Advances in Reliability, Safety and Security, Part 10 – Kołowrocki, Kosmowski (eds.)
© 2024 Polish Safety and Reliability Association, Gdynia, ISBN 978-83-68136-22-7 (printed), ISBN 978-83-68136-09-8 (electronic)

Advances in Reliability, Safety and Security

ESREL 2024
Monograph Book Series

Failure Rates Per Time For Autonomous Driving Safety Assessment From CARLA Simulation

Ivo Häring[a], Nikhilesh Sandela[b], Rachana Padariya[c], Georg Vogelbacher[a], Fabian Höflinger[a], Alexander Richter[a], Jörg Finger[a], Aishvarya K. Jain[a], Konstantin Kirchheim[d]

[a]*Franhofer EMI, Freiburg, Germany*
[b]*Work done at Fraunhofer EMI, current affiliation: University Magedburg*
[c]*Work done at Fraunhofer EMI, Master student Hochschule Zittau/Görlitz, Germany*
[d]*Institure for Intelligent Cooperating Systems, Otto von Guerike Univerität Magedburg, Germany*

Abstract

Autonomous Driving simulations like CARLA offer a variety of functionalities including the modeling of subfunctions of autonomous driving such as detection, classification, identification, and tracking of persons, objects and road elements, route planning, and maneuver planning. All of these functionalities operate at high resolution level regarding scenario details such as conditions of weather, daytime, road type, traffic conditions, etc. In contrast, analytical safety assessments like functional Failure Mode and Effects Analysis (FMEA), Fault Tree Analysis (FTA) or Markov modelling and simulation operate at an abstract level in terms of failures per time or conditional failure rates such as failure of object detection given certain weather or daytime conditions. Furthermore, simulation options can be varied over a wide range of parameters and scenario types, including statistical generation of scenarios and active road participants, i.e. spawning of objects. The article presents a quantitative approach to generate failure rates per hour from CARLA simulation for object detection under a range of environmental conditions in terms of precipitation intensity, fog density, and time of the day, single and in combination. Focus is on the derivation of failure rate expressions that are accessible from simulation data. To this end computer vision metrics are used together with additional information available within simulation setup to compute failure rates per hour. Results are presented using sample tables, box plot and violin graphs. The CARLA simulator is used to assess an object detection algorithm that has been fine-tuned with CARLA sample images. It is discussed why the obtained failure rates are consistent but rather high. Further improvement options of the overall approach are provided.

Keywords: Autonomous driving, CARLA simulation, failure rate per time, object detection, quantification, computer vision metrics

1. Introduction

Analytical approaches on autonomous driving (AD) system or functional level such as Failure mode and effects analysis (FMEA), Fault Tree Analysis (FTA) and advanced Markov simulation require as input failure rate data, potentially depending on scenario data, time and mission history. However, such data on the failure rate is only available to a very limited extent in the open scientific literature (Richter et al., 2023). For instance, in Markov simulation modeling mainly informed guess rates are assumed (Althoff and Mergel, 2011; Nyberg, 2018; Kaalen, Nyberg and Bondesson, 2019; Häring et al., 2022, 2023).

The present paper uses simulations within CARLA environment to determine failure rates for adverse environmental conditions. The approach uses short sequences of driving at a crossing setting for the assessment of object detection (Sandela, 2023). The aim is to estimate failure per time of the detection, identification and classification mechanisms. Note that this is different from existing approaches using CARLA that focus on standard computer vision evaluation metrics, e.g. in terms of false negative and positive rates for lane detection (Jeon et al., 2022), comparison of metrics (Schreier et al., 2023), robustness of object segmentation (Thirugnana et al., 2023), or conduct overall black box testing of AD systems (Norden, O'Kelly, and Sinha 2019).

The paper is organized as follows. Section 2 reviews further related literature showing that simulation-based failure assessments focus on the determination of metrics as known from computer vision. Section 3 details the present approach that tries to include scenario information. Section 4 presents mainly tabular simulation results showing that the open source fine-tuned object detection algorithms, result in rather high failure rates per time. Section 5 draws conclusions and proposes further work options.

2. State of the art, gaps addressed and approach overview

There has been a growing emphasis on the use of synthetic datasets and scenario simulation for the validation of safety and reliability in autonomous driving (AD) research. Also real-world data sets are increasingly used, see e.g. (Feng et al., 2021; Karangwa, Liu, and Zeng, 2023) for an overview. However, their limitations in capturing all possible scenarios and complexities faced by autonomous vehicles (AVs) have led researchers to explore alternative approaches. Synthetic datasets generated through scenario simulation, particularly in the popular simulator CARLA (Dosovitskiy et al., 2017; CARLA, 2024), offer a controlled and diverse environment for testing and evaluating AD algorithms. By leveraging synthetic datasets, researchers can assess the performance and reliability of AVs under various environmental conditions and encounter corner cases and edge cases that may not be present in real-world datasets.

The review paper (Niranjan, VinayKarthik and Mohana, 2021) highlights the effectiveness of CARLA simulator for training and testing object detection algorithms in AD. CARLA's open-source nature and realistic environment modeling provide advantages over other simulators. The paper discusses successful implementations of advanced algorithms and ongoing developments in 3D object detection, edge-based frameworks, and LiDAR detection. It also highlights the potential of simulation in AD research. Likewise, (Nalic et al., 2021) is a review of scenario simulation for AD, highlighting the importance of these approaches for evaluating the performance of AVs under different conditions.

Other AD scenario simulation frameworks include RRADS (Real Road Autonomous Driving Simulation) (Baltodano et al., 2015), TORCS (The Open Racing Car Simulator) (Wymann et al., 2014; TORCS, 2024), Udacity Self-driving Car Simulator (UNITY) (Lade et al., 2021; Kulshrestha, 2024), Microsoft AirSim (Stubbs, 2024; Yao et al., 2018), Autoware (Autoware, 2024; Miura et al., 2019), SVL (SVL, 2024) (Seymour, Ho and Luu, 2021), LGSVL Simulator (Rong et al., 2020), and Gazebo (Koenig and Howard, 2004; AbdelHamed, Tewolde and Kwon 2020). Among these, CARLA stands out as a preferred choice for AD scenario simulation, see the reviews (Kaur et al., 2020; Cai et al., 2022; Coelho and Oliveira, 2022; Ren and Xia, 2023). It offers a wide range of features including realistic urban environments, dynamic traffic patterns, and accurate vehicle and sensor models. CARLA's ability to generate synthetic data enables efficient training and testing of ML models. Its comprehensive and customizable environment allows for the generation of diverse scenarios, making it suitable for evaluating AD systems in various conditions. Furthermore, CARLA is open-source and provides an extensive documentation. CARLA is often preferred for research and development of AD due to its high level of realism, representativeness and flexibility.

Operational Design Domain (ODD) specifications of an AD system encompass various factors such as environmental conditions (e.g., weather, lighting/time of the day), driving conditions such as road and traffic conditions (e.g., types of roads, intersections), geographical constraints, speed limits, traffic rules, presence of pedestrians and other vehicles, functional limitations, and other relevant aspects of the operational environment. These factors and conditions define the overall operational capabilities and limitations of the entire system. The present aim is to identify critical scenario factors and determine their impact on time-dependent failure rates, both individually and in combination, using CARLA simulation.

3. Methodology and implementation details

3.1. CARLA Simulation

Basics of scenario simulation include a taxonomy of scenario suitable for scenario description (Weber et al., 2019): functional, logical, and concrete. Functional scenarios are high-level descriptions of driving situations, while logical scenarios describe specific sequences of events, and concrete scenarios are detailed descriptions of a specific environment and the interactions within it. For scenario simulation, functional scenarios can be used to describe the general driving environment and goals of the AD system (e.g., driving safely in a variety of conditions), logical scenarios to provide more detail about how the system processes sensor data and makes decisions, and concrete scenarios to provide specific examples of how the system performs in particular situations (e.g., avoiding a pedestrian in low visibility conditions).

In the following, scenario description encompasses scenario characteristics or elements that refer to the features and attributes of a scenario that are relevant for testing the behavior and performance of an autonomous system. These characteristics include factors such as weather conditions, time of day, road layout, traffic density, and the behavior of other road users.

CARLA (CAR Learning to Act) is a high-fidelity urban driving simulation environment, developed as an open-source layer over Unreal Engine 4 (UE4) (El-Wajeh, Hatton and Lee, 2022; Malik, Khan and El-Sayed, 2022). It is primarily intended for autonomous driving development, testing and validation (Berlincioni, 2022). In addition to open-source code and protocols, CARLA provides open digital assets (urban layouts, buildings, vehicles). It supports flexible specification of sensor suites and environmental conditions. CARLA is used to generate AD scenarios, related camera images, i.e., rendered sequence of video frames, along with corresponding segmented ground truth data.

CARLA has two core modules: the simulator and the Python API and allow users to control the simulations via its API. It provides users with simulated sensor and camera data which can be used to train object detection models or reinforcement learning algorithms to realize autonomous driving. For the purpose of evaluating AD object detection performance. In the present approach sample object data is generated for fine-tuning of an object detection algorithm. Furthermore, the self-driving algorithm is used to generate these images and to generate test scenarios.

3.2. Failure rate per time estimation of object detection system from CARLA simulation for AD

Failure rate is a measure of how often a system or component fails within a given period of time. It is expressed as the number of failures per unit of time, e.g. (Verma, Ajit and Karanki, 2016; Birolini, 2017; Häring, 2021b). Failure rate is generally used to evaluate the reliability of a system or components, and to estimate the likelihood of future failures.

To start from the first principles, the failure probability of a (sub) system is defined as the probability at which it fails within a given period of time $[0, t]$. This corresponds to the cumulative failure probability $F(t)$ based on a failure probability density function (PDF) $f(t) = \int_{t_0=0}^{t} f(\tau)d\tau$. This assumes that a system fails only once. From this, a failure rate $r(t) \equiv z(t) = \frac{f(t)}{R(t)} = \frac{f(t)}{1-F(t)}$, can be computed assuming an underlying distribution. An exponential distribution results in $r(t) = \lambda = const$. This exponential failure rate gives the frequency (in units per time) that a system fails assuming it has survived until time t. We assume that there is a constant failure rate for given combination types of environmental conditions (or factors) such as rain and fog intensity, or time of the day.

In the case of software dominated object detection, it can be assumed that the object detection system operates after each failure “as if it were new”, i.e. without changes. Thus, the number of failures within an observational time interval are aggregated to obtain an average failure rate per time

$$\text{Failure rate per time} = \frac{\text{Failure events within time interval}}{\text{Length of time interval}}. \quad (1)$$

In order to estimate the failure rate of the AD object detection algorithm, first, it is important to understand what constitutes a failure of the detection algorithm. In the context of AD object detection, failure rate could be defined as the frequency with which the algorithm fails to detect objects in its environment. Failure of the detection algorithm can happen because of the underlying failure modes (types of failures, e.g. as distinguished in FMEA (Häring, 2021a)). Essentially, they can occur in three basic ways:

1) False negatives (FN) or “safety-critical misses” occur when the algorithm fails to identify an object that is actually present, see e.g. (Rottmann et al., 2020).
2) False positives (FP) or “safety-critical ghosts” occur when the algorithm identifies an object as present when it is actually not there, see e.g. (Rottmann et al., 2020; Buhler et al., 2020).
3) Out-of-distribution (OoD) (Cui and Wang, 2022) errors occur when the algorithm encounters an object or situation that it has not been trained to detect, and is therefore unable to accurately classify. Typical examples could be: novelties, anomalies, corner cases.

In a simulated environment, it can be assumed that there are likely no inherent OoD errors, since the simulation is customized by the users themselves, and there is usually a well-defined and limited set of possible inputs. Therefore, it is less likely for OoD errors to occur during data generation or performance evaluation within a simulated environment. Hence, for the computation of the failure rate within a simulation environment, it is sufficient to consider the number of false negative (FN) and of false positive (FP) errors occurred in the simulation. In addition, using the total simulation time corresponding to real time simulated, the failure rate reads

$$\text{Failure rate per time} = \frac{\text{Number of FN and of FP within total simulation time}}{\text{Total simulation time}} = \frac{FN+FP}{\text{Total simulation time}}. \tag{2}$$

Failure rate per time could be, e.g., per second, per hour, or per year.

Furthermore, in the context of AD object detection, compared to FPs, FNs could potentially lead to higher critical consequences, such as missing pedestrians, vehicles, or other obstacles that could result in accidents. Thus, assuming FPs are of a lesser concern compared to FNs, they are assumed to be negligible, as argued also in (Aravantinos and Schlicht, 2020). Therefore, focusing specifically on FNs, and ignoring the potential impact of FPs, a non-conservative simplification of (2) reads

$$\text{Failure rate per time} = \frac{FN}{\text{Total simulation time}}. \tag{3}$$

In order to estimate the failure rate, we use the False Negative Rate (FNR), $FNR = FN/P$, resulting in $FN = FNR \cdot P$, where P is the number of positives. Additionally, defining the total simulation time as T, (3) updates to

$$\text{Failure rate per time} = \frac{FNR \cdot P}{T}. \tag{4}$$

Equation (4) can be assumed to give the failure rate for a single simulation. When average values $\overline{FNR}$ and $\bar{P}$ are used resulting from r simulation runs with an average duration of $\bar{T}$, one has

$$\text{Failure rate per time} = \frac{\overline{FNR} \cdot \bar{P}}{r \cdot \bar{T}}. \tag{5}$$

Furthermore, we consider the difference in the time scales in simulation and real-world, accounting for the simulation time-step size, see e.g. (Zapridou, Bartocci and Katsaros, 2020). We use CARLA configuration "synchronous mode + fixed time-step", where one time-step corresponds to $\Delta t_s = 0.05\ s$. Accordingly, for computing the failure rate per time, the average simulation time, $\bar{T}$ can be written as $\bar{T} = \bar{N}_s \cdot \Delta t_s$, where $\bar{N}_s = \text{Average simulation time}/\text{Time-step size}$ is the average number of simulation steps. Now, updating (5) in terms of average number of simulation steps reads

$$\text{Failure rate per time} = \frac{\overline{FNR} \cdot \bar{P}}{r \cdot \bar{N}_s \cdot \Delta t_s}. \tag{6}$$

Therefore, for a single simulation run (i.e., $r = 1$) and without averaging,

$$\text{Failure rate per time} = \frac{FNR \cdot P}{N_s \cdot \Delta t_s}, \tag{7}$$

where $N_s = \text{Simulation time for one scenario}/\text{Time-step size}$. Equation (7) is the final equation for estimating the failure rate per time of the object detection system for a single simulation run.

3.3. Object detection algorithm

For the object detection task, a pre-trained model is used as a starting point for fine-tuning on the custom dataset, rather than training it from scratch (Sandela, 2023). The pre-trained model fasterrcnn_resnet50_fpn of the TorchVision Model zoo (PyTorch, 2024b) is selected since the requirement is only to draw bounding boxes around the detected objects. Furthermore, the pre-trained model needs to be adapted to the custom dataset for fine-tuning. In this case, the entire model needs to be re-trained, as the custom dataset is different from the pre-trained dataset, making it more effective for the custom object detection task. This means updating all the parameters of the model, not just the last few layers. The pre-trained model can be modified in two ways: either by changing the backbone, or by adding a new head (classifier) with the appropriate number of output classes, i.e., a resnet50 backbone and a FastRCNNPredictor head. The latter option is chosen (PyTorch, 2024a).

4. Results and discussion

4.1. CARLA sample simulation results

Figure 1 shows the implementation results on test dataset: one scenario out of 450 scenarios with variations of environment factors, with original RGB color image (left) and corresponding segmentation ground truth (right).

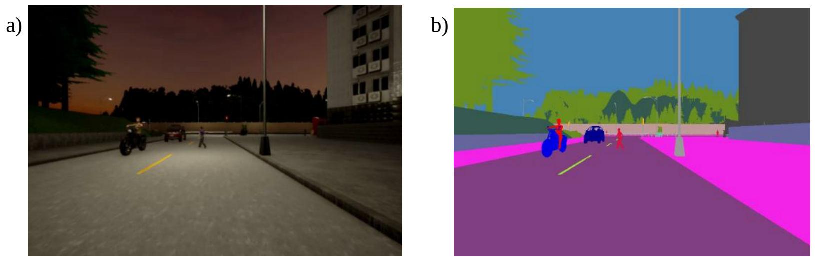


Fig. 1. Final evaluation results of the object detection performance on test dataset (on 1 scenario out of 450 variations of environment factors), with original RGB color image (a) and corresponding segmentation ground truth (b).

Table 1 lists a sample record of the results of representative AD object detection performance evaluation under the influence of a range of environment factors. Object detection metric used are for average precision (AP), and similarly for average recall (AR):

- mAP: This is the standard mean average precision (mAP) metric calculated over all classes and intersection over union (IoU) thresholds for all object sizes, see e.g. (Padilla, Netto, and Da Silva 2020).
- mAP_50 and mAP_75: are the mAP with an IoU threshold of 0.5=50% and 0.75, respectively.
- mAP_small, mAP_medium, mAP_large: They give the mAP of the model on small, medium, and large objects, respectively. The size categories are usually defined based on the relative object's area in the image.

Table 1 displays cells highlighted in red (fog_density=100, sun_altitude_angle=90, precipitation=100) indicating the most challenging environmental conditions where there is high precipitation, a high sun altitude angle, and high fog density, suggesting a strong impact on the model's performance, evident from the lowest value of mAP= 0.04763. Likewise, the cells highlighted in green (fog_density=0, sun_altitude_angle=0, precipitation=0) indicating normal or optimal environmental conditions, having the highest value of mAP= 0.33665. It can also be seen that, while object localization metrics mAR_1 is significantly lower, the mAP_small and mAR_small are totally zero under challenging conditions, signifying their impact on the model performance. This suggests that the challenging conditions of high fog density, high sun altitude angle, and high precipitation have a detrimental impact on the model's performance. The combined effect of these factors hampers the model's ability to accurately detect objects leading to lower performance across multiple metrics.

On the other hand, the normal conditions of least fog density, low sun altitude angle, and no precipitation have a more favorable effect on the model's performance. Therefore, the model is able to perform better, resulting in higher performance metrics in these conditions. Overall, the given conditions exhibit strong differences in the performance metrics for object detection and localization.

Table 1. A sample of data (14 rows out of 450 rows of raw data) showing the results of the performance evaluation of object detection model using standard evaluation metrics.

runs	map	map_50	map_75	map_small	map_medium	map_large	mar_1	mar_10	mar_100	mar_small	mar_medium	mar_large	fog_density	sun_altitude_angle	precipitation
197	0.147538915	0.251394778	0.13942489	0.086019024	0.093494743	0.379720747	0.169832647	0.170674264	0.170735389	1.06E-01	0.097297303	0.391666651	50	-60	75
198	0.162133619	0.259783268	0.194633439	0.09603367	0.112083167	0.339743614	0.188674614	0.190848529	0.190848529	0.116935492	0.125225231	0.361111104	0	-60	75
199	0.238530055	0.399156928	0.215890646	0.137957335	0.263555318	0.415208876	0.271931171	0.278326541	0.278326541	0.17507267	0.274747461	0.431250006	75	0	25
200	0.235754147	0.350545824	0.287370026	0.134833634	0.216462389	0.388441861	0.260213941	0.261404425	0.261404425	0.1592305	0.228828818	0.400000006	0	-60	50
201	0.049519613	0.058370978	0.058368698	3.08E-07	0.054707639	0.429438949	0.051449277	0.051464617	0.051464617	0.0000179	0.056756757	0.438888878	100	90	100
202	0.06103351	0.070262596	0.069148183	3.67E-06	0.082738817	0.43062529	0.062137675	0.063801229	0.063801229	0.000260432	0.086036034	0.444444478	100	90	0
203	0.073818907	0.109287322	0.076324679	0.010266583	0.087317757	0.43772471	0.078312628	0.084352508	0.084352508	0.019431824	0.089639649	0.447222233	50	90	75
204	0.236617833	0.373686522	0.239642039	0.142636225	0.188885093	0.371401221	0.259763867	0.26305443	0.26305443	1.65E-01	0.198989928	0.381249994	25	0	75
205	0.099996038	0.116698995	0.116695464	3.43E-06	0.178947225	0.434420347	0.104123712	0.104197718	0.104197718	7.99E-05	0.190909103	0.447916657	100	90	0
206	0.095710941	0.184228554	0.086262912	0.031943355	0.093271002	0.411169559	0.114329122	0.126829118	0.126829118	0.060887098	0.0954955	0.422222227	100	0	50
207	0.336654097	0.481914848	0.410009772	0.300264269	0.262853652	0.414001703	0.361849666	0.363138348	0.363138348	0.340174407	0.271717161	0.420833349	0	0	0
316	0.047633152	0.055975806	0.055975806	0	0.049035948	0.431456834	0.048322458	0.048456501	0.048456501	0	0.054472707	0.441666692	100	90	100
208	0.185025141	0.303272337	0.203422889	0.08821556	0.184633315	0.371275574	0.210947037	0.213646889	0.213646889	1.09E-01	0.195945948	0.397222221	25	-60	75
209	0.149390429	0.267140538	0.124133147	0.079945594	0.114366569	0.409265667	0.165588349	0.172252744	0.17231448	0.099517949	0.118918926	0.416666657	50	0	50

4.2. Metric analysis using violin plots

Figure 2 shows three violin plots comparing the distribution of model performance (mAP) across different levels of the environmental factors: fog density, precipitation, and sun altitude angle. It allows to analyze the aggregated impact of these factors and understand how they collectively influence the performance.

The violin plot of mAP vs. fog density on the left illustrates the combined effect of multiple factors across different levels of fog density on the performance. The levels of fog density ranging from 0 to 100. Each violin has the information of 90 simulations i.e. a combination of 5 levels of precipitation, 3 different positions of sun each, and 6 pedestrian models, with a particular value of fog density, resulting in 90 combinations

$(1 \cdot 5 \cdot 3 \cdot 6 = 90)$. Likewise, each violin contains the combined influence of other environmental factors of precipitation, sun altitude angle and pedestrian models alongside a particular level of fog density. i.e., it represents the combined influence of multiple factors, not just the isolated effect of fog density. As can be seen, the performance exhibits a wide range of distribution, as the plots demonstrate distinct patterns for different levels of fog density. Also, there is a clear difference in the distribution of mAP values between the lowest and highest levels of fog density. Additionally, the performance is rather lower at higher levels of fog density, with a wider distribution of values. At lower levels of fog density, the performance is clearly higher and more tightly clustered. Overall, this suggests that the performance is significantly affected by the influence of fog density, in combination with other factors, resulting in a range of outcomes across different scenarios.

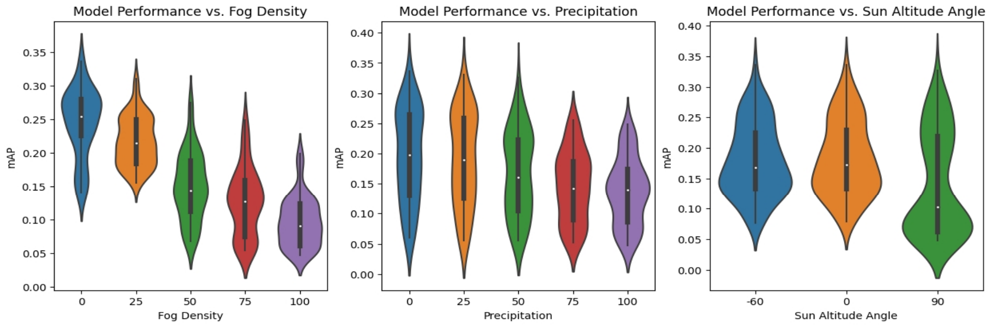


Fig. 2. Violin plots comparing the influence of different environment factors on the detection performance. The plots from left to right illustrate: mAP vs. Fog density, mAP vs. Precipitation and mAP vs. Sun altitude angle, respectively.

The violin plot in the center shows the distribution of mAP values across different levels of precipitation. It can be seen that there is no clear trend in the distribution of mAP values across different levels of precipitation. In this case, each violin is again an outcome of 90 combinations $(1 \cdot 5 \cdot 3 \cdot 6 = 90)$. As can be seen, the mAP values appear to be relatively evenly distributed across the different levels of precipitation, with a larger range of mAP values for lower levels and a smaller one for higher levels of precipitation. Overall, the performance is relatively consistent across different precipitation levels.

The violin plot on the right shows the distribution of mAP vs. sun altitude angle, with midday (90°) being the most challenging condition and sunset (0°) and night (-60°) showing a similar trend with a relatively less influence on the performance. In the plot of mAP vs. sun altitude angle of 90°, the violin contains the combined influence of multiple environmental factors of fog density, precipitation, alongside sun altitude angle of 90° on the performance, i.e., it represents again the combined influence of multiple factors, not just the isolated effect of sun altitude angle of 90°. Accordingly, each violin contains the information of 150 simulations $(1 \cdot 5 \cdot 5 \cdot 6 = 150)$. As can be seen, the violin plot corresponding to sun altitude angle of 90° reveals that the mAP values have a wider spread and lower median compared to the other positions of sun. Surprisingly, the performance is lower during the midday, compared to other times of day, with a broader distribution of values. This indicates that when the sun is at its highest point in the sky during midday, the model's performance is adversely affected. The sun glare at this angle due to strong sun during midday might introduce challenges in object detection, or smaller shadows that cannot be well separated from object shapes leading to lower mAP values. Despite the optimal lighting conditions, the combined effect of the sun altitude angle of 90 with other factors seems to negatively impact the model's performance.

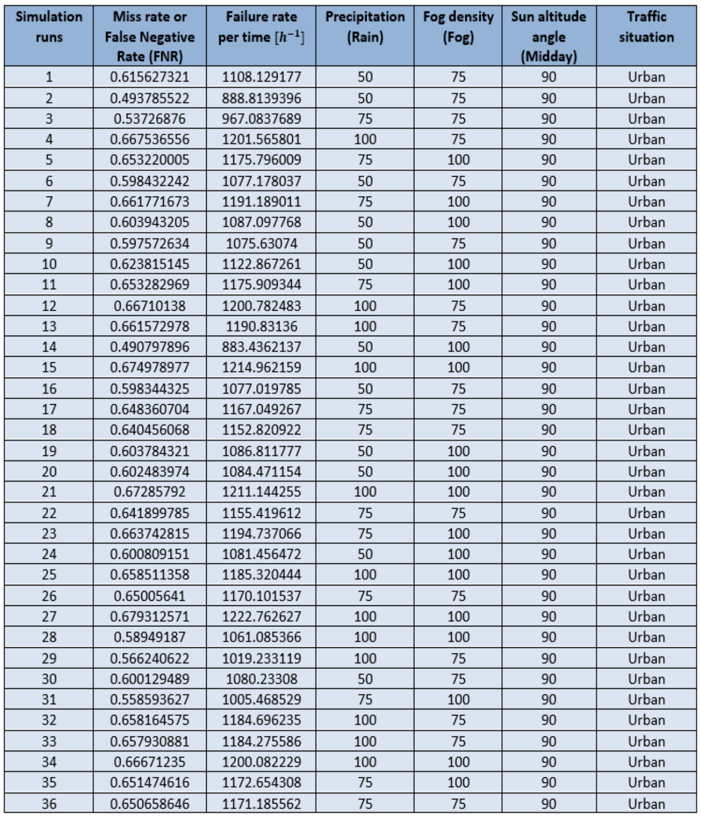
4.3. Failure rate analysis

Table 3 shows the results of the failure rate estimation of object detection system under challenging environmental conditions. It shows the False Negative Rate (FNR) and failure rate values corresponding to the most challenging 36 scenarios out of a total 450 scenarios. They are classified based on their level of influence on the system as discussed in Section 4.2.

In order to illustrate the estimation of failure rate, a sample computation is done for one scenario using (7) with FNR for simulation run 2 from Table 2 below, $FNR = 0.49378$, and $P = 10$ as in Table 1. Number of simulation steps $N_s = 400$ and $\Delta t_s = 0.05\ s$. Failure rate per time $= \frac{FNR \cdot P}{N_s \cdot \Delta t_s} = \frac{0.49378 \cdot 10}{400 \cdot 0.05\ s} = \frac{0.49378 \cdot 10 \cdot 3600}{400 \cdot 0.05\ h} = 888.814\ h^{-1}$.

In Table 2, the lowest failure rates (around 880 h^{-1}) are observed in scenarios 2 and 14, under fog density level of 50, precipitation level of 75, and midday sun angle. These scenarios seem to have the most favorable conditions for object detection, despite the presence of fog. Likewise, scenarios 3, 29, and 31 also exhibit relatively low failure rates, suggesting a reasonable performance. In contrast, the highest failure rates (ranging around 1210 h^{-1} to 1220 h^{-1}) are observed in scenarios 21 and 27. These scenarios represent the most challenging conditions with highest levels of fog density and precipitation. The remaining scenarios fall within the intermediate range of failure rates, demonstrating varying degrees of performance under different combinations of fog density, precipitation, and sun angle.

Table 2. Results of failure rate estimation of object detection system under challenging environmental conditions.



Simulation runs	Miss rate or False Negative Rate (FNR)	Failure rate per time $[h^{-1}]$	Precipitation (Rain)	Fog density (Fog)	Sun altitude angle (Midday)	Traffic situation
1	0.615627321	1108.129177	50	75	90	Urban
2	0.493785522	888.8139396	50	75	90	Urban
3	0.53726876	967.0837689	75	75	90	Urban
4	0.667536556	1201.565801	100	75	90	Urban
5	0.653220005	1175.796009	75	100	90	Urban
6	0.598432242	1077.178037	50	75	90	Urban
7	0.661771673	1191.189011	75	100	90	Urban
8	0.603943205	1087.097768	50	100	90	Urban
9	0.597572634	1075.63074	50	75	90	Urban
10	0.623815145	1122.867261	50	100	90	Urban
11	0.653282969	1175.909344	75	100	90	Urban
12	0.66710138	1200.782483	100	75	90	Urban
13	0.661572978	1190.83136	100	75	90	Urban
14	0.490797896	883.4362137	50	100	90	Urban
15	0.674978977	1214.962159	100	100	90	Urban
16	0.598344325	1077.019785	50	75	90	Urban
17	0.648360704	1167.049267	75	75	90	Urban
18	0.640456068	1152.820922	75	75	90	Urban
19	0.603784321	1086.811777	50	100	90	Urban
20	0.602483974	1084.471154	50	100	90	Urban
21	0.67285792	1211.144255	100	100	90	Urban
22	0.641899785	1155.419612	75	75	90	Urban
23	0.663742815	1194.737066	75	100	90	Urban
24	0.600809151	1081.456472	50	100	90	Urban
25	0.658511358	1185.320444	100	100	90	Urban
26	0.65005641	1170.101537	75	75	90	Urban
27	0.679312571	1222.762627	100	100	90	Urban
28	0.58949187	1061.085366	100	100	90	Urban
29	0.566240622	1019.233119	100	75	90	Urban
30	0.600129489	1080.23308	50	75	90	Urban
31	0.558593627	1005.468529	75	100	90	Urban
32	0.658164575	1184.696235	100	75	90	Urban
33	0.657930881	1184.275586	100	75	90	Urban
34	0.66671235	1200.082229	100	100	90	Urban
35	0.651474616	1172.654308	75	100	90	Urban
36	0.650658646	1171.185562	75	75	90	Urban

Figure 3 illustrates box plots (left) and a corresponding violin plots (right) comparing failure rate of object detection system under normal and challenging environmental conditions.

In Figure 3, the failure rates for challenging scenarios are higher and more concentrated in a narrower range compared to normal scenarios. The median failure rate under challenging environmental conditions is significantly higher, by a factor of ca. 1.6 with a tightly spread interquartile range (IQR), which encompasses the middle 50% of the data, compared to that under normal conditions. As expected from discussion along with Figure 2, the object detection algorithm is more likely to fail when it is exposed to challenging conditions such as extreme rain, dense fog, and midday conditions. Furthermore, the presence of a positive outlier further emphasizes variability of detection under challenging scenarios.

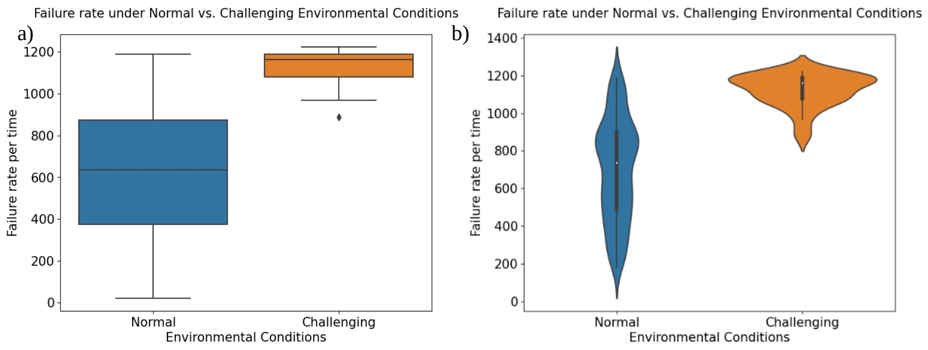


Fig. 3. Box plots (a) and Violin plots (b) comparing failure rate of object detection system under "normal" and "challenging" environmental conditions.

In contrast, the wider IQR and larger overall range with much lower minimum but only marginally lower maximum failure rate values under normal conditions in Figure 3 suggests that the model's performance is rather variable under normal conditions. The improvement is favorable in terms of detecting objects under these conditions. However, the occurrence of high failure rates for a lot of scenarios suggests much room for improvement.

5. Conclusions

On the whole, as expected, the comparison between failure rates under different environmental conditions clearly demonstrates that the object detection system faces more difficulties and exhibits less reliable performance in challenging environmental characterized by extreme rain, dense fog, and midday conditions. This underscores the need to address the challenges posed by these factors in order to enhance the robustness and reliability of the object detection system in challenging environmental scenarios.

Furthermore, the overall very high failure rates observed in the simulation environment emphasize that it is important to understand and improve the current object detection system's performance. Improvement options and strategies could be targeted towards refining algorithms, adjusting sensor configurations, or incorporating advanced sensor technologies to enhance the system's robustness in challenging scenarios to help improve the existing systems. Also, it is important to validate the performance of the object detection system in real-world conditions, given that the environmental factors have a much broader variation and exhibit different patterns and distributions in real-world.

One major factor contributing to the high failure rates is the insufficient training of the object detection algorithm for specific objects as well as specific environmental conditions (fog, rain and midday). Furthermore, the algorithm does not consider the distance of objects as well as whether they are on road or close to a road. It also conducts an assessment per image only. Although these limitations are not critical in terms of the validity of the methodology, they strongly impact the accuracy of failure rate estimates negatively since it is tested if all objects are detected. This explains why the estimated failure rates consistently fall in a high order of magnitude ($10^3\ h^{-1}$).

The simulation results reveal that a considerable number of objects, both static and dynamic, remain undetected within the given time frame. These findings align with the low mean average precision (mAP), which is less than 5% for the very challenging conditions. The low mAP further supports the observation of high failure rates based on false negative rates (FNR) and considering the spawn rate and the time duration of the scenarios. It is worth mentioning that the failure rate estimation assumes a simulation environment that represents realistic urban traffic conditions. However, it is essential to avoid overestimating the detection performance by making unrealistic assumptions, such as assuming fewer objects that need to be detected over the considered time period.

To achieve the results, CARLA simulation and deep learning techniques were integrated to evaluate the perception capabilities of AD systems, particularly in challenging environments. The primary objective was to estimate the failure rate. The CARLA simulator enabled the generation of realistic driving scenarios that can be customized and simulated under various conditions. Deep learning models were then fine-tuned using synthetic data generated from the simulator to assess object detection performance under normal and challenging conditions, with objects of interest, such as pedestrians, traffic signs, traffic lights, and cars (ignoring trucks and motorbikes). The evaluation process consisted of 450 test scenarios, covering a wide range of parameter combinations. These scenarios included variations in fog, rain, lighting conditions, pedestrian models, and pedestrian counts, representing different environmental and contextual conditions that AD systems may encounter in real-world driving situations. Out of the total scenarios, 36 combinations were specifically considered to represent challenging conditions, while the remaining scenarios represented normal conditions.

The results revealed a mean Average Precision (mAP) of 0.33 for object detection under normal conditions and 0.047 under challenging conditions. The estimated failure rates for object detection were determined to be 650 average failures per hour under normal conditions and 1150 average failures per hour under challenging conditions.

Additionally, the evaluation also examined the coverage of the parameter space for object detection. While the evaluation achieved full coverage of the parameter space within the target operational domain design (ODD), it also highlighted the challenges of achieving in-depth and realistic coverage of the overall domain of AD. Furthermore, it revealed that the extensive scope of the domain made it difficult to realistically simulate and assess all possible scenarios. Therefore, it is recommended to adopt an optimized approach that focuses on specific objects of interest within the ODD and scenarios that are particularly relevant, e.g. pedestrian detection under adverse environmental conditions. This targeted approach would allow for a more effective and efficient assessment of the object detection system's performance while maintaining realization feasibility.

Acknowledgements

The presented work is funded by the German BMWK project on Real Driving Validation (RDV) with Grant number 19A21051D and funding duration 2022-2025.

References

Abdel Hamed, A., Tewolde, G., Kwon, J. 2020. Simulation Framework for Development and Testing of Autonomous Vehicles. In: IEMTRONICS (Ed.) 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp 1–6.

Althoff, M., Mergel, A. 2011. Comparison of Markov Chain Abstraction and Monte Carlo Simulation for the Safety Assessment of Autonomous Cars. IEEE Trans. Intell. Transport. Syst. 12, pp. 1237–1247. https://doi.org/10.1109/TITS.2011.2157342.

Aravantinos, V., Schlicht, P. 2020. Making the Relationship between Uncertainty Estimation and Safety Less Uncertain. In: DATE (Ed.) 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp 1139–1144.

Autoware. 2024. Autoware Project: The world's leading open-source software project for autonomous driving. https://autoware.org/. Accessed 2 January 2024.

Baltodano, S., Sibi, S., Martelaro, N., Gowda, N., Ju, W. 2015. The RRADS platform. In: Burnett G (Ed.) Proceedings of the 7th International Conference on Automotive User Interfaces and Interactive Vehicular Applications. ACM, pp 281–288.

Berlincioni, L. 2022. Autonomous Driving Research with CARLA Simulator. SIGMultimedia Rec. 14(1), 2, 8 pp. https://doi.org/10.1145/3630646.3630648

Birolini, A. 2017. Reliability Engineering: Theory and Practice, 8th edn. Springer Berlin Heidelberg.

Buhler, A., Gaidon, A., Cramariuc, A., Ambrus, R., Rosman, G., Burgard, W. 2020. Driving Through Ghosts: Behavioral Cloning with False Positives. In: IROS 2020 (Ed.) 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp 5431–5437.

Cai, J., Deng, W., Guang, H., Wang, Y., Li, J., Ding, J. 2022. A Survey on Data-Driven Scenario Generation for Automated Vehicle Testing. Machines. 10(11), 1101, 32 pp. https://doi.org/10.3390/machines10111101.

CARLA. 2024. CARLA: Open-source simulator for autonomous driving research. https://carla.org//. Accessed 2 January 2024.

Coelho, D., Oliveira, M. 2022. A Review of End-to-End Autonomous Driving in Urban Environments. IEEE Access. 10, pp 75296–75311. https://doi.org/10.1109/ACCESS.2022.3192019.

Cui, P., Wang, J. 2022. Out-of-Distribution (OOD) Detection Based on Deep Learning: A Review. Electronics. 11(21), 3500, 19 pp. https://doi.org/10.3390/electronics11213500.

Feng, D., Haase-Schutz, C., Rosenbaum, L., Hertlein, H., Glaser, C., Timm, F., Wiesbeck, W., Dietmayer, K. 2021. Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges. IEEE Trans. Intell. Transport. Syst. 22(3), pp1341–1360. https://doi.org/10.1109/TITS.2020.2972974.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V. 2017. CARLA: An Open Urban Driving Simulator. In: CoRL (Ed.) 1st Conference on Robot Learning (CoRL 2017), PMLR 78, pp 1–16.

El-Wajeh, Y.A.M., Hatton, P.V., Lee, N.J. 2022. Unreal Engine 5 and immersive surgical training: translating advances in gaming technology into extended-reality surgical simulation training programmes. Br J Surg 109, pp 470–471. https://doi.org/10.1093/bjs/znac015.

Häring, I. 2021a. Failure Modes and Effects Analysis. In: Häring, I. (Ed.) Technical Safety, Reliability and Resilience: Methods and Processes, 1st edn. Springer Singapore, pp 101–126.

Häring, I. 2021b. Reliability Prediction. In: Häring, I. (Ed.) Technical Safety, Reliability and Resilience: Methods and Processes, 1st edn. Springer Singapore, pp 161–178.

Häring, I., Satsrisakul, Y., Finger, J., Vogelbacher, G., Köpke, C., Höflinger, F. 2022. Advanced Markov modeling and simulation for safety analysis of autonomous driving functions up to SAE 5 for development, approval and main inspection. In: 32-nd ESREL 2022, Leva, M.C., Patelli, E., Podofillini, L., Wilson, S. (Eds.), pp 104–111.

Häring, I., Mopuru, S.K.R., Puig-Walz, T., Dhanani, M., Sandela, N., Jörg, F., Vogelbacher, G., Höflinger, F., Jain, A.K., Richter, A., Kirchheim, K. 2023. Overall Markov diagram design and simulation example for scalable safety analysis of autonomous vehicles. In:

Brito, M.P., Aven, T., Baraldi, P., Čepin, M., Zio, E. (Eds.) The 33rd European Safety and Reliability Conference (ESREL 2023), The Future of Safety in the Reconnected World, pp 2261–2268.

Jeon, H., Kim, Y., Choi, M., Park, D., Son, S., Lee, J., Choi, G., Lim, Y. 2022. CARLA Simulator-Based Evaluation Framework Development of Lane Detection Accuracy Performance Under Sensor Blockage Caused by Heavy Rain for Autonomous Vehicle. IEEE Robot. Autom. Lett. 7, pp 9977–9984. https://doi.org/10.1109/LRA.2022.3192632.

Kaalen, S., Nyberg, M., Bondesson, C. 2019. Tool-Supported Dependability Analysis of Semi-Markov Processes with Application to Autonomous Driving. In: ICSRS (Ed.) 4th International Conference on System Reliability and Safety (ICSRS), pp 126–135.

Karangwa, J., Liu, J., Zeng, Z. 2023. Vehicle Detection for Autonomous Driving: A Review of Algorithms and Datasets. IEEE Trans. Intell. Transport. Syst. 24, pp 11568–11594. https://doi.org/10.1109/TITS.2023.3292278.

Kaur, P., Taghavi, S., Tian, Z., Shi, W. 2020. A Survey on Simulators for Testing Self-Driving Cars. In: MetroCAD (Ed.) 2020 International Conference on Connected and Autonomous Driving (MetroCAD), pp 62–70.

Koenig, N., Howard, A. 2004. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: IROS (Ed.) IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan, pp 2149–2154.

Kulshrestha, S. 2024. A self-driving car simulator built with Unity. https://github.com/udacity/self-driving-car-sim. Accessed 2 January 2024.

Lade, S., Shrivastav, P., Waghmare, S., Hon, S., Waghmode, S., Teli, S. 2021. Simulation of Self Driving Car Using Deep Learning. In: ESCI (Ed.) 2021 International Conference on Emerging Smart Computing and Informatics (ESCI), pp 175–180.

Malik, S., Khan, M.A., El-Sayed, H. 2022. CARLA: Car Learning to Act - An Inside Out. Procedia Computer Science 198, pp 742–749. https://doi.org/10.1016/j.procs.2021.12.316.

Miura, K., Tokunaga, S., Ota, N., Tange, Y., Azumi, T. 2019. Autoware Toolbox. In: Proceedings of the 30th International Workshop on Rapid System Prototyping (RSP'19), pp 8–14.

Nalic, D., Mihalj, T., Bäumler, M., Lehmann, M., Eichberger, A., Bernsteiner, S. 2021. Scenario Based Testing of Automated Driving Systems: A Literature Survey. In: FISITA, CAS (Eds.) FISITA World Congress 2021, 10 pp.

Niranjan, D.R., VinayKarthik, B.C., Mohana. 2021. Deep Learning based Object Detection Model for Autonomous Driving Research using CARLA Simulator. In: ICOSEC (Ed.) 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC), pp 1251–1258.

Norden, J., O'Kelly, M., Sinha, A. 2019. Efficient Black-box Assessment of Autonomous Vehicle Safety. https://arxiv.org/abs/1912.03618 Accessed 4 January 2024. 14 pp.

Nyberg, M. 2018. Safety analysis of autonomous driving using semi-Markov processes, ESREL 2018, Haugen et al. (Eds.), pp 781–788. https://doi.org/10.1201/9781351174664-97.

Padilla, R., Netto, S.L., Da Silva, E.A.B. 2020. A Survey on Performance Metrics for Object-Detection Algorithms. In: IWSSIP (Ed.) 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), pp 237–242

PyTorch. 2024a FasterRCNN_ResNet50_FPN. https://pytorch.org/vision/main/models/generated/torchvision.models.detection.fasterrcnn_resnet50_fpn.html. Accessed 4 January 2024.

PyTorch. 2024b. PyTorch: Models and pre-trained weights. https://pytorch.org/vision/stable/models.html. Accessed 4 January 2024.

Ren, J., Xia, D. 2023. Autonomous Driving Simulator. In: Ren, J., Xia, D. (Eds.) Autonomous driving algorithms and its IC Design. Springer Nature Singapore, pp 153–162.

Richter, A., Puig-Walz, T., Dhanani, M., Häring, I., Vogelbacher, G., Höflinger, F., Finger, J., Stolz, A. 2023. Components and Their Failure Rates in Autonomous Driving. In: Brito MP, Aven T, Baraldi P, Čepin M, Zio E (Eds.) The 33rd European Safety and Reliability Conference (ESREL 2023), The Future of Safety in the Reconnected World, pp 233–240.

Rong, G., Shin, B.H., Tabatabaee, H., Lu, Q., Lemke, S., Mozeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., Agafonov, E., Kim, T.H., Sterner, E., Ushiroda, K., Reyes, M., Zelenkovsky, D., Kim, S. 2020. LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving. In: ITSC (Ed.) International Conference on Intelligent Transportation, pp 1–6.

Rottmann, M., Maag, K., Chan, R., Huger, F., Schlicht, P., Gottschalk, H. 2020. Detection of False Positive and False Negative Samples in Semantic Segmentation. In: DATE (Ed.) 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp 1351–1356.

Sandela, N. 2023. Safety and Reliability Analysis of Autonomous Driving using Markov Modeling and Deep Learning. Master Thesis.

Schreier, T., Renz, K., Geiger, A., Chitta, K. 2023. On Offline Evaluation of 3D Object Detection for Autonomous Driving. In: ICCV (Ed.) IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, pp 4084–4089.

Seymour, J., Ho, D.T.C, Luu, Q.H. 2021. An Empirical Testing of Autonomous Vehicle Simulator System for Urban Driving. In: AITest (Ed.) Artificial Intelligence Testing (AITest), pp 111–117.

Stubbs, P. 2024. Open source simulator for autonomous vehicles built on Unreal Engine / Unity, from Microsoft AI & Research. https://github.com/microsoft/AirSim. Accessed 2 January 2024.

SVL. 2024. SVL Simulator: An Autonomous Vehicle Simulator. https://github.com/lgsvl/simulator#readme. Accessed 2 January 2024

Thirugnana Sambandham, V., Kirchheim, K., Ortmeier, F. 2023. Evaluating and Increasing Segmentation Robustness in CARLA. In: Guiochet J, Tonetta S, Schoitsch E, Roy M, Bitsch F (eds) Computer Safety, Reliability, and Security. SAFECOMP 2023 Workshops, vol 14182. Springer Nature Switzerland, Cham, pp 390–396.

TORCS. 2024. TORCS - The Open Racing Car Simulator. https://sourceforge.net/projects/torcs/. Accessed 2 January 2024.

Verma, A.K., Ajit, S., Karanki, D.R. 2016. Reliability and Safety Engineering. Springer London.

Weber, H., Bock, J., Klimke, J., Roesener, C., Hiller, J., Krajewski, R., Zlocki, A., Eckstein, L. 2019. A framework for definition of logical scenarios for safety assurance of automated driving. Traffic Inj Prev 20:65-70. https://doi.org/10.1080/15389588.2019.1630827.

Wymann, B., Dimitrakakis, C., Sumner, A., Espi, E., Guionneau, C. 2014. TORCS: The open racing car simulator. https://www.cse.chalmers.se/~chrdimi/papers/torcs.pdf. Accessed 2 January 2024.

Yao, S., Zhang, J., Hu, Z., Wang, Y., Zhou, X. 2018. Autonomous-driving vehicle test technology based on virtual reality. The 2nd 2018 Asian Conference on Artificial Intelligence Technology (ACAIT 2018), J. eng. 2018(16), pp 1768–1771. https://doi.org/10.1049/joe.2018.8303.

Zapridou, E., Bartocci, E., Katsaros, P. 2020. Runtime Verification of Autonomous Driving Systems in CARLA. In: Deshmukh J, Ničković D (Eds.) Runtime Verification, vol 12399. Springer International Publishing, Cham, pp 172–183.