Neural Rendering for Safety-critical Autonomous Driving Simulation

William Ljungbergh*,1,2 Adam Tonderski*,1,3 Joakim Johnander1 Holger Caesar4
Kalle Åström3 Michael Felsberg2 Christoffer Petersson1

1Zenseact 2Linköping University 3Lund University 4Delft University of Technology

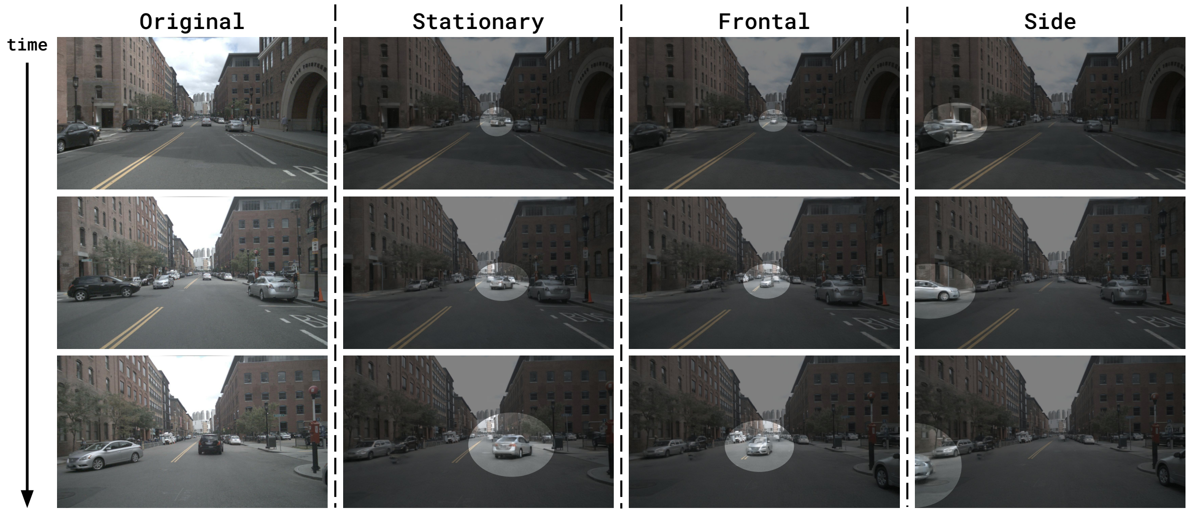


Figure 1. The core idea in this work is to leverage NeRFs to realistically simulate many safety-critical scenarios from a sequence of real-world data. Here we show the original scenario, followed by examples of our three types of collision scenarios: stationary, frontal, and side. The inserted safety-critical actor has been highlighted for illustration purposes. We can generate hundreds of unique scenarios from each log by selecting different actors, jittering their trajectories, and choosing different starting conditions for the ego vehicle. Note that scenarios are not pre-generated, but rather obtained by iteratively generating new images, computing a plan, and acting upon said plan.

Abstract

We present a versatile NeRF-based simulator for testing autonomous driving (AD) software systems, designed with a focus on sensor-realistic closed-loop evaluation and the creation of safety-critical scenarios. The simulator learns from sequences of real-world driving sensor data and enables reconfigurations and renderings of new, unseen scenarios. In this work, we use our simulator to test the responses of AD models to safety-critical scenarios inspired by the European New Car Assessment Programme (Euro NCAP). Our evaluation reveals that, while state-of-the-art end-to-end planners excel in nominal driving scenarios in an open-loop setting, they exhibit critical flaws when navigating our safety-critical scenarios in a closed-loop setting. This highlights the need for advancements in the safety and real-world usability of end-to-end planners. By publicly releasing our simulator and scenarios as an easy-to-run evaluation suite, we invite the research community to validate their AD models in controlled, yet highly configurable and challenging sensor-realistic environments.

* Denotes equal contribution.

1. Introduction

Recent work on autonomous driving (AD) [6, 7] suggests designing and training a holistic neural network for mapping sensor inputs directly to a planned trajectory. Compared to prior work that used modular software stacks, engineered interfaces between modules, or handcrafted rules, this end-to-end approach has several advantages. First, as the driving behavior is learned, the predicted trajectories are expected to resemble how a typical human driver would act. Second, the approach is scalable in the sense that more data leads to more robust as well as generalizable driving performance [1, 7] and in the sense that there is no need to manually design intermediate interfaces or cost functions. The neural network may be divided into modules, but the interfaces between them are learned in order to mitigate information loss.

Hu *et al*. [6] demonstrated that their end-to-end planner, UniAD, performed well on the popular nuScenes [2] planning benchmark. This is an open-loop benchmark, where the tested planner never influences the driving. Instead, the plans are compared to the trajectory taken by the vehicle during data collection and a score is computed based on the similarity between the two. Codevilla *et al*. [3], as well as Dauner *et al*. [4], shed some doubt about the correlation between such an open-loop score and the actual driving performance. This begs the question, how would state-of-the-art end-to-end planners fare if their predicted policy would be acted upon? Unlike regular planners that can be evaluated in a closed-loop manner using straightforward object-level simulations, end-to-end planners require complex sensor simulations to accurately predict their behavior in real-world scenarios. This introduces significant challenges due to the complexity and computational demands of high-fidelity sensor simulation. Moreover, the nuScenes benchmark contains normal driving scenarios, in which no collisions occur. It is unclear how state-of-the-art end-to-end planners would perform in safety-critical scenarios, where a crash is likely unless swift corrective action is undertaken.

In this work, we subject state-of-the-art end-to-end planners to closed-loop evaluation in safety-critical scenarios. Given sensor data, planners predict a plan. The plan is then executed under the constraints of a vehicle model in order to propagate the state of the ego-vehicle forward in time. Given the new state, we use recent advances in neural rendering – NeRFs – to resolve the problem of generating realistic sensor data. These three steps are then repeated until either a crash occurs or we deem the scenario to be over. By executing the predicted plan, we aim to reduce the gap between model evaluation and deployment.

To generate safety-critical scenarios, we take inspiration from the European New Car Assessment Protocol (Euro NCAP) for collision avoidance [5]. This protocol comprises several scenario types that have been identified as safety-critical. These scenario types are rare, but are likely to lead to a collision unless the planner properly deals with them. We craft scenarios by altering recordings of scenes from the nuScenes dataset [2]. We evaluate the driving quality by whether there is a crash, and at what velocity that crash occurs. Our benchmark should be viewed as a necessary but not sufficient condition for high quality driving. To summarize, our contributions are as follows:

1. We release an open source framework for photorealistic closed-loop simulation for autonomous driving.
2. We construct safety-critical scenarios, inspired by the industry standard Euro NCAP, that cannot safely be collected in the real world.

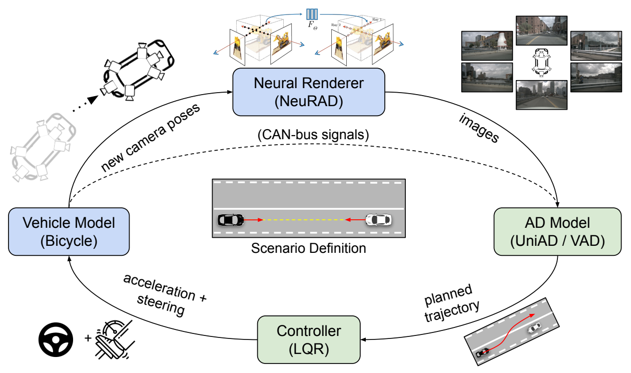


Figure 2. Our closed-loop simulation engine comprises four parts. First, given a driving log, a neural renderer (NeRF) provides photo-realistic images given the ego-vehicle state. Second, an AD model (e.g., the end-to-end planner UniAD [6]) uses these to predict a future ego-trajectory. Third, a controller estimates acceleration and steering signals. Finally, a vehicle model propagates the ego-vehicle state one step into the future. This process is then iterated to achieve closed-loop simulation. Blue indicates simulator, green indicates AD system.

3. Using the simulator and our scenarios, we design a novel evaluation protocol that focuses on collisions rather than displacement metrics.
4. We show that two SotA end-to-end planners fail severely in our safety-critical scenarios despite accurately perceiving the environment.

2. Method

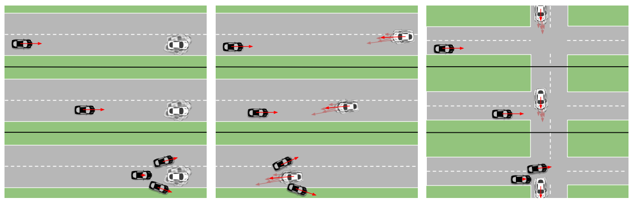
Our end-to-end planning evaluation protocol comprises a closed-loop-simulator (see Section 2.1) and a collision-focused evaluation protocol (see Section 2.2).

2.1. Closed-loop Simulator

Our closed-loop simulator repeatedly performs four steps. First, the neural renderer, trained on a log of real driving data, generates photorealistic images given the ego-vehicle state. Second, an AD model, such as an end-to-end planner, predicts a future ego-vehicle trajectory given the rendered camera input and the ego-vehicle state. Third, a controller converts the planned trajectory to steering and acceleration signals. Fourth, a vehicle model propagates the ego-state forward in time given the control inputs. This procedure is illustrated in Figure 2.

2.2. Evaluation

In contrast to common evaluation practices – *i.e*., averaging performance across large-scale datasets – we instead focus our evaluation on a small set of carefully designed safety-critical scenarios. These scenarios have been crafted such that any model that cannot successfully handle all of them, should be considered unsafe. We have taken inspi-



(a) Stationary scenario. (b) Frontal scenario. (c) Side scenario.

Figure 3. Different scenario types used in the NeuroNCAP evaluation protocol. To increase the robustness of the test and allow for multiple runs, we introduce small random perturbations to the target actor.

ration from the industry standard Euro NCAP testing [5] and define three types of scenarios, each characterized by the behavior of the actor that we are about to collide with: *stationary*, *frontal*, and *side*. The aim is to control the ego-vehicle to avoid a collision with the target actor or at least decrease the collision velocity. See Fig. 3 for an illustration of each scenario type.

For each scenario type, we create multiple scenarios. Each scenario is based on data collected from around 20 seconds of real-world driving. The ego-vehicle and target actor states are initialized such that if current speeds and steering angles are maintained, a collision will occur at approximately 4 seconds into the future. All non-stationary actors are removed from the scene and we randomly select one of these to be the target actor, taking into consideration whether the actor has been observed sufficiently closely, and under the necessary angles, to produce realistic renderings. As our renderer is limited to rigid actors, we exclude pedestrians from this selection. Finally, we randomly jitter the position, rotation, and velocity of the target actor within scenario-specific intervals. During evaluation, we run each scenario for a large number of runs (with a fixed random seed) and compute average results.

NeuroNCAP score: For each scenario, a score is computed. A full score is achieved only by completely avoiding collision. Partial scores are awarded by reducing the impact velocity. In spirit of the 5-star Euro NCAP rating system [5] we compute the NeuroNCAP score (NNS) as

$$\text{NNS} = \begin{cases} 5.0 & \text{if no collision} \\ 4.0 \cdot \max(0, 1 - v_i/v_r) & \text{otherwise} \end{cases}, \quad (1)$$

where v_i is the impact speed as the magnitude of relative velocity between ego-vehicle and colliding actor, and v_r is the reference impact speed that would occur if no action is performed.

3. Experiments

First, we start by outlining the details of our experiments in Sec. 3.1. Next, we show the quantitative results from our NeuroNCAP evaluation along with some qualitative examples in Sec. 3.2.

3.1. Experimental Setting

Dataset: We use nuScenes [2] as it has received the most widespread adaptation for end-to-end planning and features rich urban environments. We choose 14 diverse sequences from the validation set – deemed to be suitable based on the behavior of agents present in the scene – to serve as the basis for our safety-critical scenarios.

Scenarios: Each scenario is designed by hand, considering which actors are suitable for the given sequence, the most reasonable collision trajectories, as well as defining allowed ranges for the different kinds of randomization. During evaluation we run each scenario 100 times (with fixed random seed) and average the results. In total we design 10 stationary, 5 side, and 5 frontal scenarios.

Neural renderer: As our renderer, we opt to use NeuRAD [8], a SotA neural renderer developed specifically for autonomous driving and verified to work well with nuScenes. As pose information in nuScenes is limited to the bird's eye view plane, we employ pose optimization to recover the missing information. Finally, we adopt actor flipping along the symmetry axis [9] to enable realistic rendering of actors from all viewpoints.

AD models: We evaluate two current SotA end-to-end driving models, namely UniAD [6] and VAD [7], according to our proposed evaluation protocol. In both cases, we make use of the pre-trained weights made available by the authors, trained on the same dataset, without any alterations to the configuration of said models. Both of these models consume 360° camera input, along with can-bus signals and a high-level command: *right*, *left*, or *straight*, and output a sequence of waypoints up to 3 seconds into the future.

One major difference between these two models is that UniAD applies a collision-avoidance optimization post-processing step to their predicted trajectory. The optimization is performed using a classical solver with a cost-function based on predicted occupancy and the non-optimized output trajectory. This optimization was shown to drastically decrease the collision-rate when evaluated in open loop, and we can now study it in the more interesting closed-loop setting. To enable more directly comparable analysis, we implement the same collision avoidance optimization for VAD. However, as VAD does not directly predict future occupancy, we rasterize their predicted future objects and use this as the future occupancy. Note that this approach possibly overestimates occupancy, as all future modes are treated as equally likely.

For comparison we implement a naïve baseline method based on the perception outputs of UniAD/VAD. The planning logic is simply a constant velocity model unless we observe an object in a corridor in front of the ego-vehicle, in which case we perform a braking maneuver. The corridor is defined as ± 2 meters in the lateral direction and ranging from 0 to $2v_{ego}$ meters in the longitudinal direction, *i.e.* we brake if we have TTC < 2s with an object in front of us.

3.2. NeuroNCAP Results

We evaluate VAD [7] and UniAD [6], as well as the naïve baseline, on our safety-critical scenarios. We also evaluate both methods with and without perception-based trajectory post-processing. We report the NeuroNCAP score (1) and collision rate per scenario type in Tab. 1. Note that the collision rate is not averaged over time, but is defined as the ratio of scenarios that passed without any collisions.

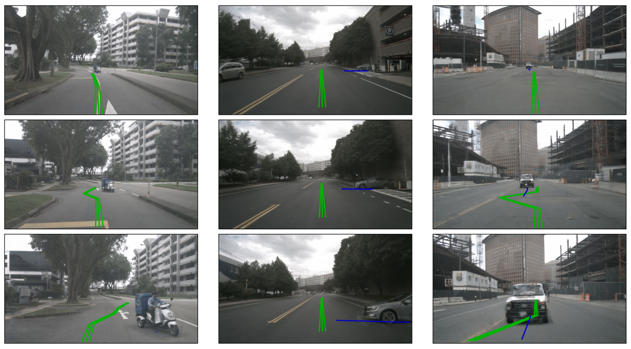
Surprisingly, we find that the plan predicted directly by the network, *i.e.* without post-processing, is extremely unsafe and crashes most of the time, even in the simple stationary scenarios. For reference, the naïve baseline achieves an almost perfect score in the stationary setting, showing both that the perception of these models is not at fault, and that very simple logic can avoid collision. Trajectory post-processing further confirms this, reducing the collision rate dramatically in the stationary setting. Side and frontal scenarios are more difficult to handle with this rule-based logic, and the baseline crashes almost 100% of the time, albeit with a lower impact speed (thus scoring higher). Surprisingly, the end-to-end methods again show almost no reaction to the impending collision, with 98-99% collision rate in frontal scenarios. Trajectory post-processing improves safety somewhat, but is not nearly as effective as in the stationary setting.

We also present some qualitative examples in Fig. 4. Especially Fig. 4b and Fig. 4c are interesting as these are two crash scenarios where it is apparent that the models are driving quite recklessly. Upon inspecting the auxilliary model outputs, which contain detected objects, we find that in both these cases (and most others) the model consistently detected the safety-critical actor but failed to take action.

We believe that these results highlight a drastic flaw in the design or training of current end-to-end autonomous driving systems. Reducing the contradictions between the predicted plan and the auxiliary outputs is a promising area of improvement for future end-to-end planners. Notably, VAD actually attempts to address this by using multiple loss terms that directly encourage the model to output a plan that is consistent with its perception and prediction outputs. However, as our experiments show, this alignment step does not generalize well, at least not to this type of safety-critical scenario.

		NeuroNCAP Score ↑				Collision rate (%) ↓			
Model	Post-proc.	Avg.	Stat.	Frontal	Side	Avg.	Stat.	Frontal	Side
Base-U	-	2.65	4.72	1.80	1.43	69.90	9.60	100.00	100.00
Base-V	-	2.67	4.82	1.85	1.32	68.70	6.00	100.00	100.00
UniAD	x	0.73	0.84	0.10	1.26	88.60	87.80	98.40	79.60
VAD[†]	x	0.66	0.47	0.04	1.45	92.50	96.20	99.60	81.60
UniAD[†]	✓	1.84	3.54	0.66	1.33	68.70	34.80	92.40	78.80
VAD	✓	2.75	3.77	1.44	3.05	50.70	28.70	73.60	49.80

Table 1. NeuroNCAP evaluation results. End-to-end planners fail in novel, critical scenarios. Trajectory post-processing, as proposed in UniAD, helps significantly. The naïve baseline uses the perception of either UniAD (U) or VAD (V) to determine braking. [†]Corresponds to the model's original setting.



(a) stationary (UniAD) (b) side (VAD) (c) frontal (UniAD)

Figure 4. Qualitative examples of three NeuroNCAP scenarios, with projected planning output (green, before controller) and the actual designed future trajectory of the target actor (blue). In some cases the planner reacts successfully (a), does not react at all (b), or attempts to avoid collision but fails (c). Our simulator can accurately render complex actors (a), but sometimes exhibits unrealistic artifacts for very close objects (b) and (c).

4. Conclusion

In conclusion, our simulation environment offers a novel approach for evaluating the safety of autonomous driving models, drawing on real-world sensor data and Euro NCAP-inspired safety protocols. Through the NeuroNCAP framework, which includes stationary, frontal, and side collision scenarios, we have exposed significant vulnerabilities in current SotA planners. These findings not only underline the urgent need for advancements in the safety of end-to-end planners but also suggest promising paths for future research. By making our evaluation suite openly available to the wider research community, we aim to catalyze progress towards safer autonomous driving. Looking ahead, we anticipate evolving the suite to tackle a wider range of scenarios, integrating more refined vehicle models, and employing advanced neural rendering techniques, thereby setting new benchmarks for safety evaluation.

Acknowledgements

We thank Georg Hess, Carl Lindström, and Adam Lilja for fruitful discussions. This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation. Computational resources were provided by NAISS at NSC Berzelius, partially funded by the Swedish Research Council, grant agreement no. 2022- 06725

References

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11621–11631, 2020.

[3] Felipe Codevilla, Antonio M Lopez, Vladlen Koltun, and Alexey Dosovitskiy. On offline evaluation of vision-based driving models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 236–251, 2018.

[4] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. *arXiv preprint arXiv:2306.07962*, 2023.

[5] EuroNCAP. Assessment protocol – safety assist - collision avoidance, 2023.

[6] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17853–17862, 2023.

[7] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. *arXiv preprint arXiv:2303.12077*, 2023.

[8] Adam Tonderski, Carl Lindström, Georg Hess, William Ljungbergh, Lennart Svensson, and Christoffer Petersson. Neurad: Neural rendering for autonomous driving. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[9] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. Unisim: A neural closed-loop sensor simulator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1389–1399, 2023.