engineering proceedings

MDPI

Proceeding Paper

Comparison of CARLA and aiSim in Testing Object Detection Neural Networks †

Bence Török and János Hollósi *

Vehicle Industry Research Center, Széchenyi István University, 9026 Győr, Hungary; torok.bence@ga.sze.hu
* Correspondence: hollosi.janos@sze.hu
† Presented at the Sustainable Mobility and Transportation Symposium 2026, Győr, Hungary, 8 October 2026.

Abstract

Synthetic data generated by simulation environments have gained importance in the evaluation of machine learning models, particularly in domains where collecting real-world data is costly or difficult to scale. Photorealistic and well-annotated synthetic datasets enable the generation of large volumes of data under controlled and repeatable conditions. The aim of this work is to compare the CARLA and aiSim simulation platforms in terms of their suitability for testing and evaluating object detection neural networks. Datasets were generated in both simulators, each accompanied by automatically produced ground truth annotations. Pre-trained object detection models are applied to image sequences obtained from both environments, and the results are compared against the corresponding ground truth data. The evaluation focuses on detection accuracy, error characteristics, and the extent to which each simulator's visual and annotation quality supports realistic model assessment. Based on the findings, conclusions are drawn regarding the strengths and limitations of each platform.

Keywords: autonomous vehicle; object detection; simulation; YOLO

1. Introduction

The rapid advancement of deep learning-based object detection has created an immense demand and need for large-scale, accurately annotated datasets. In safety-critical domains such as autonomous driving, collecting such data in the real world is time-consuming, expensive, and limited in its coverage of rare or hazardous scenarios. Simulation environments offer a compelling alternative, generating virtually unlimited pixel-accurate ground truth under controlled, repeatable conditions, which makes them well-suited for benchmarking computer vision algorithms.

Among available platforms, CARLA (0.9.15, Computer Vision Center, Barcelona, Spain) has become the de facto open-source standard for autonomous driving research, while commercial platforms such as aiSim (5.7.0, aiMotive, Budapest, Hungary) primarily emphasize photorealistic rendering and industrial deployment. Systematic comparisons of how these two environments perform as evaluation platforms for object detection remain scarce. This work addresses that gap by constructing parallel pipelines for both simulators, generating datasets under highly similar conditions, and evaluating pre-trained You Only Look Once (YOLO, Ultralytics, Frederick, MD, USA) detectors against automatically produced ground truth using Precision, Recall, and F1-score across multiple traffic actor categories such as pedestrians, cars, and trucks.

Check for updates

Academic Editors: Gábor Dogossy, Szabolcs Kocsis-Szürke, Dániel Csikor, Dániel Serfőző, Miklós Kuczmann, András Lajos Nagy, László Lendvai and Ambrus Zelei

Published: 8 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

The remainder of this paper covers Related Work (Section 2), Materials and Methods (Section 3), Results (Section 4), and Conclusions and Future Work (Section 5).

2. Related Work

2.1. Synthetic Data in Deep Learning for Perception

The use of synthetically generated data to train and evaluate machine learning models, like the YOLO models, has a long history in computer vision. Early work demonstrated that photorealistic rendering engines could serve as an effective data source for training convolutional neural networks on tasks such as optical flow estimation [1]. The key motivation is the domain of synthetic data generation: labels that would require laborious manual annotation in real footage (such as per-pixel semantic class, instance identity, or precise 3D bounding box parameters) can be obtained automatically and with perfect accuracy from a renderer's internal scene graph.

The domain gap between synthetic and real data has been an active research topic. Techniques including domain randomization [2] and adversarial domain adaptation [3] have all been proposed to reduce the divergence between the statistics of synthetic and real-world image distributions. More recent findings suggest that when simulation realism is sufficiently high, models trained or evaluated entirely on synthetic data can generalize well to real-world scenarios [4], motivating the use of simulators not only for training augmentation but also for standalone evaluation.

2.2. CARLA Simulation Platform

CARLA was introduced by Dosovitskiy et al. [5] as an open-source simulation environment designed specifically for autonomous driving research. Built on Unreal Engine, it provides a modular sensor suite including RGB cameras, LiDAR, radar, depth cameras, and semantic and instance segmentation sensors. The platform includes several pre-built urban maps of varying complexity, a configurable traffic manager for spawning and controlling NPC vehicles and pedestrians, and a Python (3.10, Python Software Foundation, Beaverton, OR, USA) API for controlling the scene and extracting data.

CARLA has been widely used in the research community for tasks ranging from driving policy synthesis to the generation of object detection training datasets [6]. Recent studies have exploited CARLA's instance segmentation sensor to obtain precise pixel-level bounding boxes for traffic participants [7], a methodology closely related to the approach adopted in this work. However, few studies have systematically quantified how well detectors, such as the YOLO models, perform on synthetic camera footage generated by CARLA.

2.3. aiSim Simulation Platform

aiSim is a modular, high-fidelity virtual simulation environment developed by aiMotive for the verification and validation of automotive perception systems under functional safety requirements. It was the world's first automotive simulator to be certified according to ISO 26262 [8] for use in development of systems up to ASIL-D (the highest automotive safety integrity level defined by the standard), which means that it can be used as a qualified tool in the development and verification of safety-critical systems such as braking and autonomous driving systems. As a result, aiSim can be integrated into the safety-relevant development workflows of automotive manufacturers.

The platform is available as a standalone application and as a modular SDK that allows custom client applications and sensor plugins to be integrated directly into the simulation environment. Its rendering engine, aiSim AIR, uses AI-based rendering to enable high-fidelity, multi-sensor simulation across camera, LiDAR, and radar modalities [9,10].

Despite its growing adoption in the automotive industry, there aren't many comparative studies evaluating aiSim against open-source alternatives such as CARLA in the context of object detection benchmarking. This work aims to fill that gap.

2.4. YOLO-Family Object Detectors

The You Only Look Once (YOLO) family of object detectors [11] has become a widely adopted baseline of real-time detection research, achieving strong accuracy–latency trade-offs that make it widely applicable in embedded and automotive contexts. Since the original introduction by Redmon et al. [11], the architecture has undergone numerous revisions. YOLOv8 [12], developed by Ultralytics, introduced a redesigned anchor-free head and improved training pipeline. YOLOv9 [13] incorporated Programmable Gradient Information (PGI) to address information bottleneck issues in deep networks. This helps to mitigate information bottlenecks and improves gradient propagation. YOLOv10 [14] further refined the detection head with NMS-free dual label assignment. YOLOv11 [15] and the recently released YOLOv12 [16] continued this trajectory with architectural improvements aimed at improving accuracy and inference efficiency.

In the context of this work, evaluating multiple YOLO generations against simulator-generated ground truth provides insight into whether performance differences between simulators are consistent across detector versions or whether they interact with specific architectural characteristics of the detectors.

2.5. Evaluation Metrics for Object Detection

Standard evaluation of object detection systems relies on the precision–recall framework, which describes the balance between the share of correct detections among all predicted boxes (Precision) and the share of ground truth objects that are detected correctly (Recall). The F1-score is the harmonic mean of these two measures. Intersection over Union (IoU) is used as the matching criterion to decide whether a predicted bounding box should be counted as a true positive relative to a ground truth annotation, with a threshold of 0.5 representing the standard evaluation setting [17].

Mean Average Precision (mAP) summarizes precision–recall curves over all classes and, in its COCO variant [18], across multiple IoU thresholds from 0.5 to 0.95. Although the present study focuses on per-image Precision, Recall, and F1 at a fixed IoU threshold consistent with standard detection benchmarks, these metrics still provide a direct and interpretable view of detector behaviour on simulator-generated data.

3. Materials and Methods

This section outlines the dataset generation pipelines used for both simulators, together with the evaluation framework applied to assess object detection model performance. The overall workflow is conceptually as follows: for each simulator, an autonomous driving scenario is rendered, sensor data are collected, automatically generated ground truth bounding box annotations are collected alongside, and the resulting image-label pairs are placed into different folders and then used to evaluate pre-trained YOLO detectors. The evaluation script derives detection statistics by matching predicted boxes to ground truth boxes using an IoU threshold.

3.1. CARLA Dataset Generation Pipeline

Dataset generation for CARLA was implemented as a modular Python pipeline consisting of five components: a central configuration module, a scene and world utilities module, a traffic spawning module, a frame capture and annotation module, and a batch automation script. The pipeline was developed and run with Python 3.10. The pipeline produces YOLO-format annotation files alongside RGB images automatically, without

any manual labelling, and supports full reproducibility through explicit random seed management. With this, we can make pixel-perfect bounding box annotations on the camera images.

3.1.1. Configuration and Scene Setup

All simulation parameters are defined in a data class hierarchy in the config file called carla_config.py. This makes the configuration structure easy to understand and modify. The file contains three nested classes (WeatherConfig, TrafficConfig, and CaptureConfig) within a top-level SimulationConfig object. This covers weather conditions, traffic density and filtering, image resolution, bounding box quality thresholds, and output paths. This file serves as a control panel for configuring the simulations.

Each run of the dataset is started through run_dataset.py, which can take command-line arguments for the target map, frame count, vehicle and walker counts, and random seeds. It then connects to the running CARLA server and loads the selected map. Output folders are named after the map, for example, dataset_Town03/, so data from separate runs do not overwrite one another. The simulation runs in synchronous mode at a fixed 20 Hz timestep, which allows deterministic, frame-accurate capture.

A preprocessing step handles a CARLA-specific artifact. Some maps contain baked-in static vehicles, such as parked cars, motorcycles, and bicycles, which are part of the map geometry and are not exposed through the actor API. These are identified via the environment object registry, hidden from the renderer, and then replaced with equivalent spawned actors at the same locations. After replacement, physics is enabled briefly so the vehicles can settle naturally onto the road surface and then disabled again to prevent two-wheeled vehicles from falling over. This way, all vehicles in the scene, including parked ones, remain visible to the annotation pipeline.

3.1.2. Traffic, Ego Vehicle, and Sensor Setup

Dynamic traffic is spawned via CARLA's Traffic Manager with a 2.5 m minimum following distance. Vehicles and pedestrians are selected from filtered blueprint libraries using a seeded random state, ensuring reproducibility across runs. The ego vehicle—a Tesla Model 3—is spawned at a separately seeded spawn point and placed in autopilot mode. Three cameras are attached to it at a forward-facing position (x = 1.5 m, z = 1.2 m): an RGB camera, a depth camera, and an instance segmentation camera, all sharing the same 1920 $\times$ 1080 resolution and 90° field of view.

3.1.3. Annotation Generation

For each captured frame, the annotation pipeline decodes the depth image into a metric distance map and the instance segmentation image into a semantic class map and a 16-bit instance index map. For each actor in the scene, the pipeline projects its 3D bounding box onto the image plane and uses a depth-validated histogram vote over the instance segmentation pixels to identify which renderer instance index corresponds to that actor. A tight, pixel-perfect bounding box is then extracted from the confirmed instance mask. Two quality filters are applied: a minimum of 12 visible pixels and a minimum fill ratio of 0.25, which rejects heavily occluded objects. For two-wheeled vehicles, separate bounding boxes are produced for the vehicle frame and the rider. All annotations are written in YOLO format (normalized centre coordinates and dimensions), one label file per frame.

3.1.4. Multi-Map Batch Generation

batch_generate.py automates sequential capture across six maps (Town01, Town02, Town03, Town04, Town05, and Town10HD) by invoking the per-map pipeline as a subprocess with consistent parameters. A 5 s cooldown between runs allows the server to

release memory. Failed runs are logged and skipped, enabling unattended overnight dataset generation.

3.2. aiSim Dataset Generation Pipeline

aiSim is a simulation environment designed for generating synthetic data for computer vision and robotics applications. It provides configurable virtual scenes in which objects, agents, and environmental conditions can be controlled, making it suitable for perception experiments and dataset generation.

In this work, aiSim was used to generate image sequences together with frame-level ground truth annotations for object detection. The simulator exported annotation data in JSON format for each frame, including 2D and 3D bounding box coordinates, object class information, visibility-related fields such as visible and total pixel counts, and occlusion values. These annotations were then processed with a custom Python script that converted the JSON files into YOLO-compatible label files. During this conversion, the bounding boxes were transformed into the standard YOLO format consisting of class ID, normalized centre coordinates, width, and height. For class indexing, a COCO-based label mapping was used. The converted annotations were stored in separate label folders alongside the corresponding rendered images, forming complete image-label datasets that could be used directly for evaluating YOLO-based object detectors.

3.3. Object Detection Model and Evaluation

Object detection is performed using five generations of the Ultralytics YOLO family (YOLOv8m, YOLOv9m, YOLOv10m, YOLOv11m, and YOLOv12m), all used as pre-trained models without any fine-tuning on simulator data. The medium-size variant was selected across all generations to ensure a consistent accuracy-to-speed trade-off. Evaluating multiple generations allows the comparison between simulators to be validated as architecture-independent rather than an artifact of a single model.

3.3.1. Evaluation Script

A custom evaluation script goes through the dataset image by image, loading the corresponding YOLO-format ground truth label and converting normalized bounding box coordinates to pixel coordinates. It then queries the model for predictions using a configurable confidence threshold (default: 0.25). Before matching, the predictions are sorted in descending order of confidence, in line with standard object detection evaluation practice, so that when multiple detections overlap the same ground truth box, the most confident one is considered first.

Predicted boxes are matched against ground truth boxes of the same class using a greedy strategy: for each prediction, the unmatched ground truth box with the highest Intersection over Union (IoU) is identified. If this IoU meets or exceeds the configured threshold (default: 0.50) and the ground truth box has not yet been claimed, the prediction is recorded as a True Positive (TP); otherwise, it is recorded as a False Positive (FP). Ground truth boxes that remain unmatched after all predictions are processed constitute False Negatives (FN). A class-filter option allows evaluation to be restricted to specific COCO classes, enabling targeted per-category analysis.

Aggregate and per-class Precision, Recall, and F1-score are computed at the end of the run:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{2}$$

https://doi.org/10.3390/engproc2026157018

$$\mathrm{F1} = \frac{2 \times \mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{3}$$

Because Precision, Recall, and F1-score describe the model at a single operating point, Average Precision (AP) was also computed. Inference is run at a very low confidence threshold so that all detections are retained, and each prediction is recorded with its confidence and whether it matched a ground truth box; sorting these by descending confidence yields a precision–recall curve per class, whose area is the AP (interpolated onto a 101-point recall grid, following the COCO convention). AP@0.5 uses an IoU threshold of 0.5, AP@0.5:0.95 averages over IoU thresholds from 0.5 to 0.95, and mAP is the mean of the per-class AP values. These quantities are independent of the operating point at which the model is deployed.

3.3.2. Evaluation of Outputs

The evaluation produces three outputs per run. A per-frame CSV records the TP, FP, FN, Precision, and Recall for each image; aggregate metrics are obtained by summing the raw TP, FP, and GT counts before computing the rates, which avoids the bias of averaging per-image percentages over frames with different object counts. A summary CSV holds the per-class and aggregate AP values and the precision–recall curves behind Figure 1, all from the same inference pass. Finally, a color-coded visualization is saved for each frame, with true positives in green, false positives in red, and ground truth in blue, so failure modes can be inspected directly.

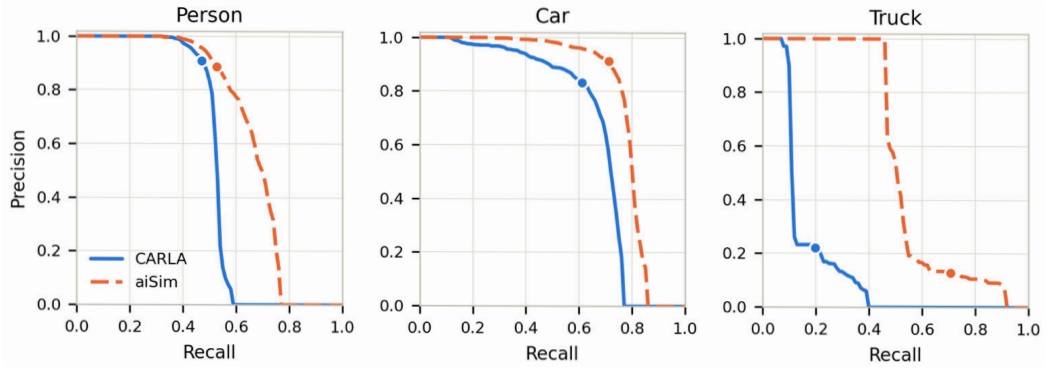


Figure 1. Precision–recall curves for the three shared object classes, evaluated with YOLOv12m. The marker on each curve indicates the operating point at a confidence threshold of 0.25.

4. Results

All results in this section were made using the same evaluation process and settings for both simulators. The confidence threshold was set to 0.25, and the IoU matching threshold to 0.50. Each simulator was evaluated using two independent simulation runs, with 1000 frames for CARLA and 1369 frames for aiSim. Because the evaluated YOLO models were applied in their pre-trained form without any fine-tuning on synthetic data, the reported metrics reflect the transfer performance of off-the-shelf detectors rather than absolute detection capability, and the comparison between simulators is the primary quantity of interest. Figure 2 illustrates a representative frame from each generated dataset, where ground truth annotations are shown in blue, true positive detections in green, and false detections in red.



(**a**) (**b**)

Figure 2. (**a**) A single frame from the dataset made with aiSim; (**b**) a single frame from the dataset made with CARLA.

Table 1 presents the overall detection performance of five YOLO generations (v8m through v12m) on both simulators, with the difference computed as aiSim minus CARLA. A consistent pattern holds across all five models: CARLA yields higher precision (averaging 81.33% versus 74.29%), while aiSim yields substantially higher recall (averaging 65.29% versus 56.25%). Because the recall advantage outweighs the precision disadvantage, the aggregate F1-score is higher on aiSim for every model, with ΔF1 averaging +0.030. The two simulators thus induce opposite error profiles: CARLA makes the detectors conservative, producing fewer but more reliable detections, while aiSim lets them recover more of the ground truth at the cost of slightly more false positives.

Table 1. Overall model comparison.

Model	CARLA				aiSim				Difference			
	Prec.	**Rec.**	**F1**	**mAP@0.5**	**Prec.**	**Rec.**	**F1**	**mAP@0.5**	**ΔPrec.**	**ΔRec.**	**ΔF1**	**ΔmAP@0.5**
YOLOv8m	78.22%	57.52%	0.6629	0.4927	71.93%	65.90%	0.6878	0.7179	−6.29%	8.38%	0.0249	0.2252
YOLOv9m	80.62%	57.31%	0.6700	0.4909	72.73%	62.93%	0.6748	0.6245	−7.89%	5.62%	0.0048	0.1336
YOLOv10m	83.93%	55.83%	0.6706	0.5571	75.21%	68.07%	0.7146	0.6845	−8.72%	12.24%	0.0441	0.1274
YOLOv11m	84.79%	56.55%	0.6785	0.4647	74.15%	64.39%	0.6893	0.6680	−10.6%	7.84%	0.0108	0.2033
YOLOv12m	79.08%	54.02%	0.6419	0.4502	77.44%	65.14%	0.7076	0.6657	−1.64%	11.12%	0.0657	0.2155

The threshold-independent metrics separate the platforms far more sharply. mAP@0.5 is higher on aiSim for every model, averaging 0.672 against 0.491 for CARLA (+0.181), and mAP@0.5:0.95 averages 0.533 against 0.333 (+0.199), again positive for all five models. This is roughly six times the magnitude of the corresponding F1 difference, indicating that the modest F1 advantage at a confidence threshold of 0.25 understates how differently the two datasets behave across the full operating range.

Table 2 reports the per-class breakdown for the representative YOLOv12m model, restricted to the three classes present in both datasets (person, car, and truck) so that the comparison reflects detection quality rather than scene composition. aiSim achieves a higher F1-score for all three classes, most clearly for cars (0.71 to 0.80; ΔF1 = +0.09) and for persons (ΔF1 = +0.04), driven largely by higher recall. At the aggregate level, the same trade-off as in Table 1 appears, with CARLA marginally ahead on precision and aiSim well ahead on recall, giving an overall F1 of 0.71 against 0.64.

https://doi.org/10.3390/engproc2026157018

Table 2. Per-class comparison (YOLOv12m).

Class	GT	CARLA				GT	aiSim				Difference	
		Prec.	Rec.	F1	AP@0.5		Prec.	Rec.	F1	AP@0.5	ΔF1	ΔAP@0.5
Person	1394	90.86%	47.06%	0.62	0.521	1412	88.77%	52.62%	0.66	0.666	0.04	0.145
Car	2994	83.20%	61.19%	0.71	0.672	2853	91.23%	71.12%	0.80	0.782	0.09	0.110
Truck	344	22.08%	19.77%	0.21	0.158	113	12.86%	70.80%	0.22	0.549	0.01	0.391
Overall	4732	79.08%	54.02%	0.64	0.450	4378	77.44%	65.14%	0.71	0.666	0.07	0.216

The AP@0.5 column changes the reading considerably. For persons and cars, it confirms the F1 ordering while widening the margin (+0.145 and +0.110), but for trucks it reverses it entirely. At a threshold of 0.25, the truck class appears almost equally poor on both platforms (F1 of 0.21 and 0.22), yet the AP@0.5 values are 0.158 and 0.549—the largest class-level gap in the study. The reason is that aiSim's truck detections are well ranked but poorly calibrated: the most confident predictions are almost all correct, and only the low-confidence tail is unreliable. A fixed threshold admits that tail and collapses the precision, which the F1 reports as failure, whereas the AP correctly credits the ranking. On CARLA, no such reserve exists, and precision collapses almost immediately.

Figure 1 shows the precision–recall curves behind these AP values, with the operating point at a confidence threshold of 0.25 marked on each. For all three classes, the aiSim curve lies above the CARLA curve across essentially the whole recall range, confirming that the advantage is not an artifact of the chosen threshold. The truck panel also makes the calibration effect visible: the aiSim curve holds near-perfect precision out to a recall of about 0.45, whereas the CARLA curve drops below 0.25 precision after a recall of only about 0.1.

5. Conclusions and Future Work

This study compared the CARLA and aiSim simulation platforms as evaluation environments for pre-trained object detection networks, using automatically generated ground truth annotations and a consistent evaluation protocol across five YOLO generations. The results in Table 1 demonstrate that the two simulators produce systematically different and complementary error characteristics rather than a simple ranking of one being uniformly superior. CARLA elicited higher precision but markedly lower recall, while aiSim elicited higher recall and a higher overall F1-score for every evaluated model. The threshold-independent metrics point the same way but far more decisively: aiSim achieved a higher mAP@0.5 and mAP@0.5:0.95 for all five models, by an average margin of +0.181 and +0.199 respectively, and the precision–recall curves in Figure 1 confirm that the advantage holds across the operating range rather than at one threshold. This pattern was robust across all five YOLO versions, indicating that the observed differences arise from properties of the simulated imagery and its annotations rather than from any single detector architecture.

The per-class analysis in Table 2 reinforced this conclusion: aiSim yielded higher F1-scores and higher AP@0.5 values for all three shared classes, most notably for cars. The truck class proved the most instructive case, as its near-identical F1-scores concealed an AP@0.5 difference of +0.391, showing that a single operating point can misrepresent detector behaviour severely when confidence calibration differs between datasets. The consistently weak truck precision on CARLA nevertheless suggests that its truck annotations may not align cleanly with the COCO truck category, a property of the annotation process rather than the imagery. Overall, aiSim provided the more favourable evaluation conditions for off-the-shelf detectors, while CARLA remains a valuable and accessible platform whose behaviour may be preferable when precision is prioritised.

https://doi.org/10.3390/engproc2026157018

Several directions follow from the limitations of this study. The most important is the construction of identical scenarios in both simulators with matched traffic density and camera placement so that performance differences can be attributed unambiguously to rendering and annotation quality. A related step is a direct audit of the class assignment produced by each annotation pipeline, particularly for the vehicle categories, since the truck results indicate that some of the observed differences may originate in labelling rather than in image quality. The evaluation should also be extended across a wider range of environmental conditions, including varying lighting levels, times of day such as dawn, dusk, and night, and weather phenomena such as rain and fog, in order to assess how robustly each simulator supports detector evaluation outside clear daytime conditions. Finally, the set of evaluated categories could be broadened to include classes such as bicycles, motorcycles, buses, traffic signs, and traffic lights, providing a more comprehensive picture of each simulator's suitability for perception benchmarking.

Author Contributions: Conceptualization, J.H. and B.T.; methodology, B.T.; software, B.T.; validation, B.T.; formal analysis, B.T.; investigation, B.T.; resources, B.T.; data curation, B.T.; writing—original draft preparation, B.T.; writing—review and editing, J.H.; visualization, B.T.; supervision, J.H.; project administration, J.H.; funding acquisition, J.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data for this study are available upon request from the corresponding author.

Acknowledgments: Project no. TKP2021-NVA-23 has been implemented with the support provided by the Ministry of Technology and Industry of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme. During the preparation of this manuscript/study, the authors used ChatGPT (OpenAI, GPT-5.5) for the purposes of language editing, stylistic improvement, and formatting assistance. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Dosovitskiy, A.; Fischer, P.; Ilg, E.; Häusser, P.; Hazırbaş, C.; Golkov, V.; van der Smagt, P.; Cremers, D.; Brox, T. FlowNet: Learning Optical Flow with Convolutional Networks. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2758–2766.
2. Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 23–30.
3. Hoffman, J.; Tzeng, E.; Park, T.; Zhu, J.-Y.; Isola, P.; Saenko, K.; Efros, A.; Darrell, T. CyCADA: Cycle-Consistent Adversarial Domain Adaptation. In Proceedings of the 35th International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018; pp. 1989–1998.
4. Nikolenko, S.I. *Synthetic Data for Deep Learning*; Springer: Cham, Switzerland, 2021.
5. Dosovitskiy, A.; Ros, G.; Codevilla, F.; López, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In Proceedings of the 1st Annual Conference on Robot Learning (CoRL), Mountain View, CA, USA, 13–15 November 2017; pp. 1–16.
6. Hurl, B.; Czarnecki, K.; Waslander, S. Precise Synthetic Image and LiDAR (PreSIL) Dataset for Autonomous Vehicle Perception. In Proceedings of the IEEE Intelligent Vehicles Symposium (IV), Paris, France, 9–12 June 2019; pp. 2522–2529.
7. Kim, J.-C. CarFree: Hassle-Free Object Detection Dataset Generation Using Carla Autonomous Driving Simulator. *Appl. Sci.* **2022**, *12*, 281. [CrossRef]
8. *ISO 26262:2018*; Road Vehicles—Functional Safety. International Organization for Standardization: Geneva, Switzerland, 2018.

9. aiMotive. *aiSim Customer Documentation*; aiMotive Kft.: Budapest, Hungary, 2026.
10. aiMotive. aiSim. Available online: https://aimotive.com/aisim (accessed on 25 May 2026).
11. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
12. Jocher, G.; Chaurasia, A.; Qiu, J. Ultralytics YOLOv8. Available online: https://github.com/ultralytics/ultralytics (accessed on 25 May 2026).
13. Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. In *Computer Vision—ECCV 2024; Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2024; Volume 15089, pp. 1–21.
14. Wang, A.; Chen, H.; Liu, L.; Chen, K.; Lin, Z.; Han, J.; Ding, G. YOLOv10: Real-Time End-to-End Object Detection. In Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 10–15 December 2024.
15. Jocher, G.; Qiu, J. Ultralytics YOLO11. Available online: https://github.com/ultralytics/ultralytics (accessed on 25 May 2026).
16. Tian, Y.; Ye, Q.; Doermann, D. YOLOv12: Attention-Centric Real-Time Object Detectors. *arXiv* **2025**, arXiv:2502.12524.
17. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
18. Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In *Computer Vision—ECCV 2014; Lecture Notes in Computer Science*; Springer: Cham, Switzerland, 2014; Volume 8693, pp. 740–755.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.