CornerSim: A Virtualization Framework to Generate Realistic Corner-Case Scenarios for Autonomous Driving Perception Testing

Alaa Daoud, Corentin Bunel, Maxime Guériau

Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia
Computer Science

www.elsevier.com/locate/procedia

The 15th International Conference on Ambient Systems, Networks and Technologies (ANT)
April 23-25, 2024, Hasselt, Belgium

CornerSim: A Virtualization Framework to Generate Realistic Corner-Case Scenarios for Autonomous Driving Perception Testing

Alaa Daoud*, Corentin Bunel, Maxime Guériau

INSA Rouen Normandie, Univ Rouen Normandie, Université Le Havre Normandie, Normandie Univ, LITIS UR 4108, F-76000 Rouen, France

Abstract

Autonomous driving development requires rigorous testing in real-world scenarios, including adverse weather, unpredictable events, object variations, and sensor limitations. However, these challenging "corner cases" are elusive in conventional datasets due to their unpredictability, high costs, and inherent risks. Recognizing the critical role of ground truth data in autonomous driving, the demand for synthetic data becomes evident. Contemporary machine learning-based algorithms essential to autonomous vehicles heavily depend on labeled data for training and validation. Simulation of scenarios not only mitigates the scarcity of real-world data but also facilitates controlled experimentation in situations that are challenging to replicate physically. The challenge extends beyond data scarcity, encompassing the impediment posed by the inability to systematically control and manipulate specific scenarios, hindering progress.

To overcome these challenges, we present *CornerSim*, a dynamic virtualization framework simplifying the creation and modification of diverse driving scenarios. Leveraging simulation, CornerSim generates synthetic environments for comprehensive testing, providing essential outputs like raw sensor data (cameras, LiDAR, etc.) and labeled data (object detection bounding boxes, classes, semantic segmentation). The unpredictable nature of real-world corner cases complicates obtaining a sufficiently large and diverse annotated dataset. CornerSim addresses this challenge by not only generating synthetic data but also supplying necessary ground truth for training and evaluating machine learning models.

This paper emphasizes the introduction of CornerSim and its ability to challenges related to testing autonomous vehicles in realistic scenarios. It focuses on the framework's capabilities, design principles, and integration, with the goal of enhancing thorough testing and validation of autonomous driving systems in a simulated environment, improving their robustness and safety. Our approach involves running simulations to generate datasets, which are statistically studied and compared with real data. Furthermore, we apply state-of-the-art detection algorithms to assess if data generated by CornerSim is suitable for both training and validation stages.

Keywords: Intelligent Transport Systems (ITS); Autonomous Driving; Simulation; Corner Case; Visual Perception Testing; Dataset Generation;

* Corresponding author. This work is funded by the French national research agency (ANR) under grant reference ANR-21-CE23-0032
E-mail address: alaa.daoud@insa-rouen.fr

1. Introduction

Recent advancements in autonomous driving technology demand thorough testing of vehicles in diverse scenarios. While traditional datasets cover normal driving situations, our focus is on “corner-case” scenarios – uncommon and challenging situations that rarely occur but pose significant challenges or anomalies to the perception system of an autonomous vehicle [1]. These scenarios involve various factors, including environmental factors such as weather or light conditions [28], object variability with unusual characteristics like pedestrians in disguises or partially obstructed objects [27, 26], unpredictable situations requiring effective perception system responses [22], scenarios exploiting sensor limitations such as poor visibility or noise [25], and failure modes related to sensor failures, system malfunctions, or degraded performance [6].

Obtaining data on corner-case scenarios in autonomous driving is technically challenging due to their rarity and unpredictability. Collecting sufficient instances is resource-intensive, while ensuring safety during data collection poses ethical concerns. Annotating and validating such data accurately is complex due to the diverse nature of corner cases, making comprehensive coverage difficult. The limited availability of datasets capturing crucial corner cases is influenced by factors like high costs and potential risks, such as pedestrians in dense traffic or a child running into the street (see Fig. 1b). To address these challenges, we introduce CornerSim, a virtualization framework prototype for creating synthetic environments in autonomous driving. CornerSim, built on the CARLA (CAR Learning to Act) [8] platform, provides a flexible tool for generating diverse driving situations, empowering researchers to explore and create challenging scenarios. This framework sets the stage for future developments, facilitating the generation of realistic corner cases crucial for testing and evaluating autonomous driving algorithms.

The rest of this document is organized as follows: In Section 2, we explore existing literature on synthetic data generation for AV perception systems. Section 3 details the design of CornerSim Virtualization Framework, while Section 4 covers the labeled dataset generation using CornerSim. Finally, in Section 5, we experimentally assess the outputs of the framework in comparison to existing driving datasets.

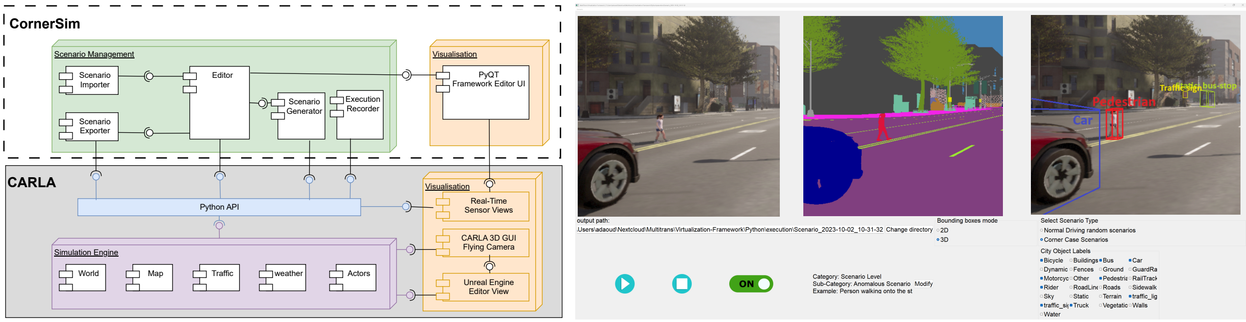
2. Related Work on synthetic data generation for AV perception systems

In recent years, the development of autonomous driving systems has underscored the need to identify and address corner cases. These scenarios can be defined as specific situations or conditions characterized by infrequent, complex, or challenging circumstances that pose significant challenges or anomalies to the perception system of an autonomous vehicle [1]. Exploring literature reveals two key aspects: the vital role of synthetic data in enhancing perception systems ([2], [12]) and the need for targeted synthetic data generation for corner cases. AV driving datasets, like NuScenes [5] (complex JSON) and Foxglove MCAP [10] (compact binary), pose challenges requiring format conversion for system performance and training/testing. Virtualization, as seen in creating simulated environments [17], becomes crucial for extensive testing and training without physical sensors, offering flexibility for diverse scenarios similar to the Waymo Open Dataset [9].

2.1. Corner Cases in Autonomous Driving Perception: A Necessity for Robust Systems

Corner cases, as defined in literature, involve a “non-predictable relevant object/class in a relevant location,” with a particular emphasis on identifying abnormal traffic behavior through a dedicated detection framework [2]. The importance of understanding visual perception in unexpected situations is mentioned by Breitenstein et al. [3], categorizing corner cases based on their deviation from normal traffic behavior and develop a systematization of complexity.

Current options for addressing corner cases include human-in-the-loop testing. Kowol et al. [12] introduce a human-in-the-loop test rig using real-time semantic segmentation in CARLA. Their approach highlights poor recognition in critical scenes and demonstrates the effectiveness of enriching training data with corner cases. Li et al. [14] address the scarcity of datasets for evaluating object detectors on corner cases, introducing the CODA dataset with real-world driving scenes, emphasizing challenges in robust perception. The Lost and Found dataset [20] focuses on road anomaly detection, while Sun et al. [23] propose a decision-making corner case generation method using Markov Decision Process and Deep Reinforcement Learning.



(a) Software architecture

(b) User interface

Fig. 1: CornerSim Virtualization Framework

2.2. Synthetic Data and Corner Case Generation: A Solution to Existing Challenges

To address the challenges in existing perception algorithms and improve robustness, it is essential to generate synthetic data focusing specifically on corner cases. The necessity for specific synthetic data generation techniques targeting corner cases arises from the inherent challenges posed by these non-predictable scenarios in real-world driving. Traditional datasets often lack comprehensive coverage of corner cases due to their rarity, unpredictability, and the associated ethical concerns and risks [13]. To advance the robustness and safety of autonomous driving perception systems, it becomes imperative to intentionally expose these systems to diverse and challenging situations that may deviate significantly from normal traffic behavior. Synthetic data generation techniques offer a controlled and cost-effective means to create a wide range of scenarios [4], allowing algorithms to learn and adapt to these scenarios. By systematically crafting synthetic corner cases, researchers can enhance the diversity of training datasets, ensuring that perception systems are well-equipped to handle the complexities presented by real-world driving environments[11, 18]. This intentional exposure through synthetic data empowers algorithms to recognize and respond effectively to the myriad challenges associated with corner cases, ultimately contributing to the overall reliability and performance of perception systems of autonomous driving [24].

In our work, we introduce CornerSim, a prototype virtualization framework for generating synthetic data focusing on corner cases. Operating at scene, object, and pixel levels, CornerSim allows alterations such as map changes, vehicle relocation, and pixel-level adjustments. To ensure accuracy, it leverages semantic segmentation and object labels from ground-truth data obtained in simulations.

3. CornerSim Virtualization Framework Description

Our virtualization framework, CornerSim, is designed to address the challenges of AV driving datasets, including the need for virtualization and format conversion to achieve a certain level of flexibility and scalability [17]. It is built on top of the CARLA (CAR Learning to Act) simulator [8], which provides a realistic 3D environment for testing autonomous driving algorithms. The framework utilizes the Python API of CARLA to communicate with the simulation server and import and modify driving scenarios. The architecture of the Virtualization Framework consists of two main layers: the scenario management layer and the simulation engine layer. These layers are connected through the Python API, with each component of the scenario management layer acting as a client for the CARLA simulation server. Fig. 1a illustrates the overall architecture of the framework.

Scenario Management Layer: Manages scenarios, imports/exports various formats, and provides tools for editing and generation. We use YAML as the main serialization format for scripting execution scenarios, ensuring compatibility. The Scenario Importer component facilitates importing predefined or externally generated scenarios, leveraging YAML's readability and flexibility.

To enhance understanding of corner cases, we adopt the systematization by Breitenstein et al. [3]. This categorizes corner cases into levels: Scenario, Scene, Object, Domain, and Pixel. Each level has specific categories and examples. The systematization of corner cases is presented in Table 1, including required components for each level. For in-

Fig. 2: Some examples of images of corner cases generated with CornerSim (the bounded image represents an object-level corner case).

stance, in a scenario-level corner case, in addition to the scenario location and environment description, the properties of actors and event timelines are crucial. In a Single Point Anomaly (Object Level), we need to describe attributes of introduced objects. Some examples of corneer cases generated with CornerSim are shown in Fig. 2 including a concrete example of object level corner case (package on the street).

Table 1: Required Components for Describing Corner Cases.

Corner Case Level	Categories	Components of the Corner-case Description file
Scenario Level	- Anomalous Scenario:(e.g. a kid on the street, car accident, ...) - Novel Scenario: (e.g. car coming from side, accessing highway, ...) - Risky Scenario (e.g. short time to collision, overtaking, ...)	{*Location, Actors, Events timeline*} *properties of each actor:(transform, speed, etc.)* *for each event: changes in actors' properties*
Scene Level	- Collective Anomaly: (e.g. traffic jam, Demonstration, ...) - Contextual Anomaly: (e.g. barrier or random objects on street)	*Location, Set of actors* *number and properties of each type of actor*
Object Level	- Single Point Anomaly: (e.g. Animals, stroller, trash, package ...)	*Location, description of (new) object properties*
Domain Level	- Domain shift: (e.g. changes in weather, sign appearance, ...)	*Description of the new background properties*
Pixel Level	- Local outlier: (Dead Pixels, Dirt on the windshield, ...) - Global outlier (Lighting conditions, Overexposure, ...)	*Sensor attributes, Lighting parameters,* *description of noise/dirt artefacts distribution*

The simulation engine layer is responsible for simulating the virtual environment and executing the scenarios. It leverages the capabilities provided by the CARLA simulator The CARLA Python API serves as the communication interface, allowing the scenario management layer to send commands to the simulation engine layer and receive data from it. This client-server architecture enables the Virtualization Framework to leverage the capabilities of the CARLA simulator and extend them with additional functionalities.

The user interface (UI) of the Virtualization Framework provides a flexible and customizable environment for generating and editing scenarios. It allows users to interact with real-time sensor outputs (RGB, semantic/instance segmentation, bounding boxes, etc.), adjust scenario settings, and control scenario execution (see Fig. 1b).

4. Labeled Dataset automatic generation using CornerSim

To automate labeled dataset generation, our approach utilizes real-time data from the CARLA Python API, incorporating steps like matching object positions, camera projection, and semantic segmentation data integration. Simulations run in synchronous mode for synchronization and data retrieval, ensuring simultaneous availability of sensor data for consistent processing. Invisible objects, defined by position information and semantic segmentation, are filtered out based on sensor attributes. The labeling process employs OpenCV[1] to compute bounding box coordinates and assign labels corresponding to object classes similar to the semantics of the Cityscapes Dataset [7].

These labeled datasets, including JSON representations, RGB and semantic segmentation images, serve as ground truth for training perception systems and enable offline analysis for system development. To assess CornerSim's out-

[1] https://opencv.org/

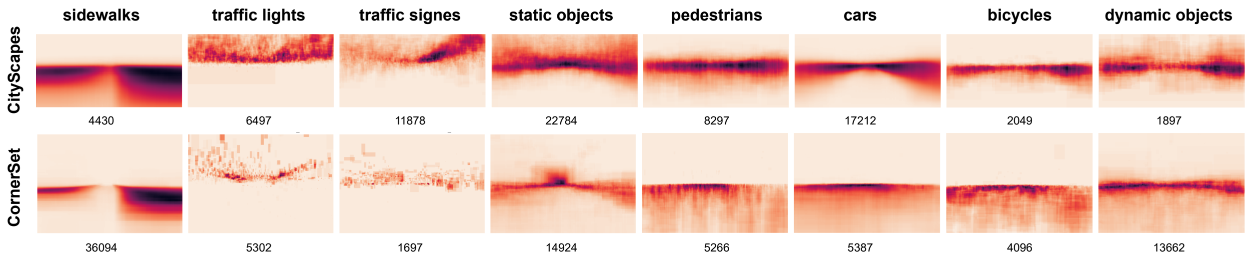


Fig. 3: Examples of population and geometrical distribution variety of different objects in the CornerSet images compared to CityScapes

put, we created an example dataset, 'CornerSet', designed for object detection tasks. CornerSet (publicly available[2]) comprises 2000 high-resolution images, each measuring 1820x1240 pixels at 90 pixels per inch (ppi). It offers diverse scenes, covering countryside, city, highway, and town locations, along with various weather conditions, day and night settings, and a synthetic distribution of actors, vehicles, pedestrians, and objects. Statistical analysis ensures a balanced distribution of objects, averaging 150 per image.

Table 2: Count and Size distribution of Objects in the Dataset compared to CityScape

Object Type	**CornerSet**					**CityScapes**				
	Count		**Size**			**Count**		**Size**		
	Total	Per image	Avg.	Min.	Max.	Total	Per image	Avg.	Min.	Max.
Building	82033	41.28	13.5K	16	1.6M	4872	2.6	361K	36	2.1M
Pole	41016	20.67	18K	16	1,4M	25435	13.6	5K	5	1.3M
Vegetation	29915	18.91	15K	16	1.8M	9658	5.2	128K	32	1.8M
Static	14885	7.71	5K	16	1.2M	22784	12.2	5K	5	0.6M
Dynamic	13649	7.03	3.3K	16	1.5M	1897	2.3	12K	40	1.1M
Rider	6876	3.76	3.1K	16	164K	2485	1.7	9.1K	14	263K
Car	5383	3.07	13K	16	1M	17212	9.5	23K	35	0.7M
Traffic Light	5292	5.36	663	16	21K	6497	6.5	1.7K	12	843K
Pedestrian	5258	3.41	3.3K	16	192K	8297	6	9K	28	450K
Bicycle	4065	2.39	3.4K	16	163K	2049	2.3	10K	16	289K
Motorcycle	2829	1.89	4K	16	208K	459	1.5	17K	18	223K
Traffic Sign	1673	3.24	650	16	21K	11878	6.7	2K	12	492K
Truck	1199	1.64	27K	16	1.2M	351	1.3	42K	132	0.9M
Bus	164	1.40	61K	16	1.6M	203	1.2	54K	120	0.7M

In Figure 3, we showcase the spatial distribution of specific objects within our synthetic dataset compared to CityScapes, leveraging heatmaps in which each pixel's darkness directly correlates with the frequency of the corresponding label, offering a quantitative representation of object distribution. These heatmaps exemplify the diverse and nuanced geometrical distribution of various object classes, offering a visual representation of the rich complexity within CornerSet. When comparing these heatmaps with those derived from the CityScapes dataset, differences in object population density become apparent. However, a more profound analysis reveals striking similarities in the geometric position and spread of high-density areas for each object type across both datasets. Despite variations in density, CornerSet successfully mirrors realistic scenarios, attesting to CornerSim's prowess in generating authentic and intricate scenes. Notably, areas with rare instances of specific objects, such as the region above the horizon line (for cars, pedestrians and bikes) or below it (for traffic lights and signs), align with real-world expectations, underscoring the spatial coherence captured by our virtualization framework. In Table 2, a comprehensive statistical analysis of both CornerSet, and the CityScapes datasets. CornerSet stands out with a wider range of sizes, while the total counts vary from class to another between the two datasets. Particularly noteworthy are specific objects like "Building" and

[2] https://drive.google.com/drive/folders/12CmNRJ_EP0XqqChPjIXEw-kOhsZlKUxM?usp=sharing

"Vegetation" which exhibit significantly greater counts and smaller average sizes in CornerSet. This disparity may arise from the manual labeling process in real-world images, potentially grouping multiple instances into a single polygon due to difficulty in differentiating borders. Conversely, labeling in CornerSim, relies on ground truth identifiers of simulation objects, accurately delineates distinct instances, even if they share class and appearance similarities. Despite these variations, the comparison underscores CornerSim's versatility in generating diverse, densely populated synthetic scenes, confirming its abilities for simulating realistic scenarios across diverse applications.

5. Experiments on object detection algorithms in a generated object-level corner-case scenarios

To be able to thoroughly assess the usability and utility of CornerSim, we have chosen object-level corner-cases for experimentation. This scenario challenges autonomous driving perception systems that rely on Object Detection algorithms [21, 16]. The following experiments, using state-of-the-art object detection models on the dataset (CornerSet) generated by CornerSim, illustrate how our framework can be useful for developing and testing corner-case scenarios for AV systems.

5.1. Setup

We use one model from each type of object detection architecture. RetinaNet is a one-stage detector, while Faster R-CNN is a two-stage detector; both of them have state-of-the-art results. We use the PyTorch implementation of those two networks and the associated weights that are pre-trained on the COCO dataset [15]. This makes our experiments easily reproducible and ensures that the comparison between models is fair. All the documentation and the corresponding weights are available online[3]. For testing and training, we use an NVIDIA RTX A2000 12GB, keeping the default parameters from PyTorch. The external dataset used for pre-training is COCO, a large and diverse object detection dataset frequently employed. The metrics for evaluation include mean average precision (mAP), precision and recall [19].

Table 3: Detection results on COCO val set 2017 and CornerSet with Faster-RCNN and RetinaNet trained on COCO.

Model		**COCO**			**CODA**			**CornerSet**	
	mAP	Precision	Recall	mAP	Precision	Recall	mAP	Precision	Recall
Evaluation on COCO classes									
Faster R-CNN	0.43	0.73	0.56	(0.05)	(0.25)	0.45	0.27	0.54	0.47
RetinaNet	0.33	0.49	0.82	(0.04)	(0.41)	0.34	0.18	0.60	0.27
Faster R-CNN finetune on CornerSet	0.26	0.61	0.48	(0.04)	(0.20)	0.39	0.42	0.75	0.66
RetinaNet finetune on CornerSet	0.24	0.85	0.36	(0.03)	(0.40)	0.32	0.26	0.83	0.34
Evaluation on Novel classes									
Faster R-CNN	-	-	-	(0.00)	(0.02)	0.06	0.01	0.20	0.08
RetinaNet	-	-	-	(0.00)	(0.03)	0.15	0.01	0.09	0.16
Faster R-CNN finetune on CornerSet	-	-	-	(0.00)	(0.01)	0.04	0.01	0.09	0.02
RetinaNet finetune on CornerSet	-	-	-	(0.00)	(0.02)	0.16	0.01	0.03	0.08

5.2. Results

We chose to illustrate our work on object-level, as it represents one of the few corner cases where the literature already has a specialized dataset with real-world images for comparison: CODA [13]. One significant challenge with this dataset is the lack of extensive annotations in the images, there are objects without labels, making it

[3] Pytorch documentation with all the models weights, implementations and examples used is available online : https://pytorch.org/vision/stable/models.html#object-detection

impossible to calculate all metrics accurately. In contrast, our simulation provides annotated objects in every image, allowing us to evaluate mAP and Precision, in addition to Recall. Our experiments results include a computation of these three metrics on the COCO classes and on the novel classes, that represent the corner-case objects we aim at detecting additionally to the classes known by the model. Experiment results are reported in Table 3. Similarly to the CODA benchmark [13], object detectors are not suited to detect novel classes, neither on CODA or CornerSet. The low values in precision and mAP for CODA can be attributed to the lack of ground truth annotation. In contrast, CornerSet enable the possibility of evaluating correct precision and mAP on both known and novel classes, thanks to fully annotated ground truth. One drawback of evaluation on synthetic data is that we can't really ensure that the performance will be the same for real data. Having only a 15 percent drop in mAP on CornerSet for models trained on real data shows that simulated objects are still close enough to real data to be recognized by the model. Thus, data generated from CornerSim may not be as accurate as real data, but experiments confirm it could be a first way to correctly evaluate models on corner-cases without costly manual labeling of data.

In addition to demonstrating that CornerSet can be used for evaluation purposes, we present results of training by fine-tuning both models on our dataset. We use only labels from the same classes as those availabe in COCO dataset to simulate conditions where we have only common class labels, similar to the CODA benchmark [13]. An improvement of 10 percent in mAP for both models confirms the potential of using our dataset and CornerSim for training and fine-tuning.

6. Conclusion and future work

This paper presents CornerSim, a user-friendly framework tackling the lack of data on corner-case scenarios in autonomous driving. We outlined its architecture and core components, highlighting its potential as a valuable tool for researchers to explore diverse scenarios, aiding in the development of autonomous vehicles for navigating complex real-world situations.

Our experiments, comparing a small synthetic dataset generated by CornerSim to COCO and CODA, demonstrate its efficacy in evaluating AV systems, particularly in Object-level corner cases. While acknowledging that relying solely on simulated data is not sufficient, this work serves as an initial step and offers a viable alternative to the high cost of annotating real data for corner cases.

Although the experiments in this paper use a limited dataset and focus on a single example among various corner cases, preliminary results suggest that CornerSim enables the configurable generation of large amounts of annotated data. Future directions will involve creating diverse corner case scenarios with various configurations, paving the way for a more comprehensive evaluation of autonomous vehicle systems using simulated corner-case datasets.

References

[1] Daniel Bogdoll, Jasmin Breitenstein, Florian Heidecker, Maarten Bieshaar, Bernhard Sick, Tim Fingscheidt, and Marius Zöllner. Description of corner cases in automated driving: Goals and challenges. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1023–1028, 2021.

[2] Jan-Aike Bolte, Andreas Bar, Daniel Lipinski, and Tim Fingscheidt. Towards corner case detection for autonomous driving. In *2019 IEEE Intelligent vehicles symposium (IV)*, pages 438–445. IEEE, 2019.

[3] Jasmin Breitenstein, Jan-Aike Termöhlen, Daniel Lipinski, and Tim Fingscheidt. Corner cases for visual perception in automated driving: some guidance on detection approaches. *arXiv preprint arXiv:2102.05897*, 2021.

[4] Vishnupriya Buggineni. Utilizing synthetic data generation techniques to improve the availability of data in discrete manufacturing for ai applications: A review and framework. *Master thesis*, 2023.

[5] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[6] Andrea Ceccarelli and Francesco Secci. Rgb cameras failures and their effects in autonomous driving applications. *IEEE Transactions on Dependable and Secure Computing*, 2022.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[8] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator, 2017.

[9] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R. Qi, Yin Zhou, Zoey Yang, Aurélien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9710–9719, October 2021.

[10] foxglove. Mcap: a modular, performant, and serialization-agnostic container file format, useful for pub/sub and robotics applications. https://mcap.dev/.

[11] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[12] Kamil Kowol, Stefan Bracke, and Hanno Gottschalk. A-eye: Driving with the eyes of ai for corner case generation. *arXiv preprint arXiv:2202.10803*, 2022.

[13] Kaican Li, Kai Chen, Haoyu Wang, Lanqing Hong, Chaoqiang Ye, Jianhua Han, Yukuai Chen, Wei Zhang, Chunjing Xu, Dit-Yan Yeung, Xiaodan Liang, Zhenguo Li, and Hang Xu. CODA: A Real-World Road Corner Case Dataset for Object Detection in Autonomous Driving, March 2022. Comment: ECCV 2022.

[14] W. Li, C. W. Pan, R. Zhang, J. P. Ren, Y. X. Ma, J. Fang, F. L. Yan, Q. C. Geng, X. Y. Huang, H. J. Gong, W. W. Xu, G. P. Wang, D. Manocha, and R. G. Yang. AADS: Augmented autonomous driving simulation using data-driven algorithms. *Science Robotics*, 4(28), mar 2019.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common Objects in Context, February 2015. Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list.

[16] Zeyu Ma, Yang Yang, Guoqing Wang, Xing Xu, Heng Tao Shen, and Mingxing Zhang. Rethinking Open-World Object Detection in Autonomous Driving Scenarios. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 1279–1288, Lisboa Portugal, October 2022. ACM.

[17] Matthias Mueller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. In *Conference on Robot Learning*, pages 1–15. PMLR, 2018.

[18] Yunhao Nie, Bo Lu, Qiyuan Chen, Qinghai Miao, and Yisheng Lv. Synposes: Generating virtual dataset for pedestrian detection in corner cases. *IEEE Journal of Radio Frequency Identification*, 6:801–804, 2022.

[19] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A Survey on Performance Metrics for Object-Detection Algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, July 2020.

[20] Peter Pinggera, Sebastian Ramos, Stefan Gehrig, Uwe Franke, Carsten Rother, and Rudolf Mester. Lost and found: detecting small road hazards for self-driving vehicles. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1099–1106. IEEE, 2016.

[21] Deepak Kumar Singh, Shyam Nandan Rai, KJ Joseph, Rohit Saluja, Vineeth N Balasubramanian, Chetan Arora, Anbumani Subramanian, and CV Jawahar. Order: Open world object detection on road scenes. In *Proc. Adv. Neural Inf. Process. Syst. Workshops*, pages 1–12, 2021.

[22] Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering*, pages 359–371, 2020.

[23] Haowei Sun, Shuo Feng, Xintao Yan, and Henry X Liu. Corner case generation and analysis for safety assessment of autonomous vehicles. *Transportation research record*, 2675(11):587–600, 2021.

[24] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 969–977, 2018.

[25] Jorge Vargas, Suleiman Alsweiss, Onur Toker, Rahul Razdan, and Joshua Santos. An overview of autonomous vehicles sensors and their vulnerability to weather conditions. *Sensors*, 21(16):5397, 2021.

[26] Zhenpei Yang, Jeffrey Z. Pan, Linjie Luo, Xiaowei Zhou, Kristen Grauman, and Qixing Huang. Extreme relative pose estimation for rgb-d scans via scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[27] Amir R Zamir, Tilman Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and Silvio Savarese. Generic 3d representation via pose estimation and matching. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 535–553. Springer, 2016.

[28] Yuxiao Zhang, Alexander Carballo, Hanting Yang, and Kazuya Takeda. Perception and sensing for autonomous vehicles under adverse weather conditions: A survey. *ISPRS Journal of Photogrammetry and Remote Sensing*, 196:146–177, 2023.