DT-Drive: A Tool for Deterministic Replay-Based Testing and Debugging of Autonomous Driving Systems

Sanjeetha Pennada, Matthew Leach, Olek Osikowicz, Donghwan Shin*
{s.pennada,m.leach,amosikowicz1,d.shin}@sheffield.ac.uk
University of Sheffield
Sheffield, UK

Abstract

While simulation is essential for testing autonomous driving systems (ADS), high-fidelity simulators like CARLA often exhibit non-determinism, leading to flaky simulation results. We present DT-Drive, a record-modify-replay framework that ensures deterministic ADS evaluation by decoupling the ego vehicle from the recorded driving scenario. This approach enables "ego-injection" for fair multi-ADS comparisons and "counterfactual replay" via environmental modifications (e.g., weather and traffic), facilitating targeted testing and debugging. Validated on 128 CARLA benchmark scenarios, DT-Drive achieved 100% deterministic evaluation results, effectively eliminating simulator-induced flakiness. We also demonstrate its utility for testing and debugging by conducting a controlled comparison of the TransFuser and TransFuser++ agents under identical and modified environmental conditions.

DT-Drive and the sample data are available at https://github.com/SanjeethaPennada/DT-Drive. The tool demo video is available at https://www.youtube.com/watch?v=wKIefEaQssk.

CCS Concepts

• **Software and its engineering** → **Software testing and debugging**; **Software functional properties**; • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Computing methodologies** → **Simulation evaluation**.

Keywords

Autonomous Driving Systems, Simulation-based Testing, Record and Replay, Flaky Tests

ACM Reference Format:

Sanjeetha Pennada, Matthew Leach, Olek Osikowicz, Donghwan Shin. 2026. DT-Drive: A Tool for Deterministic Replay-Based Testing and Debugging of Autonomous Driving Systems. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

*The first two authors contributed equally to this research.

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Rigorous testing of Autonomous driving systems (ADS) across a wide range of driving scenarios is essential to ensure their safety and reliability. A test scenario defines a specific sequence of events that the ADS must handle, including both *static* entities (e.g., road layouts, traffic lights, and buildings) and *dynamic* entities (e.g., surrounding vehicles and pedestrians). By executing the ADS in such environments, developers can verify whether the system violates safety requirements, such as collisions with other vehicles or pedestrians [1]. Due to the high cost, risks, and limited controllability of real-world testing, simulation-based testing has become a widely adopted approach for evaluating ADS performance [2].

However, high-fidelity driving simulators, such as CARLA [3], do not guarantee deterministic execution [4–6]; for example, Osikowicz et al. [6] empirically revealed that 31.3% of the CARLA benchmark scenarios exhibit potentially flaky behaviour, meaning identical scenarios can produce different safety evaluation outcomes across repeated runs, even when ADS is deterministic.

Record-and-replay (RnR) is a well-established paradigm for enhancing the reproducibility of testing and debugging complex software systems [7]. By capturing runtime events, deterministic replay frameworks enable developers to execute systems offline while strictly preserving their original behaviour [8]. This capability is particularly valuable for isolating transient faults, allowing engineers to repeatedly analyse executions to diagnose bugs that are otherwise extremely difficult to reproduce [9]. Given this proven utility, one might assume that the native RnR functionality provided by modern driving simulators like CARLA could naturally resolve the issue of simulator-induced flakiness. However, while CARLA's native mechanism successfully guarantees a deterministic replay of a previous run, it operates as an immutable playback loop. Crucially, it lacks the architectural flexibility to decouple the recorded scenario from the autonomous driving agent. Consequently, users cannot replace the ADS controlling the ego vehicle during replay, severely restricting the ability to benchmark multiple, distinct ADSs under fully identical environmental conditions. Furthermore, the native RnR system does not support the targeted modification of recorded environmental entities or surrounding traffic. This limitation prevents researchers from conducting counterfactual, "what-if" analyses, such as systematically altering weather variables or removing specific obstacles, which are vital for testing ADS robustness and isolating the causes of safety failures.

To address the challenges, we propose DT-Drive, a record-and-modify, deterministic-replay-based framework for simulation-based ADS testing that can also be easily integrated into existing scenario-generation studies. DT-Drive decouples recorded scenarios from the ego vehicle controller, enabling a new ego-injection during replay

Table 1: Key functionalities of CARLA built-in Record-and-Replay (RnR) and DT-Drive

Feature	CARLA native RnR	DT-Drive
Record and replay	Recording and replaying a driving scenario simulation. Once a recording is generated, replaying it is fully deterministic.	
Ego-injection replay	Not supported.	Swapping the ADS agent during playback to evaluate different systems under identical conditions. Enables fully deterministic multi-ADS comparison.
Counterfactual replay	Not supported.	Modifying recorded environmental entities (e.g., weather, NPCs) to test "what-if" scenarios. Enables follow-up testing and debugging.

while guaranteeing deterministic execution. In addition, it supports counterfactual replay, enabling researchers to systematically vary environmental conditions while preserving reproducibility. Table 1 compares DT-Drive's functionalities with CARLA's native Record and Replay (RnR) system.

We evaluate DT-Drive using 128 scenarios from prior work on simulator flakiness [6], implemented in CARLA [3], with TransFuser++ [10] as the ADS under test. Our results demonstrate that DT-Drive eliminates execution variability observed in standard simulator runs, producing consistent outcomes across repeated executions. Furthermore, we show that DT-Drive enables controlled and fair comparison of multiple ADSs by evaluating both TransFuser++ and TransFuser under identical deterministic scenarios as well as modified simulation environments.

In summary, this paper makes the following contributions:

- We propose DT-Drive, a novel record–modify-deterministic replay–based framework that (a) guarantees deterministic scenario execution, (b) enables interchangeable evaluation of multiple ADSs under identical conditions, and (c) supports controlled scenario modification for systematic experimentation.
- We demonstrate that DT-Drive eliminates simulation flakiness and enables reproducible, fair comparisons of ADSs across both deterministic and modified scenarios. We also show that DT-Drive can be effectively integrated into scenario-generation and analysis workflows, enabling reliable, reproducible evaluation of multiple ADSs under identical or controlled conditions.

2 DT-Drive Framework

Figure 1 shows an overview of DT-Drive. It takes as input a scenario to run (either a new or previously recorded scenario), the ADS under test, and optional scenario modifications (e.g. changing static/dynamic entities from the existing record). It then returns a deterministic ADS evaluation result (e.g. driving score) for a given scenario. Internally, DT-Drive operates through four major stages: *recording* the scenario, *modifying* entities, *replaying* the simulation with a target ADS, and *evaluating* the results.

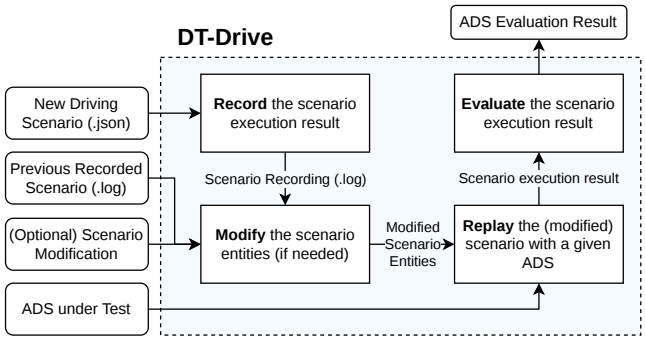

Figure 1: DT-Drive Overview

Stage 1: Record. This stage begins by executing a test scenario (.json) in CARLA, which configures the environment's *static* road layouts and *dynamic* actors, such as pedestrians and surrounding vehicles. While the ADS agent controls the ego vehicle, CARLA's native recorder captures the states and interactions of all entities to produce a deterministic binary log (.log). By utilising this native format, DT-Drive can seamlessly ingest new or existing recordings for subsequent modification and replay.

Stage 2: Modify. This stage enables *counterfactual replay* by allowing users to adjust environmental conditions and entity behaviours. DT-Drive treats each static and dynamic entity as an independent dimension in a scenario vector representation, enabling precise, isolated modifications to the original recording. This capability facilitates "what-if" testing, which is essential for debugging complex interactions and assessing ADS robustness by independently varying scenario elements before re-executing the simulation.

Stage 3: Replay. This stage ensures deterministic execution by configuring CARLA in synchronous mode with a fixed simulation time step. Its primary function is *ego-injection*, where the original ego vehicle is replaced by the ADS under test at identical initial coordinates. While other entities follow trajectories replayed from the log, the ADS interacts with the environment in a closed loop. Because of this closed-loop interaction, execution time is comparable to that of a standard CARLA simulation, unlike the typically faster, open-loop native playback. This setup facilitates a consistent, multi-ADS comparison under identical conditions, allowing developers to isolate performance variances and detect non-deterministic software behaviours.

Stage 4: Evaluate. In the final stage, DT-Drive generates a deterministic ADS evaluation result by calculating a Driving Score based on CARLA Leaderboard standards [11]. This score synthesises route completion with critical safety infractions, such as collisions, traffic violations, and lane departures, to provide a robust performance benchmark. These metrics are exported as a JSON file, allowing developers to analyse agent behaviour and verify safety requirements across identical or modified conditions.

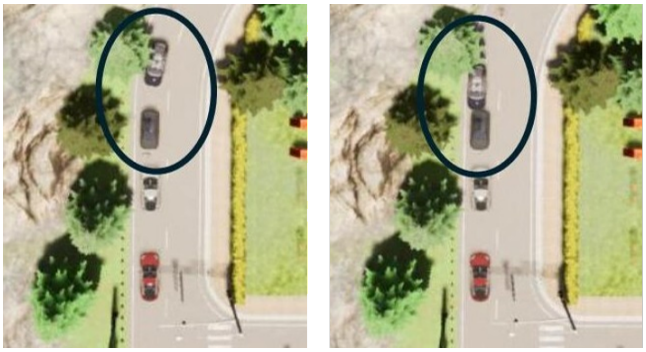

Figure 2: Ego-injection replay results. TransFuser++ (left) safely stopped when obstructed by a vehicle, whereas TransFuser (right) collided with the lead car.

Figure 3: Counterfactual replay results. TransFuser++ (left) maintains its baseline behaviour despite reduced visibility, whereas TransFuser (right) avoids the collision observed in the original scenario, demonstrating a distinct response to environmental modifications.

3 Validation Case Studies

3.1 Experimental Setup

We validated DT-Drive using 128 scenarios from the empirical study by Osikowicz et al. [6], which identified simulator non-determinism in 31.3% of the CARLA 0.9.10.1 Leaderboard benchmark[1] when executed with the TransFuser++ ADS [12]. These scenarios, known to produce flaky outcomes across repeated runs, serve as a baseline to test DT-Drive's ability to ensure determinism. It is measured against the driving score to verify if DT-Drive eliminates the evaluation flakiness inherent in standard CARLA executions.

3.2 Case Study 1: Deterministic Replay

We replayed the 128 scenarios with DT-Drive, including 40 previously identified as flaky due to simulator non-determinism [6], following the same methodology of Osikowicz et al. [6]. While the original study reported flaky test outcomes, DT-Drive produced identical results for all repeated runs as expected. This demonstrates that replaying actor trajectories via DT-Drive effectively eliminates simulator-induced flakiness and enables reproducible assessment of ADS performance.

3.3 Case Study 2: Ego-Injection Replay

We demonstrated DT-Drive's ego-injection capability by evaluating *Scenario 36*, one of the 128 scenarios, using two different ADSs, TransFuser [10] and TransFuser++ [12], under identical deterministic conditions. By replaying fixed actor trajectories, DT-Drive enabled a controlled comparison of their differing behaviours: TransFuser++ safely stopped when obstructed by a vehicle, whereas TransFuser collided with the lead car (Figure 2). Each test was repeated ten times, yielding 100% consistent outcomes. This confirms that DT-Drive facilitates fair, reproducible benchmarking by ensuring that variations in evaluation results stem solely from the ADS logic rather than simulation flakiness.

3.4 Case Study 3: Counterfactual Replay

To demonstrate counterfactual replay, we modified *Scenario 36* by removing street lights within a 20m radius and transitioning the time of day to night to reduce camera visibility. Under these modified conditions, TransFuser++ remained safely stopped, consistent with its original behaviour. However, TransFuser avoided the collision observed in the baseline run (Figure 3). Ten repetitions for each ADS yielded identical results, suggesting that TransFuser's earlier collision was sensitive to visibility conditions. This demonstrates that DT-Drive's counterfactual replay facilitates follow-up testing and debugging by allowing researchers to isolate environmental factors while maintaining strict determinism.

3.5 Integration with Existing Testing Workflows

DT-Drive integrates seamlessly with scenario generation and analysis frameworks, providing a deterministic layer without altering established workflows. By leveraging native simulator recording (see Section 2), the framework can ingest execution traces from any generation pipeline; only a few lines of instrumentation are needed to create a recording log during scenario execution. Once the log file (recording) is provided, users can perform ego injection, apply counterfactual modifications, and maintain original route definitions to ensure a fair, reproducible evaluation using DT-Drive.

This compatibility facilitates diverse research directions, such as scenario simplification [1], where strict determinism is required to verify that behavioural shifts stem solely from controlled modifications rather than simulator noise. Consequently, DT-Drive serves as a general-purpose utility that adds reliability and flexibility to a wide range of ADS testing methodologies, from critical scenario generation to automated debugging.

4 Discussion

4.1 Additional Findings from Case Studies

Non-determinism in Post-Collision. While DT-Drive guarantees deterministic execution for the majority of the simulation, we identified a limitation regarding the ego vehicle's post-collision physical dynamics. In Scenarios 2 and 29, for instance, the ADS acted deterministically and maintained identical trajectories until impact. However, the subsequent rigid-body dynamics varied slightly; in some iterations, the vehicle flipped and recovered, while in others,

[1] Support for the latest CARLA version is planned for a future update.

it remained inverted. These discrepancies stem from the simulator's physics engine. Therefore, the resulting evaluation metrics may diverge if the simulation is allowed to continue after the first collision. This highlights that while DT-Drive isolates ADS behaviour from environmental noise, it cannot fully override the inherent nondeterminism of the underlying physics engine during collision events.

Managing Flaky Behaviour Variants. DT-Drive captures one instance of multiple flaky behaviours into a deterministic record. For example, without the framework, *Scenario 8* exhibited five distinct behaviours across ten runs. Using DT-Drive, a developer can generate five unique logs, each of which can be selected for deterministic replay and analysis. This is particularly useful for debugging simulator errors; for example, one can replay individual logs, identify potential simulation bugs (e.g. a car passing through a static barrier [6]), and effectively remove them from the subsequent ADS testing and debugging. However, distinguishing between physically valid traces and subtle simulation artefacts in general remains an open research challenge.

4.2 Practical Takeaways

DT-Drive provides a practical solution for improving the reliability of simulation-based ADS testing. Based on our findings, we highlight the following key takeaways for researchers and practitioners:

- **Deterministic Debugging:** Developers can reliably reproduce failures by recording and deterministically replaying scenarios, eliminating ambiguity caused by simulator-induced variability.
- **Fair ADS Comparison:** By decoupling scenarios from the ego controller, DT-Drive allows multiple ADSs to be evaluated under identical conditions. This supports rigorous benchmarking and regression testing.
- **Reduced Computational Overhead:** Instead of repeatedly executing the same scenario to account for nondeterminism, practitioners can record only one representative execution and replay it deterministically. This significantly reduces testing cost.
- **Controlled Experimentation:** The framework enables systematic modification of scenarios (e.g., traffic or environment) while preserving determinism, facilitating robustness testing and debugging.

4.3 Limitations and Future Work

Despite its advantages, DT-Drive has a few limitations. First, it deterministically replays a single execution trace at a time, requiring multiple runs to capture the full range of flaky behaviours. Second, because DT-Drive operates on recorded simulator executions, it cannot correct inherent simulation inaccuracies; instead, it reproduces them deterministically.

Currently, the tool is implemented using an earlier version of the CARLA simulator, which fully supports the functionality required for our experiments. Migrating DT-Drive to the latest CARLA release is part of our future work to ensure compatibility with ongoing simulator development.

5 Conclusion

While simulation is a cornerstone of autonomous driving system (ADS) evaluation, its reliability is frequently undermined by simulator-induced non-determinism. This flakiness produces inconsistent outcomes for identical scenario configurations, complicating the validation of safety-critical systems. In this paper, we introduced DT-Drive, a deterministic replay framework that ensures reproducible testing by decoupling the ego vehicle from the recorded environment. By enforcing deterministic simulator settings and utilising a record-modify-replay approach, DT-Drive maintains identical actor trajectories and environmental conditions across repeated executions.

Our evaluation across 128 previously identified flaky scenarios in CARLA demonstrates that DT-Drive effectively eliminates performance variability, yielding 100% consistent results. Furthermore, our case studies confirm that the framework facilitates high-fidelity "what-if" analysis through counterfactual replay and enables fair, ADS-agnostic benchmarking through ego-injection replay.

Acknowledgments

This work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No. RS-2025-02218761, 50%) and by the Engineering and Physical Sciences Research Council (EPSRC) [EP/Y014219/1].

References

[1] D. Shin and S. Pennada, "Towards simplification of failure scenarios for machine learning-enabled autonomous systems," in *2024 IEEE 24th International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*. IEEE, 2024, pp. 1089–1090.

[2] X. Zhang, J. Tao, K. Tan, M. Törngren, J. M. G. Sánchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan *et al.*, "Finding critical scenarios for automated driving systems: A systematic mapping study," *IEEE Transactions on Software Engineering*, vol. 49, no. 3, pp. 991–1026, 2022.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[4] G. Chance, A. Ghobrial, K. McAreavey, S. Lemaignan, T. Pipe, and K. Eder, "On determinism of game engines used for simulation-based autonomous vehicle verification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 20 538–20 552, 2022.

[5] B. A. Jodat, K. Gaaloul, M. Sabetzadeh, and S. Nejati, "Automated test validators for flaky cyber-physical system simulators: Approach and evaluation," *arXiv preprint arXiv:2508.20902*, 2025.

[6] O. Osikowicz, P. McMinn, and D. Shin, "Empirically evaluating flaky tests for autonomous driving systems in simulated environments," in *2025 IEEE/ACM International Flaky Tests Workshop (FTW)*, 2025, pp. 13–20.

[7] Y. Chen, S. Zhang, Q. Guo, L. Li, R. Wu, and T. Chen, "Deterministic replay: A survey," *ACM Computing Surveys (CSUR)*, vol. 48, no. 2, pp. 1–47, 2015.

[8] H. Thane and H. Hansson, "Using deterministic replay for debugging of distributed real-time systems," in *Proceedings 12th Euromicro Conference on Real-Time Systems. Euromicro RTS 2000*, 2000, pp. 265–272.

[9] N. Honarmand and J. Torrellas, "Replay debugging: Leveraging record and replay for program debugging," *ACM SIGARCH Computer Architecture News*, vol. 42, no. 3, pp. 445–456, 2014.

[10] K. Chitta, A. Prakash, B. Jaeger, Z. Yu, K. Renz, and A. Geiger, "Transfuser: Imitation with transformer-based sensor fusion for autonomous driving," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 11, pp. 12 878–12 895, 2023.

[11] CARLA Team, "Get started with leaderboard 1.0," http://leaderboard.carla.org/get_started_v1_0/, 2020, online; accessed 2026.

[12] B. Jaeger, K. Chitta, and A. Geiger, "Hidden biases of end-to-end driving models," in *Proc. of the IEEE International Conf. on Computer Vision (ICCV)*, 2023.