Journal of Information Processing Vol.0 1–8 (??? 1992)

[DOI: 10.2197/ipsjjip.0.1]

Regular Paper

Enhancing Safety in Autonomous Driving: Integrating CARLA for Multi-Sensor Dataset Generation and Advanced Scenario Testing

Shiyang Guan[,a)] Yijun Lu[,b)] Jati H. Husen[,c)] Haowei Cheng[,d)] Hironori Washizaki[,e)] Naoyasu Ubayashi[,f)] Nobukazu Yoshioka[,g)]

Received: xx xx, xxxx, Accepted: xx xx, xxxx

Abstract: Simulation environments are vital to autonomous driving research, enabling safe and cost-effective studies of dense traffic, adverse weather, and sidewalk navigation. Yet real-world data collection for these scenarios can be hazardous and expensive. To address this, we integrate the CARLA simulator for dataset generation and scenario construction. Our approach leverages CARLA's autopilot to capture traffic-sign data via LiDAR detection and semantic segmentation, and employs a custom manual script for sidewalk data. We also vary weather and traffic density, using RoadRunner for specialized maps. Preliminary results suggest CARLA-generated data helps identify domain gaps when combined with real GTSRB data, and improves segmentation (IoU) in sidewalk scenes. Looking ahead, we propose automated scenario generation integrating with M3S. Engineers can define high-level objectives and then incorporate them into CARLA, ensuring robust evaluations for critical autonomous driving scenarios.

Keywords: CARLA Simulator, Autonomous driving, dataset generation, Semantic segmentation, GTSRB, LiDAR detection

1. Introduction

In recent years, autonomous driving systems have witnessed rapid advancements, driven by breakthroughs in computer vision, machine learning, and sensor fusion [1], [2], [3]. Nevertheless, collecting real-world data to train and evaluate these systems remains prohibitively expensive, time-consuming, and potentially hazardous. This issue is exacerbated when studying edge cases—such as adverse weather conditions or complex urban environments—where safety risks and data scarcity often impede progress [4]. To address these challenges and mitigate risks prior to real-world deployment, researchers increasingly employ a cycle of iterative development: identifying high-risk scenarios, leveraging high-fidelity simulators to test proposed solutions, and then returning to real-world experiments for validation and refinement [5].

High-fidelity simulators have thus become indispensable tools for enabling controlled and repeatable testing across a variety of conditions without endangering lives [6]. Among these, CARLA stands out due to its adaptability, offering a broad array of sensors, customizable environments, and extensible scripting options [7]. However, out-of-the-box solutions often do not adequately support more specialized tasks—such as sidewalk navigation using smaller vehicles or simulating customized weather and traffic conditions—areas that have received relatively little attention compared to main-road scenarios in advanced driver assistance systems (ADAS) [8].

This paper addresses these limitations by integrating and extending CARLA for two main objectives: (1) dataset generation and (2) scenario construction. First, we introduce a comprehensive data collection pipeline tailored for both large-scale road driving and the underexplored domain of sidewalk navigation. Using a bicycle-like platform, our approach captures multi-modal sensor data (RGB, semantic, and depth images) in environments characterized by non-deterministic pedestrian behavior—a key challenge in real-world urban sidewalks. This pipeline not only fills an existing research gap but also enables more targeted data acquisition for ADAS development.

Second, our work extends the Multi-view Modeling Framework for ML Systems (M3S) originally proposed by Dr. Husen in our lab [9]. M3S generates high-level requirements that reveal gaps in current training datasets. Building on these insights, we integrate a model training pipeline with CARLA that efficiently generates the necessary data to bridge these gaps. In our implementation,

[1] Waseda University, Tokyo 101–0062, Japan
[a)] guanshiyang0509@toki.waseda.jp
[b)] yijun@ruri.waseda.jp
[c)] jati.h@asagi.waseda.jp
[d)] haowei.cheng@fuji.waseda.jp
[e)] washizaki@waseda.jp
[f)] ubayashi@aoni.waseda.jp
[g)] nobukazu@engineerable.ai

engineers can customize vehicle paths, weather conditions, and vehicle models via simple, single-keyboard-hit commands. In addition, an API facilitates on-the-fly adjustments to environmental parameters, thereby significantly reducing the cost and complexity of data generation.

The primary contributions of this work are threefold:

- We propose a novel data collection pipeline that supports both traditional main-road scenarios and the less-explored sidewalk navigation context—addressing an important gap in current ADAS research.
- We extend the M3S framework by integrating a model training pipeline that leverages CARLA to generate targeted training data. This integration uses high-level requirements to identify missing data components and enables rapid customization of simulation parameters (e.g., vehicle path, weather, and model) with minimal user intervention.
- We demonstrate a customizable scenario construction approach that employs CARLA's configurable features—including weather parameter alterations, traffic complexity control, and custom map integration—to create varied test conditions. This facilitates a continuous Risk → Simulator → Real → Risk cycle for iterative model improvement.

By combining these methods, our approach not only addresses the inherent challenges of replicating non-deterministic pedestrian behavior in simulation but also provides a cost-effective and efficient alternative to traditional data collection methods. The remainder of this paper is organized as follows. Section 2 reviews relevant background and related work on simulation platforms and dataset generation for autonomous driving research [10], [11]. Section 3 details our methodology for both dataset generation and scenario construction, with an emphasis on sidewalk-specific data acquisition. Section 4 describes our experimental setup and presents empirical results focused on bridging the simulation-to-real gap [12]. In Section 5, we provide a comprehensive performance evaluation and discuss strategies for continuous improvement within the extended M3S framework [13]. Finally, Section 6 concludes the paper and outlines future directions for further automation and extension of the proposed framework.

2. Background and Related Work

Developing reliable machine learning (ML) components for autonomous driving systems (ADS) demands rigorous testing under a variety of operating conditions, including edge cases such as extreme weather or dense urban environments [10], [11]. Simulation-based approaches have consequently become a core strategy to address these challenges, enabling researchers to explore scenarios that might be hazardous or prohibitively expensive to replicate in the real world [14]. This section reviews relevant work in simulation-augmented testing, with particular emphasis on the Multi-view Modeling Framework for ML Systems (M3S), the CARLA simulator, custom environment generation (e.g., RoadRunner), and sim-to-real transfer. Additionally, we discuss emerging directions that extend simulation to less-studied tasks like sidewalk navigation, smaller vehicle control, and the iterative Risk → Simulator → Real → Risk framework [9].

2.1 Multi-view Modeling Framework for ML Systems (M3S)

A crucial aspect of ML system design involves ensuring traceability, iterative refinement, and alignment between ML pipelines and broader system requirements. The Multi-view Modeling Framework for ML Systems (M3S) addresses these challenges by structuring the integration of ML components at the system level [9]. Husen et al. demonstrated three distinct approaches to implementing M3S—top-down, model-first, and parallel—highlighting how each approach accommodates evolving ML architectures while maintaining system-wide consistency.

Despite these strengths, M3S typically presupposes the availability of domain-relevant datasets, which can be difficult to collect for edge-case scenarios (e.g., heavy rain or unclear road markings). Furthermore, certain tasks, such as sidewalk-centric navigation, are overlooked in most large-scale datasets. Our work extends M3S by introducing simulation-based data generation aimed at these underrepresented use cases. By doing so, we integrate the iterative loop of Risk → Simulator → Real → Risk into the M3S lifecycle, enabling continuous improvement of ML components tasked with handling rarely encountered but high-impact conditions.

2.2 CARLA Simulator for Advanced Testing

Among various simulation frameworks, CARLA has emerged as a leading open-source platform, providing high-fidelity sensor models (RGB, LiDAR, radar, and semantic segmentation) along with configurable traffic actors and weather conditions [7]. Pahk et al. leveraged CARLA to test lane segmentation models in adverse weather, such as heavy rain, demonstrating that traditional data augmentation techniques often fail to capture real-world complexities [8]. By accurately reproducing conditions like reduced visibility and slippery roads, CARLA facilitates deeper insights into ML models' performance envelopes.

Nevertheless, certain tasks remain challenging with CARLA's default capabilities. For instance, sidewalk navigation—particularly when using smaller vehicles such as bicycles—often requires additional scripting or manual-control routines. Moreover, random pedestrian movement on sidewalks is difficult to replicate with built-in autopilot scripts alone, mandating a more customized approach to achieve realistic—and often unpredictable—behaviors essential for robust data collection and testing.

2.3 Custom Map Generation and RoadRunner

Realistic virtual environments play a pivotal role in

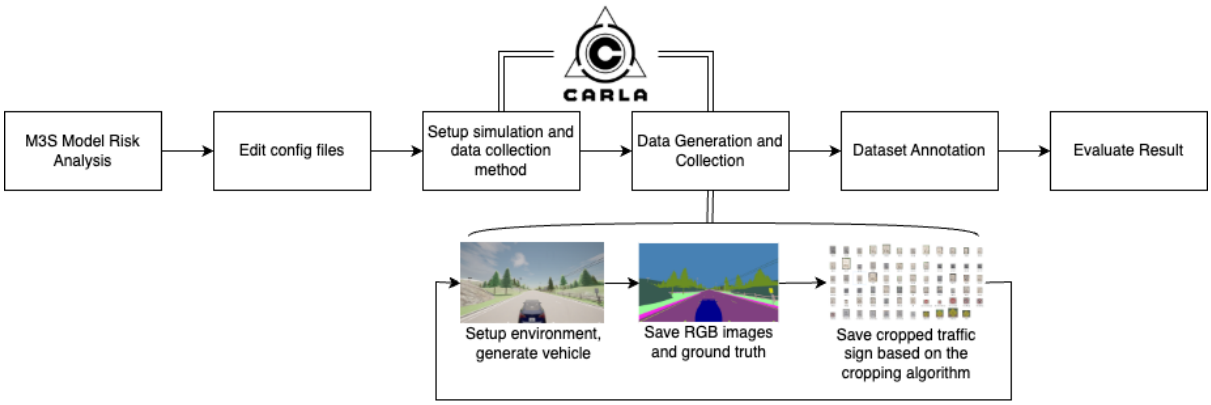

Fig. 1: Workflow of this research

simulation-based ADS testing. Shi et al. demonstrated that combining CARLA with RoadRunner enables the creation of detailed terrains featuring accurate road geometry, signals, and topographies [15]. Studies on lane-keeping assist systems (LKAS) also highlight that high-fidelity environments improve the reliability of performance metrics and reduce the risk of overfitting to simplified or idealized settings [16]. By adapting environmental complexity to specific research questions—ranging from congested urban layouts to rural roads—researchers can systematically evaluate model robustness.

In the context of sidewalk navigation or non-standard vehicle operation, custom maps become even more valuable. RoadRunner and Unreal Engine allow the inclusion of bicycle paths, pedestrian zones, and variable sidewalk widths, offering finer control over relevant features like curb geometry or crossing patterns. These capabilities are essential for capturing the nuances of smaller vehicle dynamics and the unpredictable movement of pedestrians.

2.4 Bridging the Sim-to-Real Gap

Despite the advantages of simulation, a persistent issue is the domain gap: ML models trained primarily on synthetic data often underperform when presented with real-world inputs [12]. One approach to mitigate this problem is sim-to-real transfer, which leverages techniques like CycleGAN to align simulated imagery more closely with real-world textures and lighting [13]. Domain randomization, wherein environmental parameters (e.g., lighting angles, background textures, pedestrian appearances) are randomized in simulation, also reduces overfitting to a single synthetic domain.

However, bridging the sim-to-real gap for sidewalk navigation poses additional hurdles. Pedestrian behaviors in densely populated urban areas are highly variable and can be influenced by subtle social cues, making them especially difficult to capture with generic simulations. Data-driven pedestrian models and carefully curated real-world validation remain essential to ensure that autonomy algorithms do not fail in these unpredictable “corner-case” scenarios.

2.5 Extending Simulation-Based Testing with M3S

Our work builds on these foundations by incorporating simulation-augmented data into the M3S framework, with a focus on understudied tasks such as sidewalk-centric navigation and smaller vehicle (e.g., bicycle) control. Specifically, we customize CARLA to capture traffic-sign data (inspired by the GTSRB benchmark) and sidewalk-focused scenes that default autopilot routines overlook. By adjusting environmental factors—such as rain, fog, and pedestrian density—we assemble comprehensive datasets for evaluating ML-based perception and decision-making under challenging conditions [17], [18].

Additionally, we emphasize continuous risk analysis through iterative testing, aligning with the Risk → Simulator → Real → Risk cycle. After initial deployment in simulation to gauge performance across a variety of scenarios, real-world tests further validate—and potentially update—model assumptions regarding pedestrian dynamics and driver behaviors. Through this integration of systematic simulation, sidewalk-specific data collection, and the multi-view modeling principles of M3S, we aim to identify and address critical safety gaps before real-world deployment. This holistic approach ultimately strengthens reliability for ML systems in autonomous driving contexts, extending beyond conventional lane-following and highway-driving tasks to the intricacies of urban sidewalks.

3. Research Questions

The development of safe and reliable ADS remains a formidable challenge, particularly when accounting for the complex and uncertain nature of real-world environments. In this study, we propose a scenario-based, data-driven approach to analyzing safety risks in ADS by leveraging the high-fidelity simulation capabilities of CARLA. Our method integrates both qualitative and quantitative metrics—including severity, occurrence probability, risk value, and accident likelihood—while explicitly addressing uncertainty evaluation.

This approach evaluates safety at two key levels:

- Operational Level: Assessment of environmental and traffic scenarios, such as adverse weather, variable

traffic densities, and the challenges of sidewalk navigation.

- Component Level: Evaluation of machine learning (ML) model performance and sensor inputs, with a focus on identifying potential failure modes in perception and decision-making modules.

The core objective is to provide a more comprehensive safety analysis than existing methods by bridging the gap between scenario-based hazard identification and quantitative performance metrics. To demonstrate the feasibility and effectiveness of our approach, we conduct a case study that includes:

- Scenario Definition: Utilizing CARLA to design diverse operational conditions that mimic real-world challenges, including adverse weather, sidewalk navigation, and variable traffic densities.
- Data-Driven Evaluation: Collecting and analyzing simulated sensor data (e.g., RGB, LiDAR, depth, and semantic segmentation) to assess ML model robustness and detect potential risk factors.
- Risk Mitigation Strategies: Validating safety measures—such as filtering noisy sensor data and enhancing domain adaptation—to effectively mitigate identified risks.

By systematically integrating both environment-level and model-level risk evaluations, our method not only extends existing simulation-based testing practices but also offers deeper insights into the safety-critical behavior of ML components in autonomous driving systems.

4. Methodology

Our simulation-based data collection pipeline is organized into three key components. The pipeline is implemented using the CARLA simulator, Python, and Pygame, and is designed to generate rich multi-modal datasets from challenging urban scenarios. The three components are as follows: (1) Sidewalk Scenario Simulation, (2) Extension of the M3S Framework for GTRSB Dataset Generation, and (3) Map Creation using RoadRunner. Below, we describe each component in detail, referencing the code implementation where appropriate.

4.1 Sidewalk Scenario Simulation

To capture the complexities of sidewalk navigation—a scenario underexplored in conventional autonomous driving datasets—we designed a simulation that integrates urban roads with dedicated sidewalk areas. The code implementation accomplishes this through the following steps:

4.1.1 Vehicle Deployment and Sensor Setup:

A bicycle-like vehicle (selected via the blueprint crossbike) is spawned at a random location within a CARLA map (e.g., Town03), which features both urban streets and sidewalks. The following code snippet demonstrates this:

```
bp = blueprint_library.filter("crossbike")[0]
spawn_point = random.choice(world.get_map().
    get_spawn_points())
vehicle = world.spawn_actor(bp, spawn_point)
vehicle.set_autopilot(False)  # Manual control
    is essential for sidewalk navigation
actor_list.append(vehicle)
```

After spawning the vehicle, three sensors are attached to it:

- An RGB camera is configured to capture high-resolution images (1920×1080) with a wide field of view.
- A Depth camera is initialized with similar parameters and set to use logarithmic depth conversion.
- A Semantic Segmentation camera is set up using a conversion palette (CityScapes) to provide pixel-level semantic labels.

Each sensor is attached using a fixed transform relative to the vehicle, ensuring that the data from all sensors is synchronized:

```
spawn_point = carla.Transform(carla.Location(x
    =2.5, z=0.7))
sensor = world.spawn_actor(cam_bp, spawn_point
    , attach_to=vehicle)
sensor_dp = world.spawn_actor(camera_depth,
    spawn_point, attach_to=vehicle)
sensor_semseg = world.spawn_actor(
    camera_semseg, spawn_point, attach_to=
    vehicle)
actor_list += [sensor, sensor_dp,
    sensor_semseg]
```

4.1.2 Manual Control and Data Capture:

Since CARLA lacks an auto-drive mode for sidewalks, we implemented manual control using Pygame[1]. The control loop captures keyboard inputs for throttle, brake, and steering, enabling the user to navigate the bicycle along sidewalks. A key feature is the ability to start data collection with a single key press (e.g., toggling the save_images flag when the P key is pressed):

```
for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_p:
            save_images = not save_images
            print(f"Save␣images:␣{save_images}
                ")
```

Within the main loop, after processing the control inputs, the vehicle's control state is updated, and the simulation advances with:

```
vehicle.apply_control(control)
world.tick()
```

4.1.3 Synchronized Multi-Modal Data Collection:

The sensor callback functions (registered via the listen method) invoke the store_image() function to store incoming images in a dictionary, indexed by the simulation frame. Once images from all three sensors (RGB, depth, and semantic segmentation) are available for a given frame, they are saved to disk in modality-specific folders. The following functions illustrate this process:

```python
def store_image(image, sensor_type, color_converter=None):
    global image_store, save_images
    if save_images:
        if color_converter:
            image.convert(color_converter)
        if image.frame not in image_store:
            image_store[image.frame] = {}
        image_store[image.frame][sensor_type] = image
        save_images_from_frame(image.frame)

def save_images_from_frame(frame_number):
    global image_store
    if frame_number in image_store and len(image_store[frame_number]) == 3:
        for sensor_type, image in image_store[frame_number].items():
            if sensor_type == 'rgb':
                folder = f'{dt_now}_rgb'
            elif sensor_type == 'depth':
                folder = f'{dt_now}_depth'
            elif sensor_type == 'semseg':
                folder = f'{dt_now}_seg'
            os.makedirs(folder, exist_ok=True)
            image.save_to_disk(f'{folder}/{sensor_type}_{frame_number:06d}.png')
        del image_store[frame_number]
```

This design ensures that the multi-modal sensor data is both temporally aligned and stored in an organized manner, facilitating later analysis.

4.2 Extension of the M3S Framework for GTRSB Dataset Generation

Building on the M3S framework, we extend our pipeline to generate a dataset analogous to the German Traffic Sign Recognition Benchmark (GTRSB). This extension integrates LiDAR data with RGB and semantic segmentation images to detect, crop, and annotate traffic signs. The process comprises the following steps:

4.2.1 Traffic Sign Detection via LiDAR

A LiDAR sensor is used to identify the location of traffic poles by processing the point cloud data. The function semantic_lidar_data iterates over the detections, computes the distance for each, and checks for traffic signs using the object tag. When a traffic sign (object tag 5) is detected, the corresponding RGB image is retrieved from the image queue and saved:

```python
def semantic_lidar_data(point_cloud_data):
    for detection in point_cloud_data:
        if detection.object_tag <= 22:
            distance = math.sqrt(detection.point.x**2 + detection.point.y**2 + detection.point.z**2)
            if detection.object_tag == 5:  # Traffic sign detected
                with open('traffic_sign.txt', 'a') as file:
                    file.write("Traffic sign detected, distance: {}\n".format(distance))
                image = image_queue.get()
                image.save_to_disk("traffic_sign/%06d.png" % (image.frame))
```

4.2.2 Traffic Sign Cropping and Annotation

Once a traffic sign is detected, semantic segmentation images are utilized to create a mask and compute bounding boxes around the sign. The functions get_mask and get_bbox_from_mask generate the segmentation mask and bounding boxes, respectively[19]. The cropped image is then saved for further analysis:

```python
def get_mask(seg_im, rgb_value):
    hsv_value = cv2.cvtColor(rgb_value, cv2.COLOR_RGB2HSV)
    hsv_low = np.array([[[hsv_value[0][0][0]-5, hsv_value[0][0][1], hsv_value[0][0][2]-5]]])
    hsv_high = np.array([[[hsv_value[0][0][0]+5, hsv_value[0][0][1], hsv_value[0][0][2]+5]]])
    mask = cv2.inRange(seg_im, hsv_low, hsv_high)
    return mask

def get_bbox_from_mask(mask):
    label_mask = measure.label(mask)
    props = measure.regionprops(label_mask)
    return [prop.bbox for prop in props]

# Example usage after obtaining segmentation and RGB images:
mask = get_mask(img_semseg_hsv, object_list['traffic_sign'])
bboxes = get_bbox_from_mask(mask)
output_dir = "cropped_images"
os.makedirs(output_dir, exist_ok=True)
for i, bbox in enumerate(bboxes):
    minr, minc, maxr, maxc = bbox
    cropped_image = img[minr:maxr, minc:maxc]
    cropped_image_path = os.path.join(output_dir, f"{image.frame:06d}_cropped_{i}.png")
    cv2.imwrite(cropped_image_path, cropped_image)
```

4.2.3 Dataset Structuring and Export

Finally, the cropped images and corresponding annotations (bounding boxes and segmentation masks) are compiled in COCO format. Each record includes the image file name, dimensions, and object annotations. The structured dataset is saved as a JSON file for subsequent training and evaluation:

```python
record = {}
record['file_name'] = "test_images/%06d.png" % image.frame
record['image_id'] = global_count
record['height'], record['width'] = cv2.imread(record['file_name']).shape[:2]
record['annotations'] = objects  # 'objects' is a list of annotations in COCO format

with open('dataset.json', 'w') as file:
    json.dump(processed_dataset_dicts, file)
```

This detailed extension enables the automated extraction and annotation of traffic signs from simulation data,

effectively generating a dataset tailored for traffic sign recognition tasks.

4.2.4 Map Creation Using RoadRunner

To further improve the fidelity and flexibility of our simulation, we integrate custom map creation using RoadRunner. RoadRunner, together with Unreal Engine, is used to design maps that:

- Accurately Represent Urban Environments: Maps include detailed sidewalk geometries, traffic sign placements, and pedestrian zones that are often missing in standard CARLA maps.
- Tailor Scenarios to Specific Research Needs: Custom maps enable us to replicate diverse real-world conditions, enhancing the realism of both the sidewalk and urban driving scenarios.

The maps generated in RoadRunner are imported into CARLA, where they serve as the simulation environment for both the sidewalk data collection and the GTRSB dataset generation experiments. This allows for a controlled yet highly realistic evaluation of the ML models under various conditions.

5. Experiment Setup

In this section, we describe the experimental setup, detailing the configuration of the CARLA simulation environment, scenario settings, and data collection configuration. These settings are essential for capturing accurate and realistic multi-modal datasets for our research.

5.1 CARLA Configuration

The CARLA simulator is configured to create a realistic simulation environment with appropriate sensor and vehicle settings. Our configuration includes:

- Simulation Environment: The experiments are conducted using CARLA with a selected map (e.g., Town03) that features both urban roads and sidewalk scenarios.
- Sensor Parameter Settings: We deploy multiple sensors, including RGB cameras, LiDAR sensors, and semantic segmentation cameras. Table 1 summarizes key sensor parameters. For example, the RGB camera is configured with an image size of 128×96 pixels, a field-of-view of 110 degrees, and 32 channels.
- Vehicle Model Selection: The vehicle is chosen from CARLA's blueprint library (e.g., Model3 or crossbike) based on the experimental requirements.

5.2 Scenario Settings

The simulation scenarios are configured to replicate various real-world conditions. Key settings include:

- Weather Conditions: The weather parameters (e.g., cloudiness, precipitation, sun altitude) are adjusted to simulate different environmental conditions.
- Traffic Flow Density: The number of vehicles spawned in the simulation is controlled to emulate varying levels of urban congestion.

Table 1: CARLA Attribute Table

Attribute Name	Value
Sensor	RGB_Camera
image_size_x	640
image_size_y	480
fov	110
channels	32
point_per_second	56000
rotation_frequency	80
range	25
lower_fov	-80
Transform	carla.Location(-5.5,0,2.5)
carla.Location	(0,0,2)
Blueprint	Model3

- Pedestrian Density: Pedestrian density is adjusted to generate realistic scenarios with diverse levels of pedestrian activity.

Fig. 2: Sunny, clear road condition

Fig. 3: Rainy, crowded pedestrians condition

Fig. 4: Comparison of road conditions under different environmental scenarios.

5.3 Data Collection Configuration

Our data collection configuration ensures that multi-modal sensor data is accurately captured and stored. Key aspects include:

- Data Collection Frequency: Sensor data is captured at a fixed frequency, using a fixed delta time (e.g., 0.05 seconds) in synchronous mode to maintain temporal consistency.
- Storage Format: Captured images and sensor data are saved in modality-specific folders (e.g., RGB, depth, and semantic segmentation) using standardized formats (e.g., PNG for images and JSON for annota-

tions).

- Synchronization Strategy: A frame-indexed data structure is employed to ensure that data from multiple sensors are synchronized before saving, guaranteeing consistency across the dataset.

By carefully configuring the CARLA environment, scenario settings, and data collection parameters, our experimental setup provides a robust and repeatable platform for generating high-quality multi-modal datasets for autonomous driving research.

6. Case Study

In this section, we evaluate our proposed scenario-based, data-driven risk analysis method by applying it to two representative case studies using the CARLA simulator. In the first case study, we focus on generating a training dataset for traffic sign recognition inspired by the German Traffic Sign Recognition Benchmark (GTSRB). In the second case study, we collect sidewalk data using manual control of a bicycle agent and evaluate segmentation performance using Intersection over Union (IoU) metrics.

6.1 GTSRB Dataset Generation for Traffic Sign Recognition

The primary objective of this case study is to validate the effectiveness of our extended M3S framework in generating a synthetic dataset for traffic sign recognition. We evaluate three training configurations:

(1) CARLA-generated Data Only: Synthetic images and annotations produced entirely in CARLA.

(2) CARLA-generated Data Mixed with Real Data: A hybrid dataset combining synthetic CARLA data with a portion of real GTSRB images.

(3) Real GTSRB Data Only: The standard GTSRB dataset used as a baseline.

Fig. 5: Sample image from the GTSRB dataset

These datasets are used to train our repair model. As shown in Table 2, the model trained on the CARLA-generated data (alone or mixed with real data) achieves competitive performance. In particular, the synthetic data are easy to integrate with real data, underscoring the efficiency and quality of the CARLA-generated dataset.

6.2 Sidewalk Data Collection and IoU Evaluation

For the sidewalk case study, a bicycle agent was manually driven along urban sidewalks to collect multi-modal sensor data. This data was used to train a sidewalk segmentation model. The performance of the model was then evaluated using IoU metrics on a validation set that includes varying environmental conditions.

Table 2: GTSRB Case Study Results

Dataset	Accuracy (%)	F1-Score
CARLA-generated Data Only	82.5%	0.80
CARLA + Real Data	87.3%	0.85
Real GTSRB Data Only	85.0%	0.83

Table 3 presents a comparison of IoU values for key categories between the baseline model (M2) and the model trained with CARLA-generated data (M4). The improved IoU in critical categories such as Driveable and Obstacle regions confirms that the synthetic data enhances model performance.

Table 3: IoU Comparison between M2 (Baseline) and M4 (CARLA-Enhanced)

Metric	M2 IoU	M4 IoU	Improvement
Mobility	0.0000	0.0050	+0.0050
Driveable	0.3486	0.3600	+0.0114
Obstacle	0.3392	0.3500	+0.0108
Water	0.0000	0.0000	0.0000
Indoor	0.0000	0.0000	0.0000
Void	0.0528	0.0600	+0.0072
Total	0.4559	0.4700	+0.0141

6.2.0.1 Observations:

- Traffic Sign Recognition: The repair model trained on the CARLA-generated data exhibits a strong ability to integrate with real-world data. The hybrid model (CARLA + Real) shows a notable improvement in both accuracy and F1-Score compared to using real data alone.
- Sidewalk Segmentation: The IoU metrics for the M4 model are consistently higher than those for the baseline M2 model, especially in the Driveable and Obstacle categories. This indicates that synthetic data helps the model better delineate walkable paths and identify obstacles.

These results confirm that our approach not only enables efficient dataset generation using CARLA but also improves the performance of downstream ML components in both traffic sign recognition and sidewalk segmentation tasks. Future work will explore further refinements in sensor fusion and domain adaptation to enhance model robustness under even more challenging conditions.

7. Limitations and Future Work

Although our results highlight the effectiveness of simulation-augmented risk analysis, several limitations remain:

- Real-World Transferability: While domain adaptation techniques help, a gap persists between synthetic and real-world domains, potentially affecting model performance in unmodeled conditions.
- Complex Urban Interactions: Current experiments focused on controlled variations; real urban areas feature

a wider range of unexpected scenarios, dynamic obstacles, and nuanced social behaviors.

- Computational Constraints: High-fidelity simulations, particularly those involving multiple sensors and extensive traffic, can require substantial computational resources, limiting scalability in some research or industrial settings.

So, in the future we will aim to refine and expand the proposed method:

- Enhanced Domain Adaptation: Investigate CycleGAN or other advanced generative models to reduce the sim-to-real gap for critical tasks like traffic sign recognition and sidewalk segmentation.
- Adaptive Scenario Generation: Integrate with frameworks like M3S to automatically generate high-risk scenarios based on real-time feedback from ML models, ensuring that edge cases are continually explored.
- Multi-Modal Sensor Fusion: Extend beyond LiDAR, RGB, and depth to include radar or inertial data, increasing robustness in occlusion-heavy or night-time conditions.
- Larger-Scale Evaluations: Apply the methodology to larger road networks with multi-lane highways, tunnels, and diverse pedestrian behaviors to more closely approximate complex urban environments.

8. Conclusion

Our scenario-based, data-driven methodology for analyzing and mitigating safety risks in ADS highlights several key insights. First, the integration of qualitative hazard identification (such as unsafe sidewalks and misread traffic signs) with quantitative metrics (like Intersection over Union and accident rates) provides a comprehensive understanding of system vulnerabilities. Additionally, the importance of simulation diversity was evident, as varying traffic density, weather conditions, and map layouts in the CARLA simulator proved essential for uncovering latent failure modes. This diversity helped reduce domain gaps between synthetic and real-world datasets, enhancing the robustness of our risk assessment.

Furthermore, the iterative approach to risk mitigation played a critical role in improving system performance. By continuously re-evaluating the ADS after implementing enhancements, such as advanced data augmentation techniques and refined control logic, we observed significant gains in both safety and perception robustness. Addressing these challenges not only strengthens the link between simulated testing and real-world validation but also establishes data-driven, scenario-based risk analysis as a foundational strategy for advancing next-generation autonomous driving systems across diverse operational environments.

References

[1] Li, C., Sifakis, J., Yan, R. and Zhang, J.: Rigorous Simulation-based Testing for Autonomous Driving Systems–Targeting the Achilles' Heel of Four Open Autopilots, arXiv preprint arXiv:2405.16914 (2024).

[2] Zhou, Y., Simon, M., Peng, Z., Mo, S., Zhu, H., Guo, M. and Zhou, B.: SimGen: Simulator-conditioned Driving Scene Generation, arXiv preprint arXiv:2406.09386 (2024).

[3] Biagiola, M., Stocco, A., Riccio, V. and Tonella, P.: Two is better than one: digital siblings to improve autonomous driving testing, Empirical Software Engineering, Vol. 29, No. 4, pp. 1–33 (2024).

[4] Song, Z., He, Z., Li, X., Ma, Q., Ming, R., Mao, Z., Pei, H., Peng, L., Hu, J., Yao, D. et al.: Synthetic datasets for autonomous driving: A survey, IEEE Transactions on Intelligent Vehicles (2023).

[5] Burnett, K., Yoon, D. J., Wu, Y., Li, A. Z., Zhang, H., Lu, S., Qian, J., Tseng, W.-K., Lambert, A., Leung, K. Y. et al.: Boreas: A multi-season autonomous driving dataset, The International Journal of Robotics Research, Vol. 42, No. 1-2, pp. 33–42 (2023).

[6] Enzweiler, M. and Gavrila, D. M.: Monocular Pedestrian Detection: Survey and Experiments, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 31, No. 12, pp. 2179–2195 (online), DOI: 10.1109/TPAMI.2008.260 (2009).

[7] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. and Koltun, V.: CARLA: An open urban driving simulator, Conference on robot learning, PMLR, pp. 1–16 (2017).

[8] Pahk, J., Park, S., Shim, J., Son, S., Lee, J., An, J., Lim, Y. and Choi, G.: Lane Segmentation Data Augmentation for Heavy Rain Sensor Blockage using Realistically Translated Raindrop Images and CARLA Simulator, IEEE Robotics and Automation Letters (2024).

[9] Husen, J. H., Washizaki, H., Runpakprakun, J., Yoshioka, N., Tun, H. T., Fukazawa, Y. and Takeuchi, H.: Integrated multi-view modeling for reliable machine learning-intensive software engineering, Software Quality Journal, Vol. 32, No. 3, pp. 1239–1285 (2024).

[10] Muhammad, K., Ullah, A., Lloret, J., Del Ser, J. and de Albuquerque, V. H. C.: Deep learning for safe autonomous driving: Current challenges and future directions, IEEE Transactions on Intelligent Transportation Systems, Vol. 22, No. 7, pp. 4316–4336 (2020).

[11] GerÃ³nimo, D., LÃ³pez, A. M., Sappa, A. D. and Graf, T.: Survey of Pedestrian Detection for Advanced Driver Assistance Systems, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 7, pp. 1239–1258 (online), DOI: 10.1109/TPAMI.2009.122 (2010).

[12] Hu, X., Li, S., Huang, T., Tang, B., Huai, R. and Chen, L.: How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence, IEEE Transactions on Intelligent Vehicles (2023).

[13] Zhu, J.-Y., Park, T., Isola, P. and Efros, A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks, Proceedings of the IEEE international conference on computer vision, pp. 2223–2232 (2017).

[14] Birchler, C., Ganz, N., Khatiri, S., Gambi, A. and Panichella, S.: Cost-effective simulation-based test selection in self-driving cars software, Science of Computer Programming, Vol. 226, p. 102926 (2023).

[15] Shi, L.: ROUGH TERRAIN AUTONOMOUS GROUND VEHICLE SIMULATION: CARLA SIMULATION AND DEEP LEARNING BASED PREDICTIVE MODEL, PhD Thesis, Johns Hopkins University (2022).

[16] Iqbal, M., Han, J. C., Zhou, Z. Q., Towey, D. and Chen, T. Y.: Metamorphic testing of Advanced Driver-Assistance System (ADAS) simulation platforms: Lane Keeping Assist System (LKAS) case studies, Information and Software Technology, Vol. 155, p. 107104 (2023).

[17] Harb, J., Rébéna, N., Chosidow, R., Roblin, G., Potarusov, R. and Hajri, H.: Frsign: A large-scale traffic light dataset for autonomous trains, arXiv preprint arXiv:2002.05665 (2020).

[18] Leibner, P., Hampel, F. and Schindler, C.: GERALD: A novel dataset for the detection of German mainline railway signals, Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit, Vol. 237, No. 10, pp. 1332–1342 (2023).

[19] Simulator, C.: Bounding boxes, https://carla.readthedocs.io/en/latest/tuto_G_bounding_boxes/ (2025). Accessed: Feb. 5, 2025.