CiThruS2: Open-source Photorealistic 3D Framework for Driving and Traffic Simulation in Real Time

Emilian Galazka, Teo T. Niemirepo, and Jarno Vanne *Member, IEEE*

***Abstract*—The automotive and transport sector is undergoing a paradigm shift from manual to highly automated driving. This transition is driven by a proliferation of advanced driver assistance systems (ADAS) that seek to provide vehicle occupants with a safe, efficient, and comfortable driving experience. However, increasing the level of automation makes exhaustive physical testing of ADAS technologies impractical. Therefore, the automotive industry is increasingly turning to virtual simulation platforms to speed up time-to-market. This paper introduces the second version of our open-source See-Through Sight (CiThruS) simulation framework that provides a novel photorealistic virtual environment for vision-based ADAS development. Our 3D urban scene supports realistic traffic infrastructure and driving conditions with a plurality of time-of-day, weather, and lighting effects. Different traffic scenarios can be generated with practically any number of autonomous vehicles and pedestrians that can be made to comply with dedicated traffic regulations. All implemented features have been carefully optimized and the performance of our lightweight simulator exceeds 4K (3840 × 2160) rendering speed of 60 frames per second when run on NVIDIA GTX 1060 graphics card or equivalent consumer-grade hardware. Photorealistic graphics rendering and real-time simulation speed make our proposal suitable for a broad range of applications, including interactive driving simulators, visual traffic data collection, virtual prototyping, and traffic flow management.**

Keywords — Advanced driver-assistance systems (ADAS), driving simulation, traffic imaging, photorealism, open-source software

I. Introduction

The rapid emergence of *advanced driver-assistance systems* (*ADAS*) has revolutionized the whole automotive and transport sector and the evolution continues gradually towards fully autonomous driving [1]. Modern vehicles are increasingly equipped with a broad range of passive and active ADAS technologies that protect and prevent vehicle occupants from accidents by sensing the environment and detecting other road users with a camera, RADAR, LiDAR, and other advanced-sensing technologies [2]. The current trend is towards more holistic situational awareness that is further leveraged by means of on-board sensor fusion techniques and *vehicle-to-everything* (*V2X*) communication with other traffic participants [3]. Inside the cabin, advanced *human-machine interface* (*HMI*) technologies [4] are actively being developed to adapt to the new ways of interaction between the driver and vehicle.

This paper is part of the NEWCONTROL project that has received funding within the ECSEL JU in collaboration with the European Union's H2020 Framework Programme (H2020/2014-2020) and National Authorities, under grant agreement 826653.

The authors are with Ultra Video Group, Tampere University, FI-33014 Tampere University, Finland (e-mails: {emilian.galazka, teo.niemirepo, jarno.vanne} @tuni.fi).

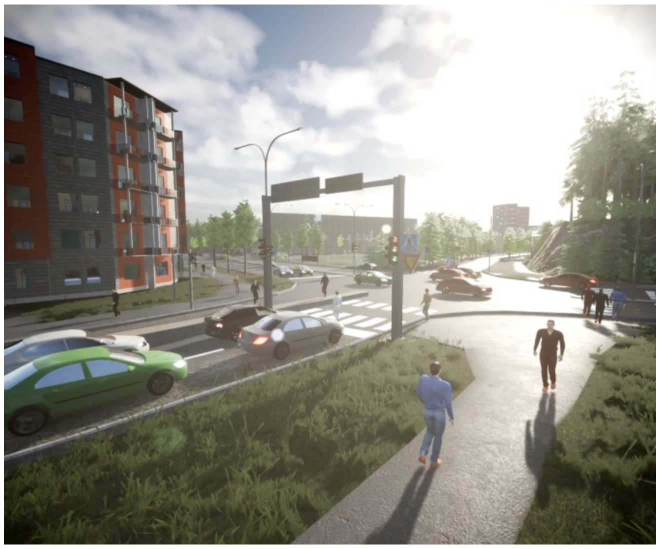
Figure 1. Snapshot of the CiThruS2 simulation environment.

Inevitably, the design complexity of ADAS increases together with the level of automation due to the explosive growth of automotive software, *electrical/electronic* (*E/E*) components, and sensor modalities [5]. Changing the role of the driver from manual execution to supervision also leads to the proliferation of test cases, which makes expensive and time-consuming physical testing with real vehicles impracticable. In addition, meeting the requirements of various safety standards and regulations puts additional pressure on functional verification and validation. Many traffic scenarios, such as accidents, can also be extremely difficult and even dangerous to demonstrate and reproduce in real-world testbeds.

To this end, developing safe, fail-operational, and cost-effective ADAS technologies call for simulation environments where different functionalities can be virtually tested, iterated, and verified under different parameter settings and traffic scenarios before being introduced into the actual systems. Key industry players have also released commercial simulation tools on the market, such as Google's Waymo [6], NVIDIA Drive Constellation [7], and LG Autonomous Driving Simulator [8], but they are not freely available to users, developers, regulators, or other stakeholders, and thus they will not be considered in this paper.

Over the past three decades, several noteworthy open-source virtual simulation platforms [9]-[16] have also been announced, including SIRCA [10], TORCS [11], CARLA [12], and AirSim [13] that are probably the most well-known solutions in the field. However, they all fall short of combining real-time speed, photorealism, realistic layout of the

TABLE I: FEATURES OF THE EXISTING AND PROPOSED OPEN-SOURCE SIMULATION FRAMEWORKS

Platform	License	Year	Photorealistic	Realistic Layout	Lightweight	Weather	Time-of-Day	Traffic	Manually Driveable
SIRCA [10]	NA	1996	No	Yes	Yes	No	No	Yes	No
TORCS [11]	GPL	2014	No	Yes	Yes	No	No	No	Yes
CARLA [12]	MIT/CC-BY	2017	Yes *	Yes *	No	Yes	Yes	Yes	Yes
AirSim [13]	MIT	2017	Yes *	Yes *	No	Yes	Yes	No	Yes
CoInCar-Sim [14]	NA	2018	No	Yes	No	No	No	Yes	No
V. R. Aparow et al. [15]	NA	2019	No	No *	NA	No	No	No	No
A. AbdelHamed et al. [16]	NA	2019	No	No *	NA	No	No	No **	Yes
CiThruS1 [17]	MIT	2019	No	No	Yes	Yes	Yes	Yes	Yes
Ours (CiThruS2)	**MIT**	**Current**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

NA, not mentioned
*scene dependent
**only pedestrians and limited vehicles

environment, all traffic participants, and other environmental effects into a single solution.

This paper gives an overview of our open-source approach called *See-Through Sight* (*CiThruS*) simulation framework. The first version of our environment [17] was published in 2019. It was derived from Windridge City Asset [18], but its road infrastructure and geometry (lane widths, curves, etc.) were found inappropriate for realistic driving simulation. These restrictions motivated us to build a completely new urban digital twin with realistic traffic infrastructure and thereby reduce the gap between virtual and real-world testing.

The new environment, a.k.a. CiThruS2 is modeled after Hervanta, a suburb of the city of Tampere, Finland. Fig. 1 depicts a snapshot of our virtual 3D urban scene. It is created using the realistic *physically based rendering* (*PBR*) [19] and it can be populated with practically any number of autonomous vehicles and pedestrians. The environment also supports a multitude of driving conditions with versatile and easily adjustable time-of-day lighting and weather effects. Time changes dynamically during the simulation, allowing the user to experience the entire 24-hour period in a single sitting. Weather conditions such as rain, snow, or fog can be toggled manually to experience specific situations. This broad spectrum of features makes our simulation software a multipurpose virtual testbench for a myriad of traffic scenarios.

The proposed simulation environment is distributed under the MIT open-source license on GitHub at

github.com/ultravideo/CiThruS2

It is built in Unreal Engine 4 using the C++ language. The system is carefully optimized for real-time simulation on high-end consumer-grade hardware. The chosen design approach makes our environment feasible for 1) interactive *driver-in-the-loop* (*DIL*) simulators [20] where immediate responses and emotional states of the cabin occupants are of the essence; 2) visual data collection (ground truth) for training neural networks and related vehicular vision techniques [21]; 3) testing and validation of vision-based ADAS algorithms in real-time *software-in-the-loop* (*SIL*) and *hardware-in-the-loop* (*HIL*) simulations [9]; or 4) visualization of transportation planning and traffic flow control schemes [22] in a realistic urban road network.

The rest of the paper is organized as follows. Section 2 characterizes the existing open-source driving simulation platforms. Section 3 provides an overview of the proposed CiThruS2 simulation framework and the implemented traffic management system. The natural and built environment is described in Section 4 with weather and lighting effects in Section 5. Section 6 evaluates the simulation performance and summarizes the implemented optimizations. The main applications of our proposal are discussed in Section 7. Finally, Section 8 concludes the paper.

II. RELATED WORK

Table 1 summarizes the main features of the most well-known open-source platforms for traffic and driving simulation. A fully-fledged virtual testbed calls for realistic traffic infrastructure and driving conditions that are reflective of the real world. Therefore, the environments are particularly characterized in terms of photorealistic graphics, realistic layout, other traffic participants, and environmental effects. The perceived realism of the environments is also crucial for visual data collection and traffic flow visualization.

On the other hand, high rendering speed is paramount for realistic DIL, SIL, and HIL simulations, especially in high-speed traffic scenarios, where an ADAS system should take over control or assist the driver on the fly. Simulating such scenarios is not possible unless the system is capable of outputting video feed at high frame rates or perform the simulation at slower-than-reality frame rates. Rendering multiple cameras in real-time is another reason for a lightweight simulation environment. For example, in typical situations, a car might have dozens of cameras and rendering them all while keeping the environment usable is difficult to achieve in real-time.

CARLA [12] and AirSim [13] platforms are the closest approaches to our proposal in terms of their feature offering. CARLA has been designed for development, training, and validation of autonomous driving systems and related self-driving algorithms. However, it is built on compute-intensive machine-learning algorithms with a lot of detailed features, so it is computationally heavy and thereby falls behind real-time performance. AirSim is primarily meant for aerial vehicle testing and evaluation, and as such falls short on the overall visual fidelity of close-to-ground details. It also excludes other traffic participants, as is the case with TORCS [11] and the environments introduced by V. R. Aparow et al. [15] and A. AbdelHamed et al. [16]. On the other hand, SIRCA [10] and CoInCar-Sim [14] include other traffic participants, but they are lacking in their overall graphical fidelity.

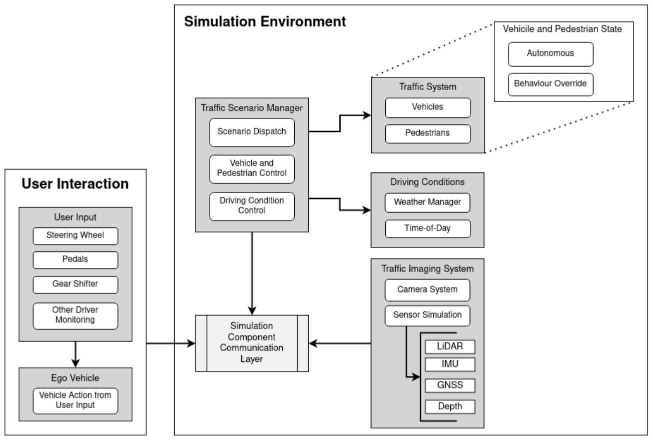

Figure 2. Block diagram of the CiThruS2 simulation framework.

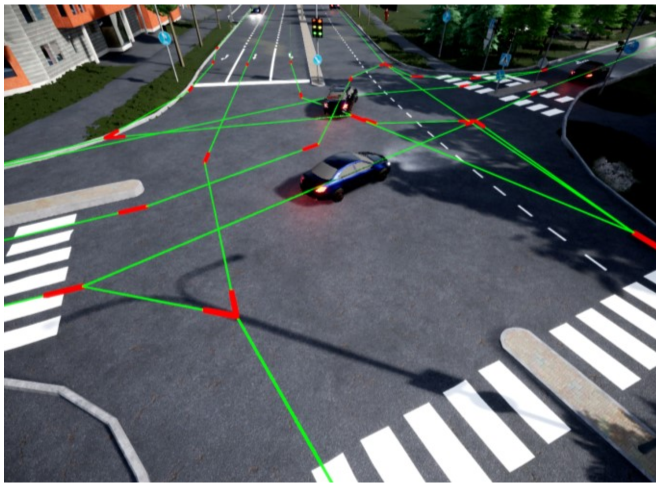

Figure 3. Vehicle pathfinding node network.

III. CiThruS2 Simulation Framework

Fig. 2 depicts a conceptual block diagram of the CiThruS2 simulation framework. It consists of two logical entities called the *user interaction* and *simulation environment*. The latter is further divided into the four main components: *Traffic Scenario Manager* (*TSM*), *Traffic System*, *Driving Conditions*, and *Traffic Imaging System*. The simulation environment can serve as a virtual testbench for various driving and traffic simulation scenarios, such as interactive driver monitoring, virtual vehicle testing, visual traffic data collection, and traffic flow management and visualization.

A. Vehicle under Test

The system accepts user input from a steering wheel, pedals, gear shifter, or more advanced HMI. The input is used to steer and control the vehicle under test, a.k.a. the ego vehicle.

The framework supports holistic virtual monitoring of the driver and other cabin occupants by providing the means of capturing the scene from various camera positions, such as first-person driver's -view, bird's-eye view, third-person's view, or another relevant camera angle. Being able to see and depict the driver's field-of-view is also essential in driving monitoring.

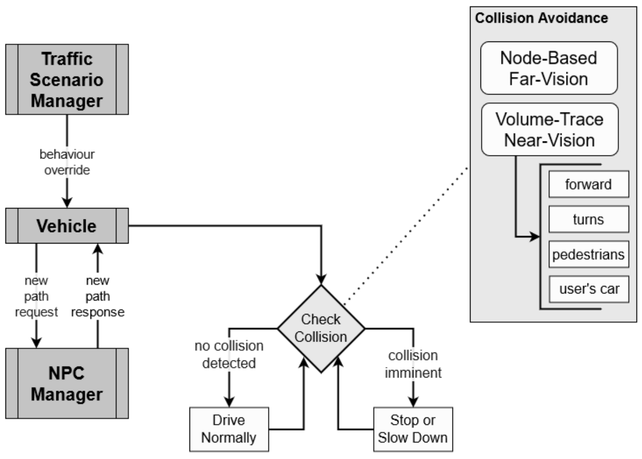

Figure 4. The operating principle of the vehicles.

The TSM is responsible for capturing video and sensor footage during the simulation. Different 2D, stereo, and 360-degree virtual cameras can be separately mounted on the desired positions of the vehicle bodies or practically anywhere in the environment. The environment can also be used to simulate various other sensors technologies applied in modern vehicles and traffic systems, such as LiDARs, *inertial measurement units* (*IMU*), and GNSS, to name a few.

B. Other Vehicles

The environment also provides a set of *Non-Player-Characters* (*NPCs*) for populating the scenes with vehicles and pedestrians. Currently, it supports a diverse selection of vehicle *NPCs*, such as sedans, trucks, and vans. These 3D models were crafted by us, using existing commercial vehicles as reference. In addition, some free ready-made assets with appropriate licenses were used.

The vehicles utilize a lightweight location-node based waypoint system for pathfinding. It allows us to clearly determine the allowed roads for vehicles and keep the simulation computationally lightweight. The debugging view of the node network is shown in Fig. 3.

To account for a human driving among autonomous vehicles, the vehicles react dynamically to the traffic and the environment around them. The operation principle of the vehicles can be seen in Fig. 4. By default, they operate in autonomous collision-avoidance mode and follow a randomly generated path. A collision is detected either 1) through the node-based far-vision approach, where the vehicle queries the nodes in front of it for possible collision-imminent vehicles; or 2) the volume-trace near-vision approach, where the vehicle checks whether there is a car on its path. The volume-traced based collision detection is also used to prepare for turns, intersections, and other complicated traffic situations.

The other operation mode for vehicles is the traffic scenario override mode, where the TSM overrides the behavior of all relevant vehicles and guides them according to the scenario-related rules. In this mode, all non-relevant vehicles are routed away for lower complexity. The vehicles also do not check for collisions with each other, as their movements are overridden by the TSM, which guarantees a no-collision flow

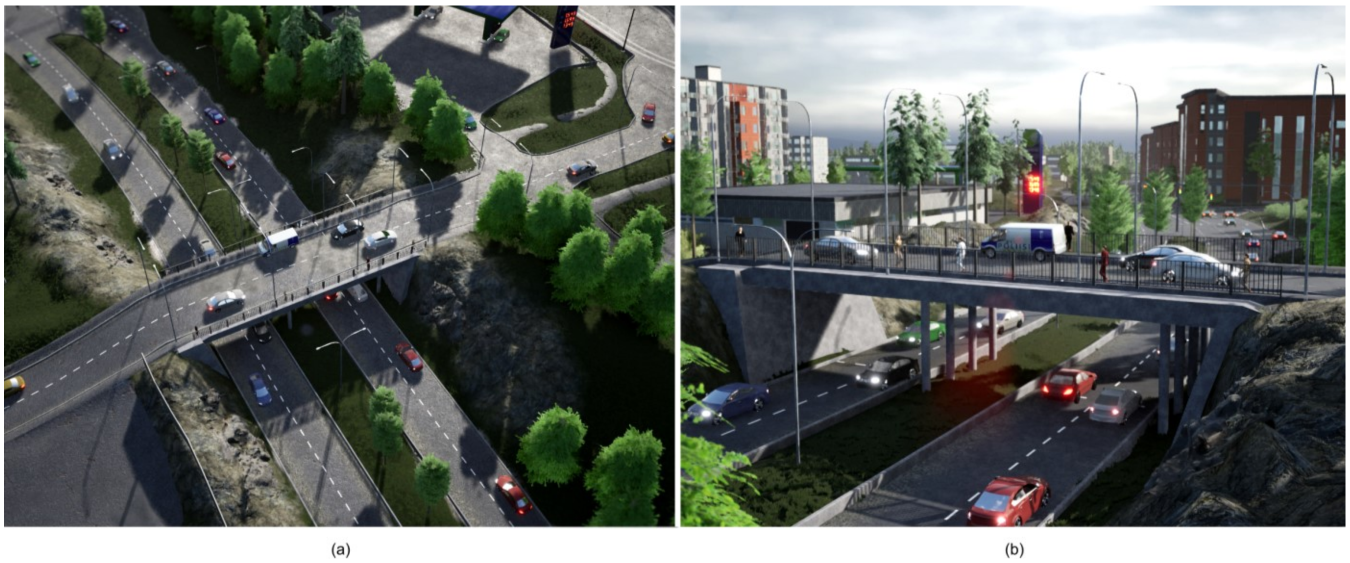
Figure 5. An example case of traffic flow in the environment. (a) A bird's-eye view. (b) An elevated third-person view.

of traffic. The other vehicles still react normally to the vehicle under test.

C. Pedestrians

Pedestrians are spawned randomly, and they walk around a chosen path. The pedestrians react to traffic lights and can cross the road accordingly. Additionally, the system can trigger random events in which the pedestrians run onto the road or decide to cross on the red light. All human characters have been made using the open-source MakeHuman [23] middleware.

D. Traffic Scenarios

The simulation environment maintains validity and stochastic flow of traffic by dynamically creating different traffic scenarios such as parking, overtaking, traffic accidents, and traffic jams. Unpredictable traffic conditions are key to monitor the human driver. TSM produces, dispatches, and manages the traffic scenarios and controls the vehicles, pedestrians, and driving conditions. The driving condition component takes care of weather conditions, lighting, and the time-of-day effects.

E. Traffic Flow

A common use case for traffic imaging systems is monitoring the flow of traffic in order to test and validate traffic flow control systems, or to create typical law-enforcement speed monitoring situations (Fig. 5).

The TSM can be used to direct the traffic flow to a specific area in the environment. The routes and paths of the vehicles and pedestrians are managed by the traffic system that seeks to maintain a smooth flow of traffic and verify that road users comply with traffic rules.

IV. Geospatial Data

The CiThruS2 virtual environment covers an area of roughly 9 km^2 (3 km × 3 km) of Hervanta, Finland. The development time of this Hervanta scene was significantly reduced by utilizing data from external sources, such as photos, topological maps, and satellite imagery. This approach also ensures the model's accuracy and consistency with its real-life counterpart.

At a high level, the graphical side of the environment was made from four layers: terrain, roads, buildings, and foliage. Using separate layers simplifies the development and allows easier environment characterization for vehicular vision algorithms.

A. Terrain

The terrain layer is derived from a heightmap [24], which was acquired with *terrain.party* [25] from OpenStreetMap [26] and National Land Survey of Finland's Topographic Database [27]. Although the acquired data was mostly accurate and provided a good starting point for terrain creation, some manual adjustment was made to achieve better correspondence between real and virtual worlds.

B. Roads

The placement and size of the roads, sidewalks, and parking lots were extracted using OpenStreetMap [26]. The road layout and placement were accurate and no further topological adjustment was needed. The model exported from OpenStreetMap also included locations and basic shapes of most of the buildings, which were replaced with more accurate and custom models.

C. Buildings

The buildings were created manually using reference images, from both ground-level and aerial footage. The manual approach was chosen over, e.g., aerial photogrammetry or laser-scanning [28], as it allowed for more consistent and precise control over the style and shape of the buildings. Having accurate buildings and textures is of utmost importance to computer vision algorithms [21].

The development time was significantly sped up by reusing most used “building blocks” or elements in buildings with similar features. This was especially important for constructs commonly not visible from the ground level.

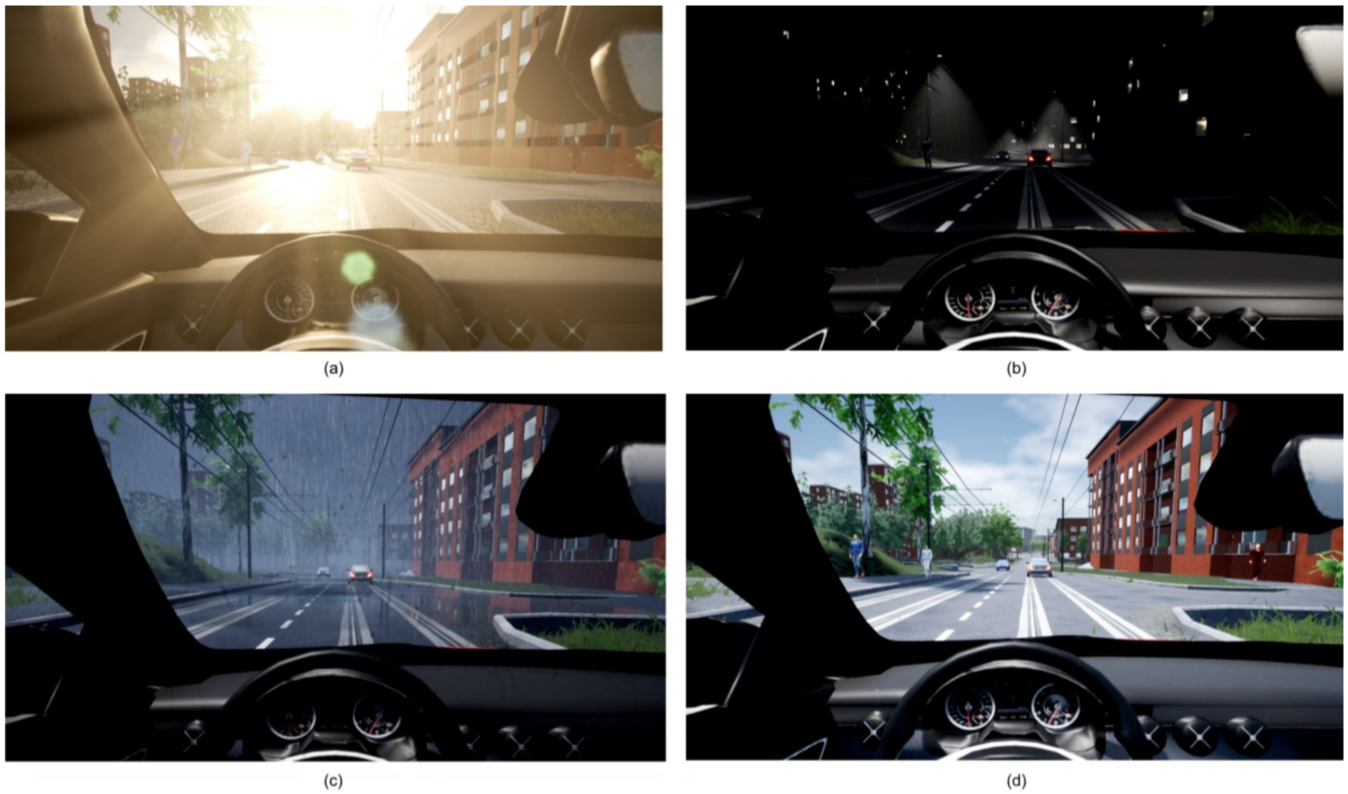
Figure 6. First-person views of different driving conditions. (a) Morning. (b) Night. (c) Daytime rain. (d) Afternoon.

D. Foliage

The city of Hervanta is surrounded by an ample amount of foliage and forestry, so well-optimized grass, trees, and other vegetation assets were crucial in creating a convincing simulation environment. In many computer graphics applications, foliage can take the largest part of the frame budget. Rendering transparency requires that the depth buffer is sorted once for every overlapping pixel with a non-unit alpha value in the fragment shader, which results in transparency overdraw. As a result, the performance is severely impacted in dense forest areas. Classic *Level-of-Detail* (*LOD*) approaches can limit the frame-time impact of foliage, but ultimately merely reducing the triangle count of individual trees is not sufficient.

Another approach for optimizing distant trees are *billboard sprites* [29], which render the tree image into a simple quad-plane with 4 vertices and 2 triangles. Although this is effective in optimizing the scene, the effect it creates is easily distinguishable and does not maintain the believability of the simulation environment. Billboards also increased the amount of layered transparent objects, especially in forest environments. It made the transparency overdraw issue worse and did not help with the performance by a significant margin.

The solution for maintaining a large volume of trees with high visual fidelity is called impostors [29]. Instead of creating this functionality from scratch, a ready-made implementation for Unreal Engine 4 [30] was used. This allows the simulation to contain over 200 000 trees inside the view-frustum without affecting the performance significantly.

V. Weather and Lighting Effects

Various weather conditions such as rain, snow, or fog are required to simulate real driving conditions. All these weather effects are also accompanied by lighting, which is arguably one of the most important aspects contributing to a photorealistic visual style. Therefore, it is critical to simulate light as realistically as possible. Lighting conditions also have a significant impact on not only the driver, but camera systems as well.

It was crucial to include as many possible lighting variations as the driver might experience on the road. Conditions like harsh evening lighting, overcast sky, or night are tackled differently by a driver, and they thereby come with their unique set of difficulties. To produce realistic lighting, we employed the following effects: a realistic day and night cycle, ambient lighting, strategically placed artificial lights, and a set of screen-space effects.

A. Volumetric Fog

Unreal Engine 4 [31] comes with a built-in system for simulating volumetric fog. In our simulation, it is turned off by default during the day as it has a minimal impact on the visuals alone. However, it creates the effect seen in Fig. 6 (a) when coupled with additional effects such as light shaft occlusion [32] and light shaft bloom [32].

However, volumetric fog is enabled during the night and rain situations. During the night, it provides a light scattering effect that becomes visible under streetlights and in front of headlights, as visualized in Fig. 6 (b). During rainfall, the fog becomes denser and falls closer to the surface in order to

TABLE II: PERFORMANCE COMPARISON ON AMD RADEON 6900 XT GPU

Platform	Scene	Rendering speed	Frame time
CARLA [12]	Town10HD	48 fps	20.8 ms
AirSim [13]	AirSimNH	65 fps	15.0 ms
Ours (CiThruS2)	**Hervanta**	**140 fps**	**7.1 ms**

imitate rain droplets scattering above the ground and add depth to the atmosphere as illustrated in Fig. 6 (c). The afternoon sky of the same area is shown in Fig. 6 (d) for comparison.

B. Rain

The rain droplet effect was implemented using a post-processing material [33], which imitates rain streaks or splashes depending on the direction of the camera. Raindrop ripples and streaks also appear on selected objects, such as car and building windows, roads, and most of the traffic control objects.

C. Snow

A realistic snow effect was implemented similarly to the rain; a particle system was responsible for the falling snowflakes. Additionally, in winter conditions the trees in the simulation lose their leaves and snow starts to pile up on flat surfaces, such as the ground, roads, and the tops of the buildings. Snow depth was imitated using a vertex displacement shader.

D. Day and Night Cycle

The day and night cycle simulates the natural progress of time in the simulation. The system positions the sun in the sky, specifies the hour of the day and is responsible for chronologically accurate lighting. The color and quality of the light vary throughout the day, which is also addressed.

E. Ambient Lighting

The environment lighting is a simulated sunlight during the day, a simulated moonlight during the night, and a skylight achieved with image-based lighting techniques [34]. The colors of these lights are set dynamically by the day and night system.

F. Artificial Lights

The simulation uses the deferred shading pipeline [35]. It was chosen over the other common shading pipeline, the forward renderer [36], because a single light imposes a virtually non-existent impact on performance. This allows the simulation to have a practically infinite number of lights that contribute to the scene illumination at the same time. The realism of the effect is amplified during the night, when all streetlights are illuminated and the vehicles in the traffic system have their head- and backlights turned on.

G. Screen-Space Effects

The simulation makes use of screen-space effects, such as screen-space global illumination [37] and screen-space reflections [38]. Although these techniques can produce limited effects, the lighting or reflections are only influenced by currently visible objects on the screen. In addition, the effect is accurate in relation to its computational complexity.

TABLE III: MAIN OPTIMIZATION TECHNIQUES

Feature	Optimization techniques
All geometry	Minimum number of triangles, LODs, culling small objects at great distance, cull distance volumes [42], and reducing the view frustrum rendering far-plane distance.
Buildings	Buildings share the same material with a single set of 1024×1024 texture atlases [41] and the triangle count is kept low.
Foliage	Using impostors over raw 3D meshes or sprite billboards
Materials	Minimum number of master materials and material instancing
Artificial lights	Shadow contribution only when in 32 m range from a user camera.
Traffic system	Traffic computations reduced on areas not seen by any camera.

VI. PERFORMANCE ANALYSIS AND OPTIMIZATIONS

According to our experiments, the proposed system can achieve a stable rendering speed of over 60 *frames per second* (*fps*) up to 4K (3840 × 2160) resolution when run on an NVIDIA GTX 1060 or an equivalent consumer-grade *graphics processing unit* (*GPU*).

It even reaches real-time performance on lower-tier hardware through a multitude of optimizations, of which the most relevant techniques are listed in Table 3. The overall simulation speed can further be fine-tuned, e.g., by decreasing the number of shadow cascades, disabling volumetric fog during the day, and using more radical distance-culling of excess grass.

Our CiThruS2 environment was also benchmarked on a more powerful AMD Radeon 6900 XT GPU that increased the 4K rendering speed beyond 120 fps. Table 2 reports the 4K performance results of our environment alongside the widely used CARLA [13] and AirSim [14] frameworks on a desktop computer equipped with an AMD Ryzen 5900X CPU, AMD Radeon 6900 XT GPU, and 64 GB of RAM. Standalone builds of the environments were used, and the selected scenes were *Town10HD* for CARLA and *AirSimNH* for AirSim. For consistent results, the virtual camera was flown 3 m above the ground and average frame rates and times were measured over a period of 120 s. The obtained rendering speed of our CiThruS2 environment was 2.1× and 2.9× as high as those of CARLA and AirSim platforms, respectively.

VII. APPLICATIONS

Photorealistic graphics together with real-time performance and a broad range of adjustable features make our CiThruS2 framework a potential solution for many automotive applications out of which the following four usage scenarios are considered the most feasible:

1) An interactive virtual environment in the DIL simulators that detect driver's and other cabin occupants' immediate emotional states, functional ability, performance, and maneuvers in different traffic scenarios. Real-time simulators with stochastic traffic models can create conditions not foreseen by the driver.
2) Ground truth data collection to train neural networks for classification, object detection, and image segmentation

in vehicular vision applications. Objects of interest may include vehicles, pedestrians or other vulnerable road users, traffic signs, road lanes, register plates, traffic congestions, traffic lights changing from red to green, etc. The dataset can be diversified with different weather, time-of-day, and lighting conditions.

3) Real-time SIL and HIL simulation of the designed vision sensors, algorithms, and embedded systems in the virtual environment before integrating them in the vehicle and thereby ramp up the development and improve reliability.
4) Visualization of transportation planning and traffic flow control schemes in a realistic road network. New schemes can effortlessly be evaluated and validated by changing traffic regulations and altering the number of road users.

VIII. Conclusion

This paper introduced our open-source CiThruS2 simulation framework for vision-based ADAS development in a realistic high-fidelity virtual environment. The created 3D scene is modeled after a real-world traffic infrastructure, where driving and traffic can be simulated with practically any number of autonomous vehicles and pedestrians under various time-of-day, weather, and lighting conditions. Hence, it can serve as a virtual testbench and test drive platform for a broad range of traffic scenarios.

Our simulator is built on Unreal Engine 4 and carefully optimized for rendering speed. To the best of our knowledge, it is the only open-source simulation framework that can perform photorealistic real-time (60 fps) 4K rendering of a 3D city scene on a consumer-grade GPU such as NVIDIA GTX 1060. This unique set of features particularly speeds up the design and validation stages where real-time performance and photorealism are in the focal point. The openness also makes it accessible to the entire automotive value chain and thereby fosters large-scale development and deployment of disruptive ADAS technologies towards autonomous driving.

In the future, a more sophisticated traffic management system will be introduced. The environment will be populated with new road users (cyclists, trams, etc.), and the ego vehicle will be equipped with dynamics and sensor models. The level of immersion will be increased by Virtual Reality compatibility. These forthcoming features will evidently broaden the application space of the framework. The inherent computation overhead will be tackled by deploying more efficient implementation techniques such as NVIDIA *deep learning super sampling* (*DLSS*) or by introducing novel rendering techniques, such as checkerboard rendering with adaptive pixel temporal consistency filtering to reduce the number of pixels to be drawn.

References

[1] SAE International, “Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles,” *Standard J3016*, June 2018.

[2] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, “Advanced driver-assistance systems: a path toward autonomous vehicles,” *IEEE Consum. Electron. Mag.*, Sept. 2018, vol. 7, no. 5, pp. 18-25.

[3] H. Zhou, W. Xu, J. Chen, and W. Wang, “Evolutionary V2X technologies toward the internet of vehicles: challenges and opportunities,” *Proc. of the IEEE*, Feb. 2020, vol. 108, no. 2, pp. 308-323.

[4] A. Koesdwiady, R. Soua, F. Karray, and M. S. Kamel, “Recent trends in driver safety monitoring systems: state of the art and challenges,” *IEEE Trans. Veh. Technol.*, June 2017, vol. 66, no. 6, pp. 4550-4563.

[5] O. Burkacky, H. Deichmann, and J. P. Stein, “Automotive software and electronics 2030: mapping the sector’s future landscape,” *McKinsey & Company Inc.*, July 2019.

[6] “Waymo,” [Online]. Available: https://www.waymo.com/.

[7] “NVIDIA Drive Constellation,” [Online]. Available: https://developer.nvidia.com/drive/drive-constellation.

[8] “LG Autonomous Driving Simulator,” [Online]. Available: https://www.sylsimulator.com/.

[9] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, “A systematic review of perception system and simulators for autonomous vehicles research,” *Sensors*, Feb. 2019, vol. 19, no. 3, pp. 648.

[10] S. Bayarri, M. Fernandez, and M. Perez, “Virtual reality for driving simulation-SIRCA,” *Commun. ACM*, vol. 39, no. 5, May 1996, pp. 72-76.

[11] “TORCS: The Open Racing Car Simulator,” [Online]. Available: http://www.torcs.org.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: an open urban driving simulator,” *in Proc. Annual Conf. on Robot Learning*, Mountain View, California, USA, Nov. 2017.

[13] S. Shah, D. Dey, C. Lovett, and A. Kapoor, “AirSim: high-fidelity visual and physical simulation for autonomous vehicles,” *in Proc. Field and Service Robotics conf.*, Sept. 2017.

[14] M. Naumann, F. Poggenhans, M. Lauer, and C. Stiller, “CoInCar-Sim: an open-source simulation framework for cooperatively interacting automobiles,” *in Proc. IEEE Intell. Veh. Symp.*, Changshu, China, Oct. 2018.

[15] V. R. Aparow, A. Choudary, G. Kulandaivelu, T. Webster, J. Dauwels, and N. d. Boer, “A comprehensive simulation platform for testing autonomous vehicles in 3D virtual environment,” *in Proc. Int. Conf. Mechatronics Syst. Robots*, Singapore, May 2019, pp. 115-119.

[16] A. AbdelHamed, G. Tewolde, and J. Kwon, “Simulation framework for development and testing of autonomous vehicles,” *in Proc. IOT, Electron. Mechatronics Conf.*, Vancouver, British Columbia, Canada, Sept. 2020, pp. 1-6.

[17] T. T. Niemirepo, J. Toivonen, M. Viitanen, and J. Vanne, “Open-source CiThruS simulation environment for real-time 360-degree traffic imaging,” *in Proc. IEEE Int. Conf. Connected Vehicles and Expo*, Graz, Austria, Nov. 2019.

[18] “Windridge City Asset,” [Online]. Available: https://assetstore.unity.com/packages/3d/environments/roadways/windridge-city-132222.

[19] M. Pharr, W. Jakob, and G. Humphreys, “*Physically Based Rendering: from Theory to Implementation*,” 3rd ed., Morgan Kaufmann, Sept. 2016.

[20] L. Bruck, B. Haycock, and A. Emadi, “A review of driving simulation technology and applications,” *IEEE Open J. Veh. Technol.*, Nov. 2020, vol. 2, pp. 1-16.

[21] B. Ranft and C. Stiller, “The role of machine vision for intelligent vehicles,” *IEEE Trans. Intell. Veh.*, Mar. 2016, vol. 1, no. 1, pp. 8-19.

[22] Q. Chao, H. Bi, W. Li, T. Mao, Z. Wang, M.C. Lin, and Z. Deng, “A survey on visual traffic simulation: models, evaluations, and applications in autonomous driving,” *Comput. Graph. Forum*, vol. 39, no. 1, Feb. 2020, pp. 287-308.

[23] “MakeHuman,” [Online]. Available: http://www.makehumancommunity.org/.

[24] M. Kirscht and C. Rinke, “3D reconstruction of buildings and vegetation from synthetic aperture radar (SAR) images,” *in Proc. IAPR Workshop Machine Vision Appl.*, Nov. 1998, pp. 228-231.

[25] “Terrain.party,” [Online]. Available: https://terrain.party/.

[26] “OpenStreetMap,” [Online]. Available: www.openstreetmap.org.

[27] “National Land Survey of Finland,” [Online]. Available: https://www.maanmittauslaitos.fi/en.

[28] P. Kudela, M. Palčák, K. Zábovská, and B. Bučko, “Integration of photogrammetry within laser scanning approach,” *in Proc. Int. Conv. Inf., Commun. Electron. Technol.*, Opatija, Croatia, Sept. 2020, pp. 1691-1694.

[29] NVIDIA, “GPU Gems 3,” Part IV, Chapter 21, Aug. 2007. [Online]. Available: https://developer.nvidia.com/gpugems/gpugems3/contributors

[30] R. Brucks, “Impostor Baker Plugin,” [Online]. Available: https://github.com/ictusbrucks/ImpostorBaker.
[31] “Unreal Engine 4,” [Online]. Available: https://www.unrealengine.com/en-US/.
[32] U. E. 4. Documentation, “Light Shafts,” [Online]. Available: https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/LightShafts/index.html.
[33] U. E. 4. Documentation, “Post process materials,” [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/PostProcessEffects/PostProcessMaterials/index.html.
[34] NVIDIA, “GPU Gems,” Part III, Chapter 19, Mar. 2004, [Online]. Available: https://developer.nvidia.com/gpugems/gpugems3/contributors
[35] U. E. 4. Documentation, “Rendering Overview,” [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/Overview/index.html.
[36] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs, “A sorting classification of parallel rendering,” *IEEE Comput. Graph. Appl.*, July 1994, vol. 14, no. 4, pp. 23-32.
[37] U. E. 4. Documentation, “Screen Space Global Illumination,” [Online]. Available: https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/ScreenSpaceGlobalIllumination/index.html.
[38] U. E. 4. Documentation, “Screen Space Reflections,” [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/PostProcessEffects/ScreenSpaceReflection/index.html.
[39] H. Singh, S. Midlam-Mohler, and P. Tulpule, “Simulation based virtual testing for safety of ADAS algorithms - case studies,” *SAE Technical Paper*, Apr. 2021.
[40] J. Cohen, M. Olano, and D. Manocha, “Appearance-preserving simplification,” *in Proc. Annual Conf. Computer Graphics and Interactive Techniques*, July 1998.
[41] NVIDIA, “SDK white paper: improve batching using texture atlases,” July 2004.
[42] U. E. 4. Documentation, “Cull distance volume,” [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/VisibilityCulling/CullDistanceVolume/index.html.