Sim-on-Wheels: Physical World in the Loop Simulation for Self-Driving

Yuan Shen, Bhargav Chandaka, Zhi-Hao Lin, *Graduate Student Member, IEEE*, Albert Zhai, Hang Cui, David Forsyth, *Fellow, IEEE*, and Shenlong Wang, *Member, IEEE*

***Abstract*—We present Sim-on-Wheels, a safe, realistic, and vehicle-in-loop framework to test autonomous vehicles' performance in the real world under safety-critical scenarios. Sim-on-wheels runs on a self-driving vehicle operating in the physical world. It creates virtual traffic participants with risky behaviors and seamlessly inserts the virtual events into images perceived from the physical world in real-time. The manipulated images are fed into autonomy, allowing the self-driving vehicle to react to such virtual events. The full pipeline runs on the actual vehicle and interacts with the physical world, but the safety-critical events it sees are virtual. Sim-on-Wheels is safe, interactive, realistic, and easy to use. The experiments demonstrate the potential of Sim-on-Wheels to facilitate the process of testing autonomous driving in challenging real-world scenes with high fidelity and low risk.**

***Index Terms*—Autonomous agents, simulation and animation, robot safety.**

I. Introduction

EVALUATING how a self-driving car performs in dangerous scenarios is hard. Pure real-world evaluations create situations that are dangerous to participants, while pure simulation evaluations may simulate various scenarios inaccurately, such as cases in which the vehicle has extreme control inputs. This letter describes a mixed method, Sim-on-Wheels. In Sim-on-Wheels, we run actual autonomy stack on real cars, but create scenarios by inserting people and objects into the sensor feed in real-time. This means we can evaluate the autonomy stack in scenarios known to be dangerous to pedestrians without risking harm because the pedestrians are simulated. Furthermore, we apply the control inputs to a real vehicle. If the autonomy could cause an uncontrolled skid, we will be able to measure that. Fig. 1 illustrates our evaluation pipeline and Table I compares it to previous methods. In contrast to previous approaches, Sim-on-Wheels is simultaneously safe, interactive, realistic, and easy to use.

There is no current consensus on evaluation protocols for autonomous vehicles. Safety evaluation is typically through a combination of **real-world road-tests**, **off-policy data collection**, and **computer simulation**. Real-world testing is a resource-intensive and risky process, and testing in some scenarios is unethical (because there is a strong chance of hurting a participant). Annoyingly, these are the cases where evaluation is particularly important. Off-policy data can be an effective tool for training and evaluating perception algorithms, but does not yield a closed-loop evaluation of the safety of the entire autonomy stack. Computer simulation is safe and scalable, but is not currently reliable in extreme physical and mechanical situations. Sim-on-Wheels is a mashup of real-world road tests (so we can observe true vehicle behavior) and computer simulation (so we don't have to risk harm to participants).

An ideal self-driving evaluation environment should be safe, realistic, and closed-loop. Achieving **safe** evaluation is challenging, because one should be evaluating dangerous scenarios but experiments that pose a risk to life are unethical. **Realistic** evaluation is essential – we need to be sure that evaluation predictions reflect real-life behavior. Finally, **closed-loop** evaluation is essential because we must faithfully evaluate how the controller behaves during actual interaction with the environment. Sim-on-Wheels is intrinsically safe and closed-loop. We show that Sim-on-Wheels results are realistic by both evaluating the realism of the inserted objects and by comparing conclusions on real and Sim-on-Wheels scenarios (Section IV-D). We use Sim-on-Wheels to "spoof" a total of 40 variations of safety-critical scenarios using two different autonomous vehicle pipelines (Section IV-C). With the capability of testing scenarios configured at system limit, our Sim-on-Wheel framework reveals our modular agent is more cautious than our end-to-end learned agent in terms of obstacle avoidance, achieving a lower collision rate but taking longer to reach the goals.

Manuscript received 25 May 2023; accepted 26 September 2023. Date of publication 18 October 2023; date of current version 1 November 2023. This letter was recommended for publication by Associate Editor J. D. Hernández and Editor A. Bera upon evaluation of the reviewers' comments. This work was supported in part by the Amazon Research Award, Nvidia Hardware Grants, the Insper-Illinois Innovative Grant, the NCSA Faculty Fellow under NSF Award 2331878, and in part by the Illinois Smart Transportation Infrastructure Initiative under Grant STII-21-07. *(Yuan Shen and Bhargav Chandaka contributed equally to this work.) (Corresponding author: Shenlong Wang.)*

Yuan Shen, Bhargav Chandaka, Zhi-Hao Lin, Albert Zhai, David Forsyth, and Shenlong Wang are with the Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL 61820 USA (e-mail: yshen47@illinois.edu; bhargav9@illinois.edu; cl121@illinois.edu; azhai2@illinois.edu; daf@illinois.edu; shenlong@illinois.edu).

Hang Cui is with the Center for Autonomy, University of Illinois Urbana-Champaign, Champaign, IL 61820 USA (e-mail: hangcui3@illinois.edu).

Additional results and open-sourced code are available on our project page here: https://sim-on-wheels.github.io/.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3325689, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3325689

II. Related Work

Self-driving autonomy: There are two paradigms for self-driving autonomy: (i) modular [9], [26] and (ii) end-to-end [27]. A modular stack has multiple sub-tasks in a pipeline framework, including localization [28], [29], perception [30], [31], [32], planning [33], and control [9], [34], [35]. Advantages include interpretability, modularity, and versatility [27], but tuning the pipeline can be challenging, errors can propagate, and runtime may be slow. End-to-end autonomy directly maps sensor input to planner or controller commands [36], [37], [38]. End-to-end

TABLE I
COMPARING RELATED WORKS

	Realistic Sensor	Real-world Physics	Closed-loop	Safe	Convenient
Off-policy datasets [1], [2], [3]	✓			✓	✓
Real-world: road test [4], [5], [6]	✓	✓	✓		
Real-world: test track [7], [8], [9]	✓	✓	✓	✗	
Simulation: CG-based [10], [11], [12], [13]			✓	✓	✓
Simulation: data-driven [14], [15], [16], [17], [18], [19]	✗		✓	✓	✓
Vehicle-in-the-loop [20], [21], [22], [23], [24], [25]		✓	✓	✓	✓
Sim-on-Wheels (ours)	✓	✓	✓	✓	✓

A reliable self-driving vehicle evaluation framework requires providing realistic sensors, realistic physics, and closed-loop interaction, all while being safe and easy to use. We situate past frameworks along these five dimensions and discuss them in Section II.

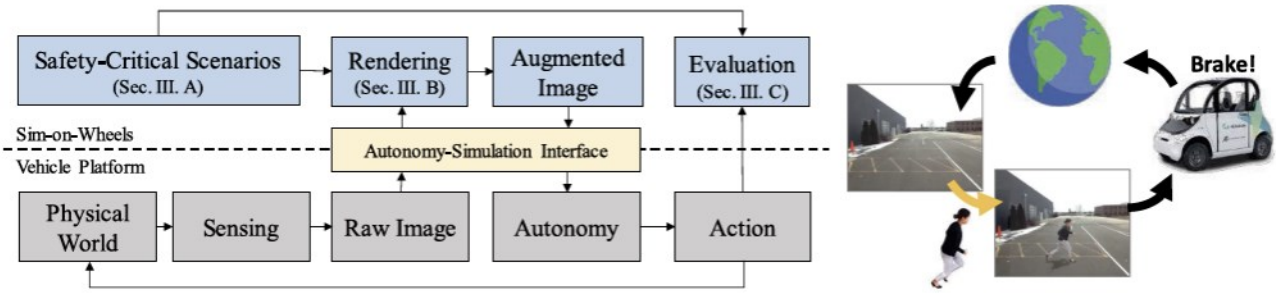

Fig. 1. Sim-on-Wheels Pipeline. In Sim-on-Wheels' evaluation paradigm, vehicle autonomy is evaluated on images that are perceived in the real world but transmitted to the onboard simulator and manipulated in real-time to show important and dangerous traffic scenarios. The autonomy is asked to react to the manipulated sensory input as if the scenario actually happened. Onboard evaluation can be conducted in real-time to verify the safety and effectiveness of the autonomy.

methods are usually simple to develop and fast to run, but tend to be difficult to interpret and less robust to environmental changes, making it hard to diagnose errors, establish safety guarantees, and incorporate traffic rules [39]. Recent advances in end-to-end autonomy have demonstrated promising results by combining the strengths of both paradigms [37], [40]. Sim-on-Wheels is designed to be agnostic to the autonomy methods under evaluation, providing a platform for comprehensive hardware-in-the-loop evaluation of any autonomy stack. We evaluate both modular and end-to-end autonomy in Section IV-C.

Self-driving evaluation: Evaluation practice is shaped by an important tension between safety and accuracy (accurate evaluation requires assessment of dangerous scenarios, posing risks to life). One strategy is to use off-policy datasets [1], [2], [3], which are safe and convenient. But because they do not close the perception-action loop, such evaluations cannot accurately assess a full autonomy stack. Another is to use real-world road tests [4], [5], [6]. These are expensive and pose large risks to safety [41], and so are necessarily limited in scope. Road tests on test tracks [7], [8], [9] are somewhat safer than actual road tests, but are expensive to set up and necessarily lack environment diversity.

Yet another is to use a simulator, which is safe and convenient. Simulated sensor inputs (as in [10], [11], [12], [13]) face a sim2real gap, despite significant literature on improving the realism of simulation (e.g. data-driven simulation in [14], [15], [17], [18]; dynamic models in [42], [43], [44]; environments in [16], [45], [46], [47]). It is extremely difficult to be sure that a simulator captures all relevant physical modeling, especially for dangerous scenarios, where one expects extreme control inputs and odd physics may become important. For example, reverted rubber hydroplaning is an effect where very aggressive braking causes tire rubber to break down and capture a surface water film that breaks contact with the road; this and similar effects significantly affect the safety of a stack, but may not appear in simulators. Although modeling capacity could be added to simulators, it remains difficult to know what to add and when to stop. In contrast, Sim-on-Wheels uses a real vehicle (and so relies on nature for these effects) but simulates dangerous scenarios (and so does not endanger participants).

Vehicle-in-the-loop simulation: Sim-on-Wheels is considered a vehicle-in-the-loop simulation, because it incorporates the entire vehicle into the test. Early such methods use a simulated driving environment [20], [48], [49], [50], with attendant sim2real problems. MiRE [21] improves realism by using a body tracking system to map a human into the scene to act as a pedestrian, but the environment is far from realistic. AR on LiDAR [22] inserts objects in a perceptually realistic manner into LiDAR point clouds (but not RGB images). WIL [23] is a general framework for integrating simulated sensor inputs and real inputs, but does not attend to rendering realism. Hallerbach et al. propose to automatically generate scenario configurations for various X-in-the-loop settings, including car-in-the-loop [24], [25]. However, they only simulate at the traffic level, without explicit camera sensor simulation when testing on a real car. In contrast, Sim-on-Wheels provides realistic rendering aimed at specific, safety-critical scenarios.

III. SIMULATION IN THE PHYSICAL WORLD

Sim-on-Wheels operates by inserting actors, objects, and their animations into the camera stream observed by a controller for a physical autonomous vehicle platform (the "ego-vehicle") moving in a real test space. Fig. 1 depicts the entire pipeline of our framework. There are three main components. *Authoring:* one must first author a driving scenario to be

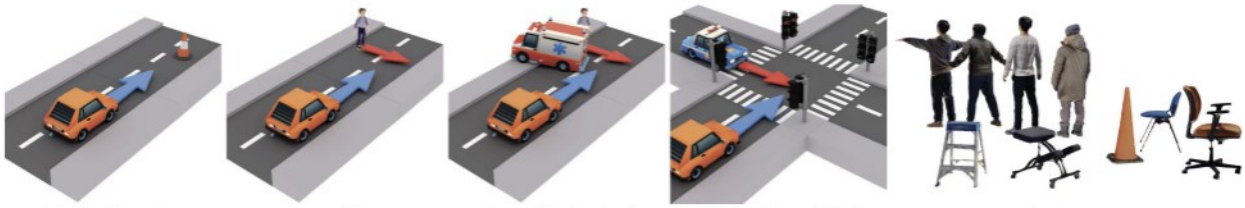

Fig. 2. Illustration of Testing Scenarios and 3D Assets. Left: Our scenarios depict common pre-collision events such as road obstacles, jaywalker, jaywalker with occlusion, and traffic light violations. These scenarios are represented as customizable and reproducible spatial-temporal waypoint trajectories for all actors, with triggers, and can be easily expanded upon. Right: We show a subset of our 3D assets, generated from real-world 3D scans using an iPhone equipped with LiDAR.

evaluated (which is likely to be safety critical), including defining appropriate real-world waypoints and animation sequences for virtual actors/objects, and determining the planned path for the ego-vehicle to follow (Section III-A). *Insertion rendering:* a real-time procedure takes raw RGB-D images, composites the simulated events into the image stream, and re-publishes the composite images to the agent. Sufficiently realistic insertion rendering means the ego-vehicle should react in real-time to the inserted objects as if they were truly present (Section III-B). *Evaluation:* metrics such as collision rate, trip completion time, comfort, and goal reaching rate are computed onboard to evaluate the effectiveness and safety of different autonomy agents (Section III-C).

A. Authoring Safety-Critical Scenarios

We choose safety-critical pre-crash scenarios based on the NHTSA pre-crash event report [51]. Our scenarios encompass common traffic events such as static obstacles, jaywalking, jaywalking with occlusions, and traffic light violations. They are modeled as spatial-temporal waypoint trajectories for all actors, allowing for full reproducibility of each scenario. The testing is conducted to mimic both rural and urban environments, either in a straight road segment or a four-way intersection. A selection of the scenarios is depicted in Fig. 2.

Authoring involves selecting from a rich collection of 3D assets, including artist-designed assets from SketchFab [52] and in-house created assets reconstructed using an iPhone and multi-view reconstruction software [53]. Once 3D assets (in real-world scale) are generated, we place the static objects manually in the scene. As for dynamic agents, based on manually-set start and end positions, we sample a feasible trajectory based on the occupancy from a real-world map and animate the agent along it using Mixamo [54] with realistic and diverse human animations, such as walking and running.

The evaluation procedure involves triggering each scenario as the ego-vehicle reaches the trigger zone at a certain speed range. A crash will occur if the vehicle fails to conduct any evasive action. The hyper-parameters of each scenario can be adjusted to control the level of difficulty, including the type and trajectory of static objects for the static obstacle scenarios, the type and trajectory of traffic light runners and trigger distance for the intersection scenarios, and the walking speed, type and number of jaywalkers and trigger distance for the jaywalking scenarios, etc. Our scenario bank can be easily expanded to cover additional safety-critical events. One unique advantage of Sim-on-Wheels is that it enables setting aggressive hyper-parameters without any risk for physical harm to any vehicles or pedestrians, and the results of the evaluations provide a comprehensive understanding of the performance limits of each autonomy stack. Another is that the effect of (say) actor motion or dress on outcomes can be assessed by changing the evaluation scenario accordingly.

B. Insertion Rendering

Insertion rendering involves producing realistic frames of a scene by inserting assets (image fragments; 3D models; etc.) into a target image (variants in [55], [56], [57], [58], [59]). Realistic frames can be produced very fast *if* difficulties presented by lighting, shadows, and geometrical consistency can be managed. Once the scenario is triggered, we render the simulated scenarios and compose them into images in real time. This requires the insertion rendering to be realistic, efficient, and geometrically consistent. To achieve this, we adopt a real-time OpenGL-based rasterization pipeline [60].

We first place the object accurately in the predefined world coordinate, and the camera pose is acquired in real-time through an RTK-INS localization module. The lighting consists of the skybox and the sunlight; parameters are inferred from real-time weather, GPS, and the time of day.

The rendering process is then conducted using a customized physical-based rendering (PBR) shader that follows the split-sum shading model, as described by the equation: $L(\mathrm{x}, \omega_o) = L_a + L_s(\omega_s) f_r(\mathrm{x}, \omega_o, \omega_s)(\omega_s \cdot \mathrm{n})$, where x is the observed point, ω_o is the outgoing ray and ω_s is the incoming ray. $L(\mathrm{x}, \omega_o)$ is observed radiance; L_a is ambient sky color and L_s is the directional sunlight; f_r is the Cook-Torrence reflection model [61]: $f_r = k_d f_d + k_s f_s$, f_d is diffuse reflection under sunlight, and f_s describes specular reflection, which accesses the base color, roughness, and metallic textures of the object to compute specular reflection. k_d and k_s are the ratios of the respective components. The resulting output, as shown in Fig. 3, exhibits a visually appealing surface appearance.

Shadows cast by the inserted objects contribute to the perceived realism. In our framework, a two-pass shadow mapping procedure is applied [62]. The first pass renders a depth buffer from the lighting source to the visible surface, and the second pass renders per-view depth from the camera perspective. The shadowed areas are the pixels at which the two passes have inconsistent depth. Poisson sampling is used to reduce aliasing, and occlusion reasoning is conducted by comparing the rendered and observed depth. Finally, the rendered objects are composited via alpha-channel blending.

C. Evaluation

The performance of an autonomous driving stack is being evaluated in a simulated safety-critical scenario using recorded behaviors. We adopt the evaluation metrics from the CARLA platform [10], which include the collision rate, trip completion time, comfort metric, and goal reaching rate. The collision rate

Fig. 3. Rendering Quality. We evaluate the quality of insertion rendering by comparing reconstructed 3D humans/objects in Sim-on-Wheels with their real-world counterparts under the same pose. The results demonstrate that our real-time insertion rendering can produce realistic, high-fidelity appearances, and cast-shadows. Table III quantitatively measures the sim2real gap.

represents the percentage of scenarios where the ego-vehicle experiences at least one collision, which is determined by checking for overlap between the oriented bounding box of the vehicle and other virtual objects. In addition to ensuring safety, we aim for our autonomous vehicle to be as efficient as possible, which is reflected in the trip completion time. During each run, the trip completion time is measured until the vehicle is within 5 m of a static obstacle, or in other scenarios, within 1.5 m of the end of its planned path. In any scenario, if the vehicle is not able to reach its goal, we penalize that run by recording its time metric as 100 s. For our passenger comfort metric, we record the maximum absolute vehicle acceleration, measuring how hard the vehicle brakes. Finally, the goal reaching rate is the percentage of scenarios where the ego-vehicle reaches its goal. Furthermore, in order to account for real-world uncertainties, we report the mean of all four metrics over multiple runs under different hyper-parameters for one scenario.

IV. Experiments

The goal of the experiments in this section is to address the following three crucial questions: (1) Can the Sim-on-Wheels framework, as proposed, be utilized as a rigorous and comprehensive benchmark for evaluating the performance of various autonomous stacks? (2) Can we empirically validate the authenticity of our simulation? (3) To what extent does the onboard simulation result in an increase in latency?

In this section, we first provide an overview of the hardware platform and the test track used for our experiments. We then benchmark the performance of two self-driving agents in various safety-critical scenarios using the Sim-on-Wheels framework and conduct a comprehensive analysis of their performance. We then conduct an empirical analysis of the sim2real gap at both the perception and the action level, followed by additional discussions of our framework.

A. Real-World Testbed

All of our experiments are carried out on the Polaris GEM e2, a street-legal, two-seater electric vehicle with a top speed of 25 mph. The sensor stack of the vehicle includes a Velodyne-16 LiDAR, a Novatel RTK GNSS+INS unit, a ZED 2 Stereo Camera, and a Delphi ESR 2.5 radar. The vehicle supports drive-by-wire through the PACMod kit. Our experiments utilize the AStuff Spectra 2 [63], an industrial-grade edge computing platform equipped with an NVIDIA A4000 GPU. This computer is connected to a built-in monitor.

The experiments were carried out in a shared testing track facility with a secure testbed area. A designated safety driver and safety lookout were present at all times.

B. Evaluated Autonomous Agents

We subject two distinct autonomous agents for evaluation using the Sim-on-Wheels framework: (1) a modular autonomy stack, and (2) an end-to-end imitation learning stack. These two autonomy stacks were chosen to represent the mainstream approaches for self-driving cars. Note that developing new autonomy methods is not the focus of our letter.

1) Modular Autonomy Agent: Our modular autonomy pipeline takes as input the RGB-D image stream and a coarse planned path. It is composed of four components: detection, tracking, motion prediction, and rule-based longitudinal planning.

Obstacle detection is split into static and dynamic obstacles. Static obstacles are detected via a pretrained foreground segmentation model [64], and dynamic obstacles (pedestrians and cars) are detected via a pretrained instance segmentation model [65]. Each instance mask is converted to a 3D position by unprojecting the pixels and taking the median.

At the tracking stage, greedy matching [66] is performed to associate the latest detected object and existing tracks based on a bird's eye view. We then estimate the state (velocity and position) through a linear motion model. Using these states and the ego-car's current speed and planned trajectory, we predict the positions of each entity at every $0.2\,\mathrm{s}$ step up to $10\,\mathrm{s}$ into the future and identify potential collisions. At each future time step, we search for collisions within a fixed collision radius ($3\,\mathrm{m}$) and travel distance threshold ($5\,\mathrm{m}$). This threshold can be increased to compensate for latency. If a collision is found, we output a desired speed of zero. Otherwise, we output $2\,\mathrm{m/s}$.

2) Imitation Learning (IL) Agent: We train an end-to-end neural controller using behavior cloning. The network takes as

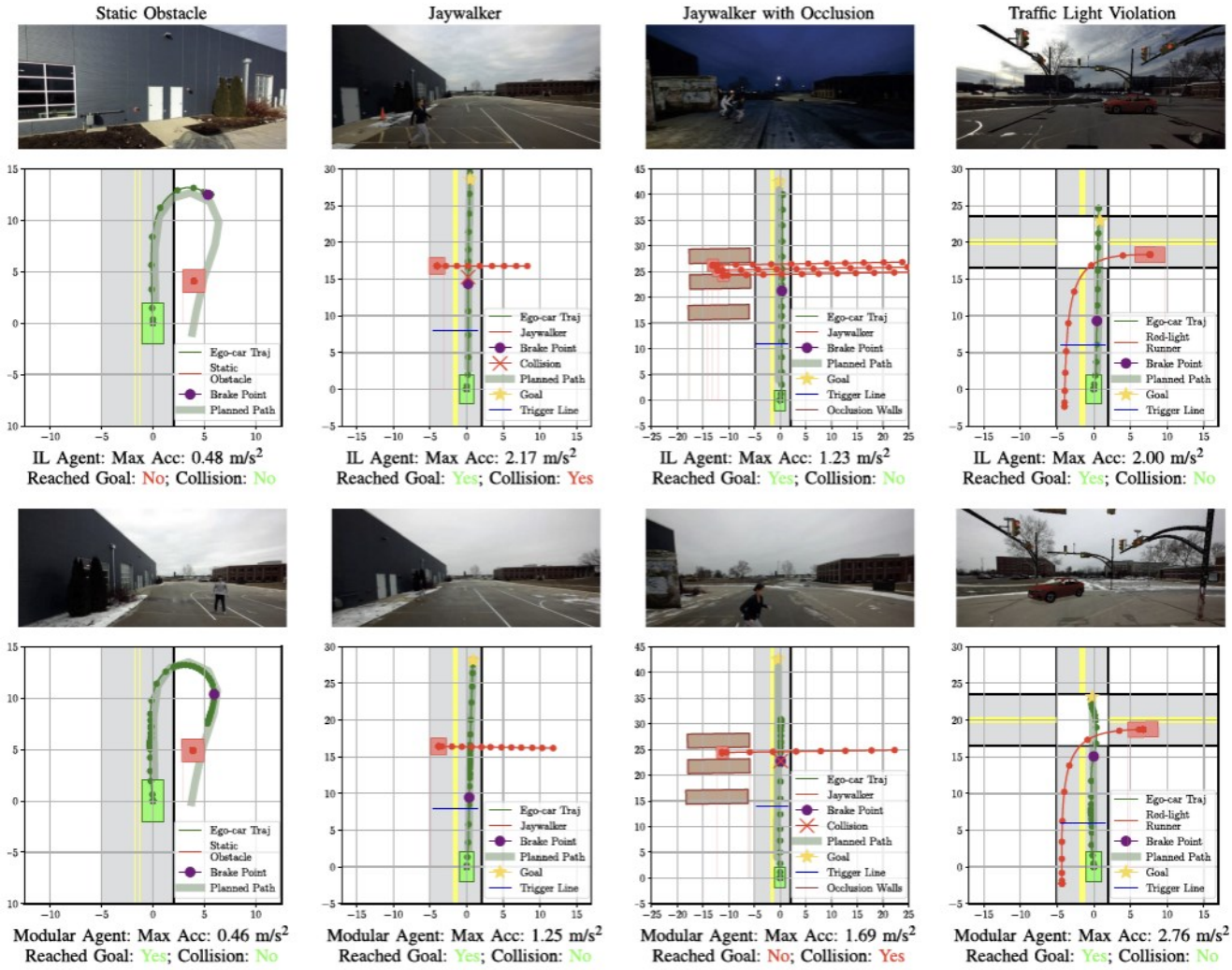

Fig. 4. Qualitative driving results.: We show bird's-eye view layouts of four different scenarios across both agents in ego-vehicle coordinates (meters). The images are captured at the first braking point before approaching an obstacle. The ego-car trajectory points are plotted every two seconds. We can see that the modular agent tends to be more cautious and takes longer to reach the goal, while the imitation learner drives smoothly but brakes too early/late in certain scenarios.

TABLE II
SIM-ON-WHEELS BENCHMARK RESULTS

Agent Type	Static Obstacle				Traffic Light Violation				Jaywalker				Jaywalker with Occlusion			
	Coll. Rate ↓	Trip Time ↓	Max. Acc. ↓	Goal Rate ↑	Coll. Rate ↓	Trip Time ↓	Max. Acc. ↓	Goal Rate ↑	Coll. Rate ↓	Trip Time ↓	Max. Acc. ↓	Goal Rate ↑	Coll. Rate ↓	Trip Time ↓	Max. Acc. ↓	Goal Rate ↑
Modular	**0.0**	53.33	**0.85**	0.7	**0.0**	86.02	**1.28**	0.5	**0.125**	34.35	**1.49**	0.75	0.96	37.81	2.23	0.33
IL	0.33	**27.78**	1.17	**0.78**	0.33	**22.90**	1.74	**1**	0.875	**19.58**	1.66	**0.96**	**0.58**	**29.41**	**1.85**	**1**

Metrics are described in Section III-C. Coll. Rate stands for collision rate. Trip completion time is measured in seconds. Max. Acc. (comfort metric) is measured in m/s².
Findings: The results show that the modular pipeline is better overall at avoiding collisions, but the imitation learning pipeline reaches the goal faster.
The bold values represent the best value for the corresponding metric among all methods compared in that column.

input the last eight frames of speed, position, and RGB image, separated by 0.2 seconds. The controller outputs a continuous brake command to be executed 0.2 s in the future, accounting for the latency of the overall pipeline.

The network consists of a COCO-pretrained FCN backbone [67], [68] and a GRU for temporal aggregation [69]. Brake values are decoded by a multi-layer perceptron.

To train the network, diverse real-world data is collected from human driving where the driver brakes for static objects and jaywalking pedestrians (195 static, 200 jaywalking). The network learns to predict the human brake inputs using a L_1 regression loss, with an Adam optimizer and a learning rate of 5e-4. Weighted sampling is applied to focus on snippets right before brakes. During training, we apply color jittering, random cropping, and dropout regularization ($p = 0.8$).

3) Vehicle Controller: The output from both the modular agent and the IL agent is sent to the same vehicle controller to produce the final vehicle command. The longitudinal direction is controlled by a proportional-integral (PI) speed controller [70]. Meanwhile, the lateral direction is controlled by the Stanley controller utilizing a bicycle model [71].

4) Human Driver Reference: We also benchmark human driver performance within our Sim-on-Wheel system as a reference. The human driver is tasked with navigating along the planned path while avoiding collisions by monitoring the augmented video displayed in real-time. To prevent the human driver from preemptively avoiding obstacles, we randomized the agent's spawn point, forcing the driver to react on the fly.

C. Autonomy Benchmark Results

Table II reports the performance metrics of our driving agents using the Sim-on-Wheels evaluation framework. For each of the two agents, we performed 15 experiments for static obstacles, 3 for traffic light violations, 8 for jaywalker, and 24 for jaywalker with occlusion. The human driver was only evaluated on the jaywalker with occlusion scenario. Qualitative examples of these scenarios are depicted in Fig. 4.

Our results indicate that the modular agent generally takes longer to reach the destination due to false positive detections of obstacles that lead to intermittent braking. Nevertheless, it is capable of safely reaching the goal in most scenarios, except for jaywalker-with-occlusion. This scenario involves a very limited reaction time window, so the agent usually brakes too late. Furthermore, the challenging textures of the occluding walls causes some missed detections.

In contrast to the modular agent, the imitation learning agent does not experience intermittent braking and achieves a shorter time to reach the goal. Furthermore, it performs acceptably well in the traffic intersection scenario, which was not part of the training data. However, the agent generally tends to react late, resulting in more jaywalker collisions. This may be due to latency differences during training and onboard deployment. It is important to note that such drawbacks were not frequently observed during offline validation, highlighting the importance of real-world vehicle-in-the-loop testing.

We also found that the modular agent had a lower max acceleration in all scenario types, except for Jaywalker with Occlusion. Based on an acceleration threshold of $4\,\mathrm{m/s^2}$ for driving comfort suggested by Wang et al. [72], both agents drove comfortably in terms of acceleration in all scenarios.

Note that none of the agents, including the human driver (Collision Rate: 0.08, Trip Completion Time: 31.79 s, Comfort: $1.94\ \mathrm{m/s^2}$, Goal Reaching Rate: 1), achieve a zero collision rate across all scenarios, which highlights the difficulty of our designed scenarios. Fig. 5 visualizes the episode where the human driver failed. It is worth mentioning that many of the test cases, particularly the jaywalking ones, are impractical to test in the real world due to safety concerns and can only be physically evaluated with Sim-on-Wheels.

D. Analysis

Reality Gap Analysis: In Fig 3, we qualitatively assess our insertion rendering by comparing real vs. simulated results. The image pairs appear quite similar overall, demonstrating the realism of the framework. However, some minor differences do exist, such as incomplete 3D reconstructed shapes, slight differences in shaded color, and variations in sunlight intensity, shadow shape, and cloud patterns due to two images being taken in a windy outdoor environment at different times.

Table III reports the quantitatively measured reality gap using the peak signal-to-noise ratio (PSNR), structural similarity index (SSIM) [73], and a learned perceptual similarity metric (LPIPS) [74]. The mean absolute error (MAE) of each pixel is calculated and the percentage of outlier pixels with errors over a threshold of 25.5 under the RGB intensity range $(0, 255)$ were reported. As indicated in Table III , the results show that the virtual object insertions exhibit high fidelity and realism with a low percentage of outlier pixels.

Fig. 5. Human Driving Failure Case: The one episode in which the human driver collided was the variation of Jaywalker with Occlusion scenario that involved multiple fast jaywalkers. The left image is captured at the braking point.

TABLE III
REALITY GAP

	Sensor			Perception
	PSNR ↑	SSIM ↑	LPIPS ↓	mIoU ↑
Sim-on-Wheels	**21.8**	**0.833**	**0.108**	**0.839**
Baseline: No lighting	19.9	0.831	0.143	0.727
Baseline: No PBR	20.1	0.831	0.142	0.727

The reality gap is assessed using three metrics: 1) sensor image fidelity between reality and simulation, 2) mIoU of perception algorithm output 3) similarity of final actions (brake values). We compare with baseline rendering methods which do not perform lighting or physically-based rendering (PBR). Results indicate a small reality gap for sim-on-wheels, validating its efficacy and reliability as an evaluation framework.
The bold values represent the best value for the corresponding metric among all methods compared in that column.

We also evaluate the impact of the reality gap on the performance of our autonomy pipelines. Firstly, we measure the mean Intersection over Union (mIoU) between the static obstacle segmentation network's outputs on real and simulated images. Although the silhouettes of the obstacles match closely, the mIoU is not perfect, which is likely due to the sensitivity of the network to slight variations in color and the background, such as cloud movement. This limitation is a result of the time required to physically arrange obstacles.

To better convey the quality of our rendering, we ablate on two aspects of our insertion rendering: physically-based rendering (PBR) and lighting. Compared to the baselines without lighting or PBR, ours renders better across all metrics.

Additionally, we evaluate the action-level reality gap by comparing agent behavior in real and simulated scenarios. The scenarios were set up to be identical except for the obstacle (real object vs digital twin). We obtained two metrics: an average endpoint offset of 1.09 ± 0.49 meters and a Δ predicted brake (L_1) of 0.18 ± 0.08. Our results show that the modular agent can stop without collision in both real and Sim-on-Wheels experiments. However, the nonzero trajectory discrepancy suggests that the reality gap is not fully closed yet. This may also be due to other factors, such as changes in background illumination as mentioned above.

Generalization to other environments: We can easily adapt to new environments as long as the scenario is compatible with

TABLE IV
SYSTEM RUNTIME BREAKDOWN

	Sim-on-Wheels	Autonomy	ROS	Total	Relative %
Offline	43.5ms	-	-	-	-
Online	65.8ms	-	100.4ms	163.9ms	38.7
Online + IL	73.6ms	101.5ms	106.6ms	281.7ms	21.13
Online + Modular	99.3ms	135.6ms	133.8ms	368.7ms	26.93

IL stands for imitation learning agent, and modular for modular agent. The ROS column refers to the time for processing input sensor data and sending/receiving synced messages across nodes.

Original

Sim-on-Wheels

Fig. 6. Sim-on-Wheels on KITTI-360: Two jaywalkers on a street are simulated on top of KITTI-360 camera data.

the surroundings. As one example, we generalize to KITTI-360 data [75] as shown Fig. 6.

Runtime Analysis: The runtime performance of the Sim-on-Wheels system (Table IV) was evaluated on an onboard computer equipped with a single Intel Xeon(R) E-2278G CPU @3.40 GHz ×16 and a single NVIDIA RTX A4000 GPU. The standalone rendering component of the system processes pre-recorded sensor data with a frame rate of 23 FPS. Upon integration with ROS, the frame rate is impacted by the consumption of system resources by hardware drivers, communication modules, the autonomy agent, and the controller. Nonetheless, the system still can publish rendered images at a minimum frame rate of 10 FPS, even when the autonomy agents run concurrently.

Our results show that the end-to-end latency from the camera capture time to vehicle command increases by 21–27% with the integration of Sim-on-Wheels, which could be mitigated by equipping the vehicle with two onboard GPUs. Additionally, both autonomy agents can predict future states and actions to compensate for such latency.

Limitations: As shown in Table IV, ROS consumes a significant amount of time. We plan to optimize node communication to further reduce the ROS latency. In addition, despite our use of strong insertion techniques, the domain gap between simulation and real world is challenging to close completely. Another future direction is to marry Sim-on-Wheels with robust verification techniques to provide theoretical guarantees about performance in the real world.

V. CONCLUSION

We propose Sim-on-Wheels, a vehicle-in-the-loop framework for evaluating the performance of autonomous vehicles in real-world scenarios in a safe and realistic manner. To the best of our knowledge, Sim-on-Wheels is the first framework of its kind to support the integration of simulation and real-world testing practices for the safe and realistic evaluation of autonomous vehicles. Our results demonstrate the versatility and reliability of Sim-on-Wheels as a framework for evaluating various agents. To further support the research and development of autonomous driving, we will open-source Sim-on-Wheels to the community and establish a safe, closed-loop, end-to-end, real-world benchmark.

ACKNOWLEDGMENT

The authors are grateful for the support of the Center for Autonomy and John M. Hart. We thank Wei-Chiu Ma and Jason Ren for proofreading.

REFERENCES

[1] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.
[2] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2446–2454.
[3] M. Cordts et al., "The cityscapes dataset for semantic urban scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 3213–3223.
[4] D. A. Pomerleau, "ALVINN: An autonomous land vehicle in a neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1988, pp. 305–313.
[5] K. D. Kusano, K. Beatty, S. Schnelle, F. Favaro, C. Crary, and T. Victor, "Collision avoidance testing of the Waymo automated driving system," 2022, *arXiv:2212.08148*.
[6] "Waymo, Cruise dominate AV testing," 2023. Accessed: Feb. 04, 2023. [Online]. Available: https://www.eetimes.com/waymo-cruise-dominate-av-testing/
[7] "TRC - transportation research center," 2019. Accessed: Feb. 04, 2023. [Online]. Available: https://www.trcpg.com
[8] "Illinois autonomous and connected track," 2022. Accessed: Feb. 04, 2023. [Online]. Available: https://ict.illinois.edu/i-act/testing-arena
[9] S. Thrun et al., "Stanley: The robot that won the DARPA grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006.
[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, 2017, pp. 1–16.
[11] "NVIDIA DRIVE Sim," 2021. Accessed: Feb. 04, 2023. [Online]. Available: https://developer.nvidia.com/drive/drive-constellation
[12] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Proc. Field Serv. Robot.: Results 11th Int. Conf.*, 2018, pp. 621–635.
[13] "Unreal engine simulation for automated driving," 2023. Accessed: Feb. 04, 2023. [Online]. Available: https://www.mathworks.com/help/driving/ug/3d-simulation-for-automated-driving.html
[14] S. Manivasagam et al., "LiDARsim: Realistic LiDAR simulation by leveraging the real world," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11167–11176.
[15] Y. Chen et al., "GeoSim: Realistic video simulation via geometry-aware composition for self-driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 7230–7240.
[16] S. Tan, K. Wong, S. Wang, S. Manivasagam, M. Ren, and R. Urtasun, "SceneGen: Learning generate realistic traffic scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 892–901.
[17] A. Amini et al., "Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2022, pp. 2419–2426.
[18] J. Wang et al., "CADSim: Robust and scalable in-the-wild 3D reconstruction for controllable sensor simulation," *Proc. Conf. Robot Learn.*, vol. 205, pp. 630–642, 2022.
[19] Y. Shen, W.-C. Ma, and S. Wang, "SGAM: Building a virtual 3D world through simultaneous generation and mapping," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 22090–22102.
[20] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian, "Mixed reality for robotics," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 5382–5387.
[21] M. F. Drechsler, J. B. Peintner, G. Seifert, W. Huber, and A. Riener, "Mixed reality environment for testing automated vehicle and pedestrian interaction," in *Proc. 13th Int. Conf. Automot. User Interfaces Interactive Veh. Appl.*, 2021, pp. 229–232.

[22] T. Genevois, J.-B. Horel, A. Renzaglia, and C. Laugier, "Augmented reality on LiDAR data: Going beyond vehicle-in-the-loop for automotive software validation," in *Proc. IEEE Intell Veh. Symp.*, 2022, pp. 971–976.

[23] C. Hildebrandt and S. Elbaum, "World-in-the-loop simulation for autonomous systems validation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2021, pp. 10912–10919.

[24] S. Hallerbach, Y. Xia, U. Eberle, and F. Köster, "Simulation-based identification of critical scenarios for cooperative and automated vehicles," *SAE Int. J. Connected Autom. Veh.*, vol. 1, no. 2, pp. 93–106, Apr. 2018.

[25] H et al., "Continuous cooperation reduced & remote adaptable prototype-in-the-loop," in *Proc. IMAGinE*, 2022. [Online]. Available: https://www.imagine-online.de/en/news-events/final-event-agenda-1.html

[26] J. Daudelin, G. Jing, T. Tosun, M. Yim, H. Kress-Gazit, and M. Campbell, "An integrated system for perception-driven autonomy with modular robots," *Sci. Robot.*, vol. 3, no. 23, 2018, Art. no. eaat4983.

[27] A. Tampuu, T. Matiisen, M. Semikin, D. Fishman, and N. Muhammad, "A survey of end-to-end driving: Architectures and training methods," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 4, pp. 1364–1384, Apr. 2022.

[28] J. Levinson et al., "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE Intell. Veh. Symp.*, 2011, pp. 163–168.

[29] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun, "Learning to localize using a LiDAR intensity map," in *Proc. Conf. Robot Learn.*, vol. 87, pp. 605–616, 2018.

[30] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1907–1915.

[31] Z. Li et al., "BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 1–18.

[32] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 641–656.

[33] A. Kelly and B. Nagy, "Reactive nonholonomic trajectory generation via parametric optimal control," *Int. J. Robot. Res.*, vol. 22, no. 7-8, pp. 583–601, 2003.

[34] E. F. Camacho and C. B. Alba, *Model Predictive Control*. Berlin, Germany: Springer, 2013.

[35] M. A. Johnson and M. H. Moradi, *PID Control*. Berlin, Germany: Springer, 2005.

[36] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2005, pp. 739–746.

[37] W. Zeng et al., "End-to-end interpretable neural motion planner," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8660–8669.

[38] M. Bojarski et al., "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.

[39] M. Bojarski et al., "Explaining how a deep neural network trained with end-to-end learning steers a car," 2017, *arXiv:1704.07911*.

[40] W. Zeng, S. Wang, R. Liao, Y. Chen, B. Yang, and R. Urtasun, "DSDNet: Deep structured self-driving network," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 156–172.

[41] N. T. S. Board, "Collision between vehicle controlled by developmental automated driving system and pedestrian," *NTSB Accident Rep. NTSB/HAR-19/03 PB2019-101402*, 2019.

[42] S. Höfer et al., "Sim2Real in robotics and automation: Applications and challenges," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 2, pp. 398–400, Apr. 2021.

[43] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 3803–3810.

[44] O. M. Andrychowicz et al., "Learning dexterous in-hand manipulation," *Int. J. Robot. Res.*, vol. 39, no. 1, pp. 3–20, 2020.

[45] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "TrafficSim: Learning to simulate realistic multi-agent behaviors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 10400–10409.

[46] L. Feng, Q. Li, Z. Peng, S. Tan, and B. Zhou, "TrafficGen: Learning to generate diverse and realistic traffic scenarios," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 3567–3575.

[47] Q. Sun, X. Huang, B. C. Williams, and H. Zhao, "InterSim: Interactive traffic simulation via explicit relation modeling," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 11416–11423.

[48] T. Bokc, M. Maurer, and G. Farber, "Validation of the vehicle in the loop: A milestone for the simulation of driver assistance systems," in *Proc. IEEE Intell. Veh. Symp.*, 2007, pp. 612–617.

[49] A. Albers and T. Düser, "Implementation of a vehicle-in-the-loop development and validation platform," in *Proc. FISITA World Automot. Congr.*, 2010, pp. 173–182.

[50] M. F. Drechsler, V. Sharma, F. Reway, C. Schütz, and W. Huber, "Dynamic vehicle-in-the-loop: A novel method for testing automated driving functions," *SAE Int. J. Connect. Autom. Veh.*, vol. 5, no. 4, pp. 367–380, 2022.

[51] W. G. Najm et al., "Pre-crash scenario typology for crash avoidance research," NHTSA, Washington, D.C. USA, Tech. Rep. DOT-VNTSC-NHTSA-06-02, 2007.

[52] "Sketchfab search," 2022. Accessed: Feb. 04, 2023. [Online]. Available: https://sketchfab.com

[53] "Realityscan," 2023. Accessed: Feb. 04, 2023. [Online]. Available: www.capturingreality.com/

[54] Adobe, "Mixamo," 2021. Accessed: Feb. 04, 2023. [Online]. Available: https://www.mixamo.com

[55] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi, "Photo clip art," *ACM Trans. Graph.*, vol. 26, no. 3, p. 3, 2007.

[56] Z. Liao, K. Karsch, H. Zhang, and D. Forsyth, "An approximate shading model with detail decomposition for object relighting," *Int. J. Comput. Vis.*, vol. 127, pp. 22–37, 2018.

[57] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem, "Rendering synthetic objects into legacy photographs," in *Proc. SIGGRAPH Asia*, 2011, pp. 1–12.

[58] K. Karsch et al., "Automatic scene inference for 3D object compositing," *ACM Trans. Graph.*, vol. 33, no. 3, pp. 1–15, 2014.

[59] Z. Liao, A. Farhadi, Y. Wang, I. Endres, and D. Forsyth, "Building a dictionary of image fragments," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3442–3449.

[60] S. Dombi, "Moderngl, high performance Python bindings for OpenGL 3.3.," 2019. Accessed: Feb. 04, 2023. [Online]. Available: https://github.com/moderngl/moderngl

[61] R. L. Cook and K. E. Torrance, "A reflectance model for computer graphics," *ACM Trans. Graph.*, vol. 1, no. 1, pp. 7–24, 1982.

[62] L. Williams, "Casting curved shadows on curved surfaces," in *Proc. 5th Annu. Conf. Comput. Graph. Interactive Techn.*, 1978, pp. 270–274.

[63] "Astuff spectra 2," 2022. Accessed: Feb. 04, 2023. [Online]. Available: https://autonomoustuff.com/products/astuff-spectra-2

[64] J. Xu, Z. Xiong, and S. P. Bhattacharyya, "PIDNet: A real-time semantic segmentation network inspired from PID controller," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 19529–19539.

[65] T. Cheng et al., "Sparse instance activation for real-time instance segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 4433–4442.

[66] M. Dyer, A. Frieze, and B. Pittel, "The average performance of the greedy matching algorithm," *Ann. Appl. Probability*, vol. 3, pp. 526–552, 1993.

[67] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.

[68] T. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[69] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*.

[70] L. Zheng, "A practical guide to tune of proportional and integral (PI) like fuzzy controllers," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, 1992, pp. 633–640.

[71] P. Polack, F. Altché, B. d'Andréa Novel, and A. d. L. Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 812–818.

[72] P. Wang, L. Wang, Y. Li, and W. Guo, "Improved cooperative collision avoidance (CCA) model considering driver comfort," *Int. J. Automot. Technol.*, vol. 16, pp. 989–996, 2015.

[73] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[74] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 586–595.

[75] Y. Liao, J. Xie, and A. Geiger, "KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3292–3310, Mar. 2023.