI. Autonomous vehicle benchmarking using unbiased metrics. 2020, arXiv:2006.02518.
[30] Chib PS, Singh P. Recent advancements in end-to-end autonomous driving using deep learning: A survey. 2023, arXiv:2307.04370.
[31] Merkel D. Docker: Lightweight linux containers for consistent development and deployment. Linux J 2014;2014(239):2.
[32] Behavior Metrics contributors. Behavior metrics GUI mode video example. 2024, https://www.youtube.com/watch?v=ze_LDkmCymk. [Online; accessed 10 January 2024].
[33] Behavior Metrics contributors. Behavior metrics headless mode video example. 2024, https://www.youtube.com/watch?v=rcrOF5t3MC4. [Online; accessed 10 January 2024].
[34] Paniego S, Paliwal N, Cañas J. Model optimization in deep learning based robot control for autonomous driving. IEEE Robot Autom Lett 2024;9(1):715–22. http://dx.doi.org/10.1109/LRA.2023.3336244.
[35] Shinohara E. Autonomous driving in traffic using end-to-end deep learning. Master's thesis, Universidad Rey Juan Carlos; 2023.
[36] Zhao M. Obstacle avoidance for autonomous driving in CARLA using segmentation deep learning models. 2023, URL https://theroboticsclub.github.io/gsoc2023-Meiqi_Zhao/blog/.