# Cam2Sim: Neural Scenario Reconstruction for Closed-Loop Autonomous Driving Simulation

Davide Jannussi
Politecnico di Torino
Torino, Italy
s331391@studenti.polito.it

Stefano Carlo Lambertenghi
TUM, fortiss
Munich, Germany
stefanocarlo.lambertenghi@tum.de

Constantin Carste
TUM
Munich, Germany
constantin.carste@tum.de

Andrea Stocco
TUM, fortiss
Munich, Germany
andrea.stocco@tum.de

## Abstract

Simulation-based testing enables safe and repeatable evaluation of autonomous driving systems, but its effectiveness is limited by the gap between synthetic simulator outputs and real-world camera observations. To address this problem, we present Cam2Sim, a tool that transforms real-world driving recordings into playable CARLA simulation scenarios. Starting from camera images and poses, Cam2Sim reconstructs road geometry, ego trajectories, parked vehicles, and simulation assets, and augments the reconstructed environment with Gaussian Splatting to render camera observations that resemble the original recording. The framework supports ROS-based data extraction, parked-vehicle detection, OpenStreetMap-based map generation, CARLA scenario construction, Gaussian Splatting training, trajectory replay, and closed-loop execution with a system under test. We validate Cam2Sim on a real-world urban-driving scenario with a camera-based end-to-end driving model, comparing reconstruction quality, image-generation quality, and closed-loop behavior against both a simulation-only baseline and the real-world target. Results show that Gaussian-Splatting-based rendering reduces the visual gap with respect to standard simulator rendering and improves behavioral similarity to the real-world reference runs. The artifact is publicly available at https://github.com/ast-fortiss-tum/cam2sim, and a screencast showing the tool is available at https://youtu.be/KmZ74l1__lI.

## 1 Introduction

Autonomous driving systems (ADS) must be evaluated before deployment because failures can have safety-critical consequences [21]. Real-world testing provides strong evidence, but it is expensive, difficult to reproduce, and unsafe for many corner cases [11, 16].

Simulation-based testing addresses this limitation by enabling the ADS to actively control the ego vehicle in an interactive environment, but it suffers from a significant sim-to-real gap [10, 16]. Differences in textures, illumination, and scene appearance can shift the ADS input distribution and lead to unrealistic behaviors. Recent advances in neural rendering and scene reconstruction, including GANs, diffusion models, Neural Radiance Fields, and Gaussian Splatting, have shown promise in generating visually realistic scenes from real-world recordings [7, 9, 22]. However, existing approaches such as DriveRecon [13] mainly focus on rendering viewpoints or image sequences rather than executable, closed-loop simulations. Conversely, prior work on scenario reconstruction from police reports and structured accident descriptions focuses on rebuilding executable scenarios [6], but does not address the perception-level sim-to-real gap. To the best of our knowledge, there is currently no integrated framework that combines reconstruction and neural rendering to transform real-world 2D driving recordings into playable 3D simulation environments for closed-loop ADS testing.

To address this gap, we present Cam2Sim, the first framework to the best of our knowledge that transforms real-world driving recordings and camera datasets into playable CARLA scenarios. Starting from front-facing camera images and poses, Cam2Sim reconstructs road geometry, ego trajectories, parked vehicles, and simulation assets inside CARLA [4]. The framework then augments the reconstructed environment with Gaussian Splatting [8] to render camera observations that resemble the original recording. The resulting scenarios support both trajectory replay and closed-loop ADS testing, where the ADS receives the GS-rendered camera observations during execution. Cam2Sim supports ROS-based extraction, parked-vehicle detection from camera or LiDAR data, OpenStreetMap-based map generation, CARLA scenario construction, local GS training, trajectory replay, and closed-loop execution with a camera-based ADS. It outputs CARLA-ready maps, trajectories, parked-vehicle placements, GS models, replay videos, steering logs, and execution traces for downstream analysis.

We evaluate Cam2Sim on a real-world urban-driving scenario using a camera-based end-to-end driving model. Our evaluation assesses reconstruction quality and closed-loop behavioral fidelity under simulator-rendered and GS-rendered observations Results show that GS-rendered observations reduce the visual discrepancy with respect to standard simulator rendering while enabling behaviors that more closely resemble the real-world reference runs.

This tool paper makes the following contributions:

**Tool.** We present Cam2Sim, a framework that transforms real-world recordings into executable CARLA scenarios for closed-loop ADS testing through reconstruction and Gaussian-Splatting-based rendering in CARLA.

**Evaluation in a real-world urban driving scenario.** We demonstrate improvements in reconstruction quality and behavioral similarity compared to a simulation-only baseline. In our experiments, the GS-augmented simulation successfully completes all runs, while the simulation-only baseline consistently fails the scenario.

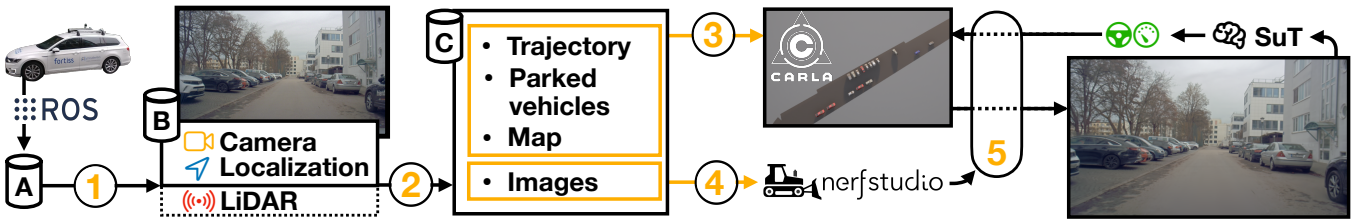


**Figure 1: Overview of Cam2Sim: (1) Data Extraction, (2) Dataset Processing, (3) Simulation Data Generation, (4) Gaussian Splatting Preparation, and (5) Driving Simulation.**

## 2 Cam2Sim Framework

Figure 1 shows the Cam2Sim workflow, which consists of five components: (1) data extraction, (2) dataset processing, (3) simulation-asset generation, (4) Gaussian Splatting preparation, and (5) driving simulation. Together, these components transform raw recordings or prepared camera datasets into CARLA-ready scenarios with map data, trajectories, parked-vehicle positions, GS-training inputs, and executable replay or closed-loop simulations.

### 2.1 Data Extraction

The data extraction component converts ROS-based recordings into the structured dataset format used by Cam2Sim. It can be skipped when front-facing camera images and corresponding camera poses are already available. The minimum required input is a sequence of camera images and poses, which define the camera trajectory and are sufficient for scenario reconstruction, GS training, and replay.

When ROS bags are used, Cam2Sim extracts camera frames and associates each frame with the ego pose at the corresponding timestamp. It can also extract optional LiDAR, odometry, steering angles, and model-prediction streams. These signals improve reconstruction and support validation: LiDAR enables more accurate parked-vehicle detection, while steering and model-prediction logs support comparison with the ADS. Sensor streams with different frequencies are synchronized with ego poses through timestamp-based interpolation. The output is a structured dataset containing camera frames, poses, and point clouds, and trajectories.

### 2.2 Dataset Processing

The dataset processing component extracts the information required for scenario reconstruction, map generation, GS training, and simulation. The camera trajectory is used as the reference path for reconstructing and replaying the scenario.

Cam2Sim detects parked vehicles and provides a graphical interface for manual refinement. From camera data, it applies monocular 3D object detection with FCOS3D [19], transforms detections into world coordinates using camera-to-ego calibration and timestamp-aligned ego poses, and clusters repeated detections across frames. Final positions are computed as confidence-weighted averages [17], while lane-side and orientation labels are obtained through majority voting. When LiDAR is available, Cam2Sim can instead detect parked vehicles with PointPillars [12]; users can then move, rotate, insert, or delete boxes to produce refined parked-vehicle files.

This component also prepares map and GS inputs. From the trajectory or a manually specified address, Cam2Sim retrieves OpenStreetMap data through the Overpass API [5]. For GS training, it selects frames at a configurable interval, removes the visible ego-vehicle hood from the selected frames, generates sky masks using SegFormer trained on Cityscapes [3, 20], and splits the route into overlapping chunks containing images, masks, and pose metadata.

### 2.3 Simulation Data Generation

The simulation-data generation component converts the processed scenario into CARLA-ready assets: the OpenDRIVE map, replay trajectory, initial hero-vehicle pose, and parked-vehicle spawn positions. Cam2Sim exports both center-position and rear-axle trajectory variants to account for the difference between recorded sensor poses and CARLA vehicle transforms.

Coordinate conversion maps dataset positions to CARLA coordinates through the OpenDRIVE map. When geographic coordinates are used, Cam2Sim converts WGS84 coordinates into the OpenDRIVE reference frame and adapts the axis convention for CARLA. The tool also provides utilities to load the generated map, inspect trajectory alignment and parked-vehicle placement, and prepare the final CARLA scenario.

### 2.4 Gaussian Splatting Preparation

Using the processed frames and reconstructed trajectories, the GS preparation component trains the models used to render camera observations that resemble the original recording. Cam2Sim trains local GS models to improve scalability and specialization to local scene appearance.

Cam2Sim runs COLMAP [15] with the selected images, calibrated camera model, and optional sky masks to estimate camera poses and a sparse 3D point cloud. It then trains GS models with Nerfstudio [18], using `splatfacto` or `splatfacto-big`; sky masks exclude sky regions so the model focuses on static scene content. After training, Cam2Sim aligns the Nerfstudio and dataset coordinate systems by matching reconstructed camera poses with the original pose annotations and estimating the corresponding alignment parameters. These models are later used to render GS images from CARLA camera poses.

Because the deployed ADS directly consumes the rendered camera stream, successful closed-loop execution also provides an operational assessment of the generated observations: severe rendering artifacts or missing scene information would directly affect steering behavior and trajectory stability.

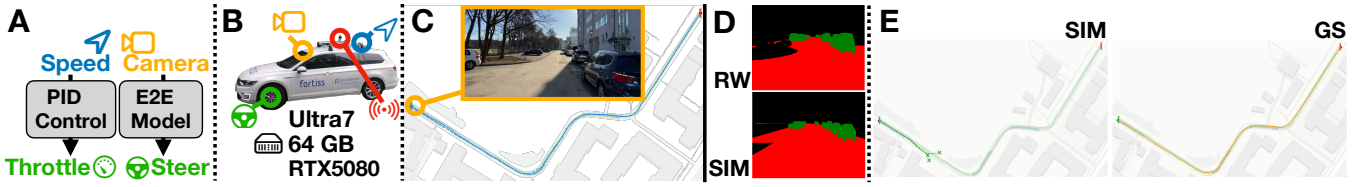


**Figure 2: Validation setup and results: (A) the ADS, (B) the autonomous-driving platform used to collect the target recording, (C) the real-world collection scenario, (D) results of Scenario Reconstruction, and (E) Behavior Fidelity.**

## 2.5 Driving Simulation

The driving simulation component executes the reconstructed scenario in CARLA using the generated map, trajectories, parked vehicles, camera configuration, and, when enabled, the trained GS models and alignment files. Cam2Sim supports both trajectory replay and closed-loop execution.

In trajectory-replay mode, the hero vehicle follows the recorded trajectory frame by frame. This mode is used to inspect the reconstructed scenario and compare observations from the same viewpoints as the original recording. Cam2Sim can save CARLA-rendered images, GS-rendered images, and side-by-side outputs.

In closed-loop mode, the ADS controls the ego vehicle. Cam2Sim currently supports a camera-based DAVE-2 model [1], served by a separate inference process that predicts steering commands from RGB images. The ADS can receive either raw CARLA images, forming the simulation-only baseline, or GS-rendered images to evaluate whether GS-based rendering reduces the gap to the original driving scenario. The output includes replayed camera streams, trajectories, steering commands, and execution logs for evaluating reconstruction quality, visual fidelity, and behavior similarity.

# 3 Preliminary Validation

We evaluate whether Cam2Sim can transform a real-world driving recording into a reconstructed simulation that is geometrically consistent with the original scenario, visually closer to the original camera stream, and suitable for closed-loop ADS testing. The validation follows the structure of the pipeline. First, we assess whether the reconstruction and simulation-generation components correctly reproduce the road geometry, ego trajectory, and parked vehicles of the target scenario. Second, we evaluate whether Gaussian Splatting preserves visually relevant scene information required for closed-loop ADS execution.

To this aim, we conduct a real-world data-collection campaign using an ADS-equipped research vehicle on a public urban road and use the resulting recording as the target scenario for Cam2Sim. The evaluation compares three domains: (i) the real-world reference execution, (ii) a simulation-only baseline using the reconstructed CARLA scenario, and (iii) a GS-augmented simulation in which camera observations are rendered using Gaussian Splatting.

**System Under Test.** We use a camera-based end-to-end DAVE-2 model [1] that predicts continuous steering directly from RGB images captured by the front-facing camera. During execution, longitudinal control is handled independently by a PID controller that maintains a fixed speed of 20 km/h (Figure 2 (A)).

**Autonomous Driving Platform.** We use *fortuna* [2], a modified Volkswagen Passat Variant GTE research vehicle equipped with cameras, Velodyne LiDARs, and GNSS/INS localization. The front-facing camera is used both for scenario reconstruction and as input to the ADS, while the roof-mounted LiDAR supports parked-vehicle detection during real-world data collection (Figure 2 (B)).

**Collection Scenario.** Experiments are conducted in Munich on a 450 m two-way residential street with a 30 km/h speed limit. The scenario includes an unmarked roadway, sidewalks, three bends, and parked vehicles partially narrowing the driving lane, providing a realistic yet controlled urban evaluation environment (Figure 2 (C)).

## 3.1 Scenario Reconstruction Quality

We first evaluate whether Cam2Sim reconstructs the target scenario with sufficient geometric fidelity for simulation-based testing. We apply the reconstruction pipeline to the collected recording and generate a CARLA scenario containing the reconstructed road map, replay trajectory, and parked-vehicle placements. We then execute trajectory replay and compare the rendered semantic maps against the real-world semantic observations from the same viewpoints.

We assess reconstruction quality using Intersection over Union (IoU) over the road, car, and background classes. Across 3,144 replayed frames, Cam2Sim achieves a mean IoU (mIoU) of 0.774 ($\sigma$ = 0.071). The reconstructed road layout achieves the highest agreement with the real-world reference, with a mean IoU of 0.847 ($\sigma$ = 0.056), while background regions achieve 0.889 ($\sigma$ = 0.037). Vehicle reconstruction is more challenging due to localization inaccuracies and partial occlusions, yet the car class still achieves a mean IoU of 0.577 ($\sigma$ = 0.206). An example semantic comparison is shown in Figure 2 (D). Overall, these results suggest that Cam2Sim reconstructs urban-driving scenarios with sufficient geometric fidelity to support replay and closed-loop ADS testing.

## 3.2 Behavior Fidelity

Finally, we evaluate whether the GS-augmented simulation improves system-level testing realism. We execute the driving-simulation component in closed-loop mode with the deployed ADS in three domains: the real-world setting, a simulation-only baseline, and a GS-augmented simulation. In the simulation-only baseline, the ADS receives raw CARLA camera images; in the GS-augmented setting, it receives camera observations rendered by the GS model.

To account for variability and the non-determinism of the end-to-end driving model, we execute each domain three times. We compare the generated trajectories against the real-world reference executions using failure and completion rates, as well as trajectory-similarity and steering-smoothness metrics, including Fréchet distance, corridor violations, lateral excess, and steering jitter.

**Table 1: Behavior fidelity comparison between the real-world execution, the simulation-only baseline, and the GS-augmented simulation. Higher completion rates and lower trajectory-deviation metrics indicate behavior closer to the real-world reference. OR = out of road, CC = car crash.**

| Metric | Real | Cam2Sim | Sim |
|---|---|---|---|
| Failure Rate | 0/3 | 0/3 | 3/3 |
| Completion Rate (%) | 100–100–100 | 100–100–100 | 21–23–20 |
| Failure Type (OR–CC) | – | – | 2–1 |
| *Trajectory Similarity* | | | |
| Fréchet Distance (m) | 0.00 | 2.22 | – |
| Corridor Violations (%) | 0.0 | 67.3 | – |
| Mean Excess (m) | 0.00 | 0.330 | – |
| Excess When Out (m) | 0.00 | 0.491 | – |
| *Steering Smoothness* | | | |
| Average Jitter (rad/s) | 1.167 | 0.867 | – |
| Maximum Jitter (rad/s) | 6.452 | 4.983 | – |

Results are shown in Figure 2 (E) and Table 1. The simulation-only baseline consistently fails to complete the scenario, while the GS-augmented simulation successfully completes all runs, matching the real-world executions. Moreover, the GS-based simulation produces trajectories substantially closer to the real-world reference and produces stable steering behavior across all runs. Overall, these preliminary results suggest that GS-rendered observations improve the robustness and behavioral fidelity of closed-loop ADS testing compared to standard simulator rendering.

## 4 Conclusions

We presented Cam2Sim, a tool that transforms real-world recordings or prepared camera datasets into playable CARLA scenarios augmented with Gaussian Splatting. Starting from camera images and poses, the tool reconstructs scenario geometry, prepares simulation-ready assets, trains local GS models, and supports both trajectory replay and ADS closed-loop execution.

As future work, we plan to extend Cam2Sim with dynamic-object reconstruction, support for additional neural-rendering backends and sensor modalities, and larger-scale evaluations across diverse driving environments and ADS architectures.

## 5 Data Availability Statement

Cam2Sim is released as a GitHub repository under the Apache-2.0 license [14]. The artifact includes the tool implementation, the ROS bag used in our validation, the camera-based ADS, example outputs, and scripts for reproducing the GS-rendering and closed-loop evaluations. The repository also documents the required dependencies, including CARLA, ROS, COLMAP, Nerfstudio, and a CUDA-capable GPU. Our validation setup uses an Intel Core Ultra 9 machine with 32 GB RAM and an NVIDIA RTX 4090 GPU.

The review dataset is provided for reproducibility purposes only; an anonymized version with blurred faces and license plates will be publicly released under a CC BY-NC 4.0 license. A public DOI is not released at submission time to avoid premature dissemination; the artifact is available as a GitHub repository for review and will be archived with a DOI upon acceptance.

## References


[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. arXiv:1604.07316 [cs.CV]

[2] Martin Buechel, Malte Schellmann, Holger Rosier, Tobias Kessler, and Alois Knoll. 2019. Fortuna: Presenting the 5G-Connected Automated Vehicle Prototype of the Project PROVIDENTIA.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[4] Alexey Dosovitskiy, Germán Ros, Felipe Codevilla, Antonio López, and Vladlen Koltun. 2017. CARLA: An Open Urban Driving Simulator. *CoRR* abs/1711.03938 (2017).

[5] Overpass-API Drolbr. 2024. Drolbr/overpass-API: A database engine to query the OpenStreetMap data.

[6] Alessio Gambi, Tri Huynh, and Gordon Fraser. 2019. Automatically Reconstructing Car Crashes from Police Reports for Testing Self-Driving Cars. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*.

[7] Yuan Gao, Mattia Piccinini, and Yuchen et al. Zhang. 2026. Foundation Models in Autonomous Driving: A Survey on Scenario Generation and Scenario Analysis. *IEEE Open Journal of Intelligent Transportation Systems* (2026).

[8] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. 42, 4, Article 139 (July 2023), 14 pages.

[9] Stefano Carlo Lambertenghi and Andrea Stocco. 2024. Assessing Quality Metrics for Neural Reality Gap Input Mitigation in Autonomous Driving Testing. In *Proceedings of ICST '24*. IEEE.

[10] Stefano Carlo Lambertenghi, Mirena Flores Valdez, and Andrea Stocco. 2025. A Multi-Modality Evaluation of the Reality Gap in Autonomous Driving Systems. In *40th IEEE/ACM International Conference on Automated Software Engineering, ASE 2025, Seoul, Korea, Republic of, November 16-20, 2025*. IEEE.

[11] Stefano Carlo Lambertenghi, Matthias Weil, and Andrea Stocco. 2026. Real-World Perturbation Testing of Autonomous Driving Systems. In *Proceedings of the 41st IEEE/ACM International Conference on Automated Software Engineering (ASE '26)*.

[12] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *CVPR 2019*.

[13] Hao LU, Tianshuo Xu, Wenzhao Zheng, Yunpeng Zhang, Wei Zhan, Dalong Du, Masayoshi Tomizuka, Kurt Keutzer, and Ying-Cong Chen. 2025. DrivingRecon: Large 4D Gaussian Reconstruction Model For Autonomous Driving.

[14] replication-package 2026. Replication package. https://github.com/ast-fortiss-tum/cam2sim.

[15] Johannes L. Schönberger and Jan-Michael Frahm. 2016. Structure-from-Motion Revisited. In *CVPR 2016*.

[16] Andrea Stocco, Brian Pulfer, and Paolo Tonella. 2023. Mind the Gap! A Study on the Transferability of Virtual Versus Physical-World Testing of Autonomous Driving Systems. *IEEE Transactions on Software Engineering* 49, 04 (apr 2023).

[17] Andrea Stocco and Paolo Tonella. 2021. Confidence-driven Weighted Retraining for Predicting Safety-Critical Failures in Autonomous Driving Systems. *Journal of Software: Evolution and Process* (2021).

[18] Matthew Tancik, Ethan Weber, and Evonne et al. Ng. 2023. Nerfstudio: A Modular Framework for Neural Radiance Field Development. In *ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23)*.

[19] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. 2021. FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*.

[20] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. 2024. SegFormer: simple and efficient design for semantic segmentation with transformers. In *Proceedings of NIPS '21 (NIPS '21)*. Article 924, 14 pages.

[21] Ziyuan Zhong, Yun Tang, Yuan Zhou, Vania de Oliveira Neves, Yang Liu, and Baishakhi Ray. 2021. A Survey on Scenario-Based Testing for Automated Driving Systems in High-Fidelity Simulation. arXiv:2112.00964 [cs.SE]

[22] Huixin Zhu, Zhili Zhang, Junyang Zhao, Hui Duan, Yao Ding, Xiongwu Xiao, and Junsong Yuan. 2024. Scene reconstruction techniques for autonomous driving: a review of 3D Gaussian splatting. *Artificial Intelligence Review* 58 (2024).