CiThruS2: Open-Source Virtual Environment for Simulating Real-Time Drone Operations and Piloting

Emilian Gałązka, Arttu Leppäaho, and Jarno Vanne, *Member, IEEE*
Ultra Video Group
Tampere University
Tampere, Finland
{emilian.galazka, arttu.leppaaho, jarno.vanne}@tuni.fi

***Abstract*—Drone technology is gaining ground in many civilian, commercial, and military applications. However, the ever-increasing complexity of modern drones makes their conventional real-world testing impractical, so drone technology and service providers are turning to virtual simulation platforms for shorter times to market and lower development costs. This paper introduces the latest version of our See-Through Sight 2 (CiThruS2) open-source simulation framework that brings support for lifelike simulation of drone operations and piloting in real time. Our realistic virtual city twin comes with various zones, such as a city center, residential areas, bodies of water, and forests. This supplies the virtual drone flights with photorealistic aerial views of city life with a day and night cycle and various weather conditions. Our simulator reaches a 4K rendering speed of up to 60 frames per second on an NVIDIA GTX 1060 graphics card. The high visual fidelity and real-time performance make CiThruS2 a suitable platform for many virtual use cases like vision-based remote piloting, environment perception, cargo delivery, Internet-of-Drones, and visual data collection.**

***Keywords*—Unmanned aerial vehicle (UAV), real-time simulation, photorealism, open-source software, CiThruS2**

I. INTRODUCTION

The deployment of *unmanned aerial vehicles* (*UAVs*) is gaining momentum across many civilian, commercial, and military applications [1]. Smart UAVs, a.k.a. drones, are on a course to revolutionize the traditional operating models by providing means to execute versatile tasks at a faster speed, a lower cost, and a higher flexibility that goes beyond human performance and physiological characteristics. Therefore, UAVs are increasingly incorporated into existing workflows, supply chains, and ecosystems, where they are used to tackle various technological and societal challenges.

Drone flights can be classified into remote control, semi-autonomous, and full-autonomous operations, and the evolution is gradually continuing towards fully autonomous aerial systems without any human intervention [2]. The design complexity of autonomous UAVs will inevitably increase together with the level of automation due to the explosive growth of drone flight controller software, *electrical/electronic* (*E/E*) components, and sensor modalities [3]. Holistic situational awareness is sought by analysis models that need data from cameras, RADARs, LiDARs, and other environment-sensing technologies.

This work is part of the ADACORSA 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 876019.

Fig. 1. Aerial view of the CiThruS2 virtual city twin.

Changing the role of the drone operator from manual execution to supervision or setting the UAV into an autopilot mode explodes the number of possible test scenarios, which severely reduces the practicality of exhaustive real-life testing due to the financial and time constraints [4]. Furthermore, the need to comply with various safety standards and regulations adds to the pressure of ensuring proper functional verification and validation. Various environmental or weather conditions can also be cumbersome to reproduce consistently or stochastically in real-world test fields. In the worst case, physical tests may also lead to unwanted accidents and system failures [5].

All these reasons motivate developers to design future drone technologies in a safe, fail-operational, and cost-effective manner. This design approach calls for virtual simulation environments, where different functionalities can be tested, iterated upon, and verified under different controllable parameter settings and environmental conditions prior to implementation in the real-life systems.

There are several commercial drone simulators in the market, such as DJI Drone Simulator [6], RealFlight RF9.5 Drone Simulator [7], and droneSim Pro Drone Simulator [8]. However, none of them provide open access to developers, regulators, or other stakeholders, so they are left out of the scope of this paper.

The most noteworthy open-source virtual UAV simulation platforms include Flightgear [9], Gazebo [10], jMAVSim [11], and AirSim [12], [13]. There are also numerous academic approaches, of which this paper considers the works of Yang et al. [14] and Chen et al. [15] due to their similarities in digital twin creation and algorithmic research. It is also

TABLE I. FEATURE COMPARISON OF EXISTING AND PROPOSED OPEN-SOURCE SIMULATION FRAMEWORKS

Platform	License	Year	Photorealistic	Realistic layout	Lightweight	Weather	Traffic participants	Manual control
Flightgear [9]	GPL	1997	Yes[1]	Yes	Yes	Yes	Yes[2]	Yes
Gazebo [10]	Apache 2.0	2012	No	No	Yes	No	No	Yes
jMAVSim [11]	GPL	2013	No	No	Yes	No	No	Yes
AirSim [12]	MIT	2017	Yes[3]	Yes[3]	No	Yes	No	Yes
Yang et al. [14]	NA	2020	No	No	Yes	No	No	No
Chen et al. [15]	NA	2022	No	Yes[4]	Yes	No	No	Yes
Gymnasium [16]	MIT	2023	No	No	Yes	No	No	No
CiThruS2 (Ours)	MIT	2023	Yes	Yes	Yes	Yes	Yes	Yes

[1]Far view only [2]Only airports [3]Scene dependent [4]Limited

relevant to acknowledge toolkits for AI-driven development, such as Gymnasium [16]. However, none of these open approaches can provide users with a combination of real-time speed, photorealism, a realistic layout of the environment, populated scene, and other environmental effects.

This paper introduces the latest version of our open-source simulation framework called *See-Through Sight, version 2* (*CiThruS2*). The first version of the framework [17] was primarily designed for land vehicles, and it was published in 2019 and built upon the Windridge City [18] environment. However, the road infrastructure and geometry (lane widths, curves, etc.) of this virtual city template were found to be too different from a real-world city. These limitations motivated us to develop a new digital twin environment with a realistic city infrastructure and thereby bridge the gap between virtual and physical testing. This resulted in the creation of CiThruS2 in 2021 [19]. Since then, it has been actively developed by enhancing graphical fidelity, optimizing performance, and introducing new features. Even though the environment was originally developed for driving and traffic simulations on the ground, the added verticality with consistent quality and extra features allowed us to extend it to UAV simulation as well.

CiThruS2 is modeled after Hervanta, a suburb of the city of Tampere, Finland. Fig. 1 depicts a snapshot of our virtual 3D urban scene that can be populated with hundreds of simulated traffic participants. The simulation environment uses realistic *physically based rendering* (*PBR*) [20] and it also offers a wide range of adjustable parameters and simulation conditions, including control over lighting and weather effects. The passage of time can be simulated dynamically, allowing the user to experience a full 24-hour period in a single session. Additionally, the user can manually enable specific weather conditions, such as rain, snow, or fog, to practice drone piloting in specific scenarios.

Our environment is distributed under the permissive MIT open-source license on GitHub at

github.com/ultravideo/CiThruS2

CiThruS2 was created in Unreal Engine 4.26 using the C++ programming language and the Unreal Engine toolset. Special attention is paid to performance optimization, and the latest version achieves real-time rendering speed even on lower-end consumer-grade hardware.

The broad spectrum of features makes our simulation software a potential virtual testing platform for a wide variety of drone flight scenarios, such as 1) simulating *beyond visual line-of-sight* (*BVLOS*) operations [21] through low latency video streaming from the onboard camera(s); 2) validating onboard *software-in-the-loop* (*SIL*) and *hardware-in-the-loop* (*HIL*) setups in vision-based environment perception; 3) modelling computation and communication in *Internet-of-Drones* (*IoD*) [22]; 4) visualizing human-in-the-loop and autonomous transportation planning and cargo delivery in a realistic urban environment; and 5) collecting visual data (ground truth) for training neural networks and related computer vision techniques [23].

The rest of the paper is organized as follows. Section 2 characterizes the existing open-source UAV simulation platforms. Section 3 provides an overview of the proposed CiThruS2 simulation framework and its main functional blocks. The environment itself, including its natural and man-made components, is described in Section 4, followed by the weather and lighting effects in Section 5. Section 6 evaluates the simulation performance and implemented optimizations. The main application domains 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 UAV simulation. A fully-fledged virtual drone testbed calls for a realistic urban environment and conditions that reflect the real world. As such, we primarily characterize the environments in terms of their photorealism, parity with a real-life layout, and natural effects.

A high rendering speed is paramount for realistic human-in-the-loop and autonomous simulations, especially in usage scenarios requiring high UAV speed or precision. It is not possible to simulate such scenarios unless the system is able to output a video feed at a frame rate comparable to real video cameras.

Being able to render multiple virtual cameras in real time is another motivation for a lightweight simulation environment. In a typical case, a UAV might have multiple visual sensors attached to it, and it is challenging to render them all in real time in a virtual environment. UAV remote control systems can also introduce additional overhead due to payload transmission and latency in handshaking.

The Flightgear [9] and AirSim [13] platforms offer the most comparable feature set over our simulation environment. Flightgear has been developed for large-sized aircraft that operate in vast open-air spaces but not in

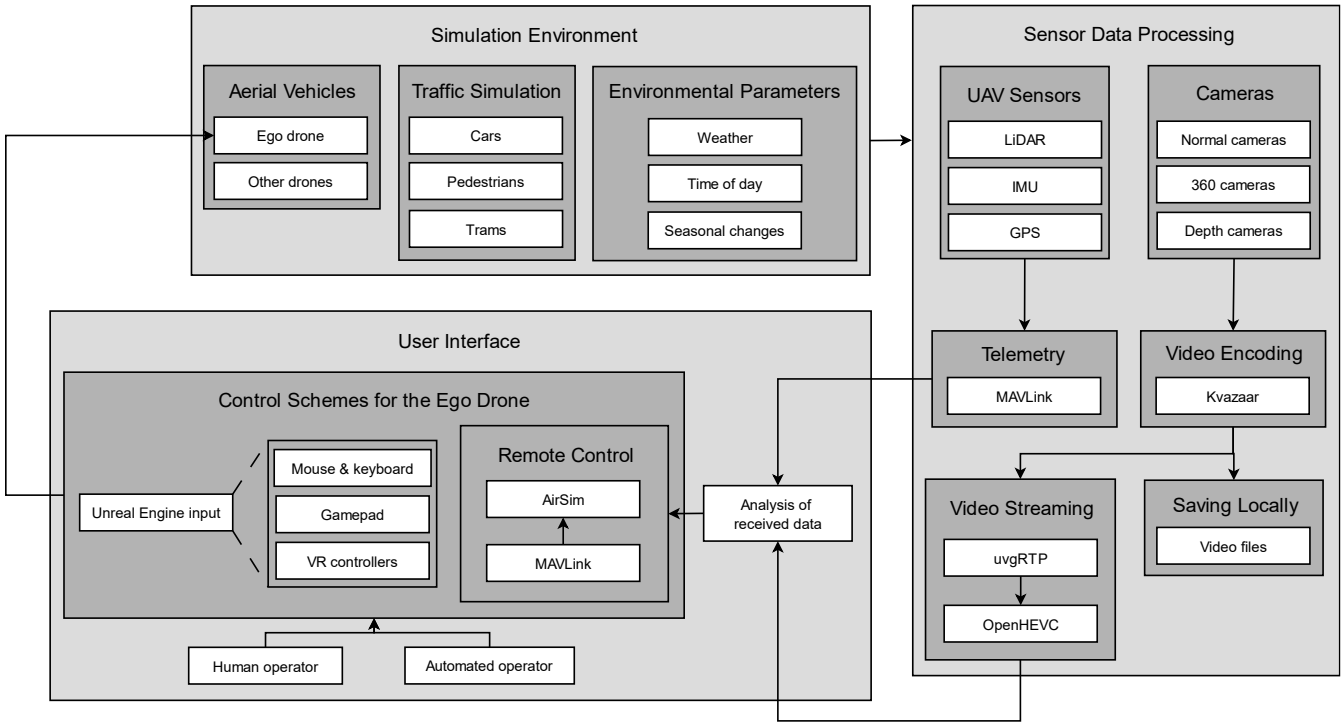

Fig. 2. A diagram of the CiThruS2 simulation framework.

congested urban zones. AirSim is primarily meant for aerial vehicle testing and evaluation, but it fails to provide high visual fidelity in many scenes, lacks the high frame rate, is missing features such as recording 360-degree video, and does not support other traffic participants such as cars or pedestrians.

Gazebo [10] offers an extensive toolkit for designing the behavior and functionality of a UAV, but it lacks the large, realistic environment for testing. Correspondingly, jMAVSim [11] comes without a photorealistic environment where tests could be conducted. Similar shortcomings can be found in academic simulators released by Yang et al. [14] and Chen et al. [15], as the digital twin fidelity is not sufficient or the available functionality does not cover real-time video streaming.

III. CiThruS2 Simulation Framework

Fig. 2 depicts a conceptual block diagram of the CiThruS2 simulation framework that consists of three main functional blocks: the simulation environment, user interface, and visual data processing. The state of the simulation environment can be modified through the user interface and various settings, e.g., for weather and traffic. Afterwards, sensor data can be captured from the environment in various formats and either streamed to the user in real time or stored for later use.

A. Simulation Environment

The CiThruS2 simulation environment supports 1) dynamic entities such as vehicles and other moving objects; 2) static entities such as terrain, roads, buildings, and trees that are detailed in Section 4; and 3) simulated weather, time-of-day and seasonal effects described in Section 5.

Ground level dynamic entities include passenger cars, trams and pedestrians that utilize a waypoint graph system for autonomously moving along paths on the ground according to traffic rules. The geometry of these vehicles was crafted by us using existing commercial vehicles as reference. In addition, some free, ready-made assets with appropriate licenses were used. The visual human models of the pedestrians were created using the open-source MakeHuman [24] software. As for aerial vehicles, a common quadcopter UAV model is supported through the AirSim plugin for Unreal Engine 4 [13], which simulates the forces and torques of each of the copter's rotors realistically and accounts for drag, friction, and gravity as well.

Further variety can be easily added by introducing vehicles of different models or types or pedestrians of varying body types, outfits, age groups or ethnicities. As such, it is possible to capture diversified footage, e.g., for computer vision purposes.

B. User Interface

CiThruS2 is built on top of Unreal Engine, which supports many common video game control schemes that are also available for controlling virtual UAVs in the simulation environment. Available schemes include mouse and keyboard, commercial gamepads such as Xbox controllers or PlayStation controllers, and controllers used alongside

commercial VR headsets such as the HTC Vive or Valve Index.

CiThruS2 also lets the user control a virtual ego drone remotely by utilizing the MAVLink [25], [26] support included in AirSim [13]. Since MAVLink is commonly used by real-life UAVs [13], [26], the virtual experiments smooth out the way for real-world field tests.

Both manual and automatic remote-control schemes can be used, as MAVLink makes both possible [25]. As such, the ego drone can be piloted accurately by both humans and computers without any additional interfaces between the simulation framework and control peripherals or any other additional hardware or software with existing MAVLink support.

C. Sensor Data Processing

2D or 360-degree video footage can be captured in real time from anywhere in the environment, including moving objects like simulated UAVs. Captured video footage can be streamed forward or saved into video files as needed.

Besides cameras, it is also possible to simulate sensor technologies like *light detection and ranging* (*LiDAR*), *inertial measurement units* (*IMU*) consisting of accelerometers, gyroscopes and magnetometers, *global positioning system* (*GPS*) tracking, barometers, and distance sensors. This sensor data is transmitted through MAVLink [25].

These sensors are provided by the AirSim [13] plugin, and the implementations it provides for them are fast but close to reality. The output of the simulated sensors has been observed to be similar to the output of equivalent real-world sensors [13]. It also includes simple artifacts, such as white noise and latency, which are based on the datasheets and behavior of real sensors.

The video streaming system uses the open-source Kvazaar encoder [27] to compress the captured video to the *High Efficiency Video Coding* (*HEVC*) format and the open-source *Real-Time Transport Protocol* (*RTP*) library uvgRTP [28] to transmit the HEVC bitstream to the receiver. Practically any HEVC decoder, such as OpenHEVC [29], can be used to decode the video.

IV. Geospatial Data

The CiThruS2 virtual environment is a digital replica of an approximately 9 km^2 (3 km × 3 km) area in Hervanta, Tampere, Finland. The development of this digital twin was accelerated by using data from external sources like photos, topological maps, and satellite imagery. This approach also made it easier to ensure that the model was accurate and consistent with the real-world area. At a high level, the graphical side of the environment is made up of four layers: terrain, roads, buildings, and foliage, each contributing to the final look of the environment.

A. Terrain and Roads

The terrain layer is derived from a heightmap [30] acquired from OpenStreetMap [31] and National Land Survey of Finland's Topographic Database [32]. Although the acquired data provided a good starting point for terrain creation, some manual adjustments were made for better consistency between the real and virtual worlds. Proper city layout and road placement were also extracted from OpenStreetMap [31], with only minor adjustments for better performance.

B. Buildings

The model exported from OpenStreetMap also included the locations and basic shapes of most buildings. Initially, they served as placeholders for size and location and were subsequently replaced with custom, more accurate models.

The buildings were created with a combination of manual and procedural workflows using reference images from both ground-level and aerial footage. This approach was chosen over, e.g., aerial photogrammetry or laser scanning [33], as it allowed for more consistent and precise control over the style and shape of the buildings. It also allowed for highly controllable performance management and ensured consistent quality of the objects at any viewing angle and distance. Photorealistic buildings and textures are crucial to properly functioning computer vision algorithms [23].

The development time was significantly sped up by reusing modules and elements in buildings with similar features. This was especially important for constructions not commonly visible from the ground level.

The buildings are textured with an easily extensible material layers system, which also results in better performance than traditional materials. Windows are made of a layered material, which provides shader descriptions for reflectance, glass, curtains, and simulated interiors using a displaced cube map.

C. Foliage

The city of Hervanta is surrounded by a significant amount of foliage and forestry. A realistic simulation calls for well-designed grass, trees, and other vegetation assets. In many computer graphics programs, foliage can consume a significant portion of the total processing power, which gave motivation for its careful optimization. Rendering transparency requires that the depth buffer is sorted once for every overlapping pixel with a non-unit alpha value in the pixel shader, resulting in rendering the same pixel multiple times. As a result, the performance is severely impacted in areas with layered transparency, and that often includes heavy foliage areas. *Classic Level-of-Detail* (*LOD*) techniques can mitigate the impact of foliage on performance, but simply reducing the triangle count of individual foliage elements is not sufficient.

Another approach for optimizing distant trees is called billboard sprites [34], which render the tree image into a

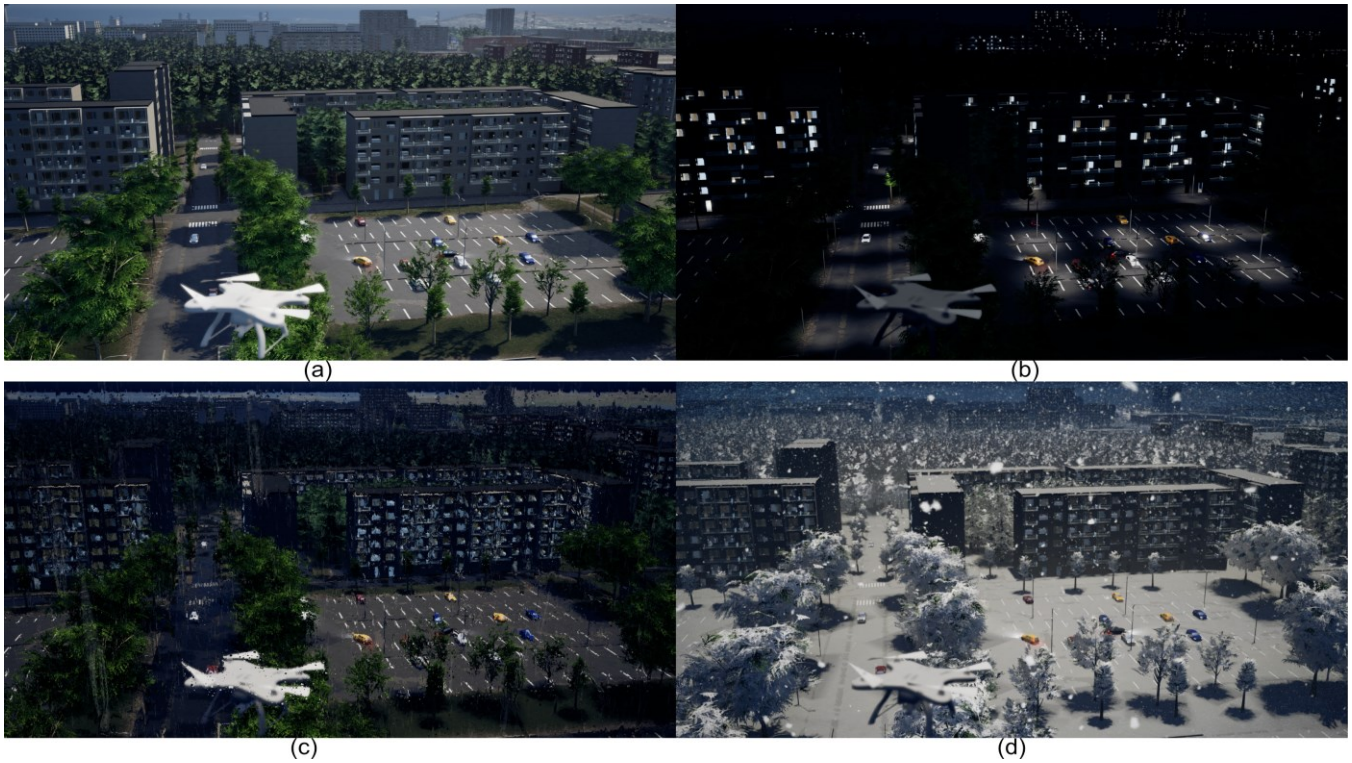

Fig. 3. View from behind the drone in various weather conditions: (a) Clear, (b) Night, (c) Rain, (d) Snow.

simple quad-plane. Although it is an efficient optimization technique, the effect is easily distinguishable and does not maintain the believability of the virtual scene. It is particularly easy to spot when viewed from the drone perspective, as the billboards are exclusively 2D. Billboards also increase the amount of layered transparent objects and make the transparency overdraw issue worse without any significant performance boost.

Octahedral impostors [34] were found to allow the creation and maintenance of a large volume of trees with high visual fidelity. CiThruS2 makes use of a ready-made implementation for rendering octahedral impostors in Unreal Engine 4 [35]. It allows the simulation to contain more than 200 000 trees inside of view frustum without any performance penalty.

V. Weather and Lighting Effects

The environment allows for control of multiple common weather conditions including rain, snow, or fog. The weather control is required to fine-tune the precise state that the environment supplies for the UAV simulation. These weather effects are also accompanied by lighting, which is one of the most important contributors to a photorealistic visual style. Therefore, it is critical to simulate light as realistically as possible. Lighting conditions also have a significant impact on the camera systems installed on a UAV, e.g., darkness, glare, or reflections may pose a challenge when parsing visual information from the camera footage.

It was crucial to include as many possible lighting variations as the user could realistically experience. Therefore, the system implements a realistic day and night cycle, ambient lighting, artificial lights, and a set of screen-space effects. Conditions like harsh evening lighting, overcast sky, or night require different actions to safely control the drone and thus come with their unique sets of difficulties.

A. Volumetrics

Unreal Engine 4 [36] comes with a built-in system for simulating volumetric fog [37]. In our environment, it is turned off by default during the day, as it has minimal impact on the visuals alone. It can, however, produce natural effects such as light shaft occlusion [38] and light shaft bloom [38].

Volumetric fog is enabled during the night and in rainy weather. At night, it provides a light scattering effect that becomes visible under streetlights and in front of car headlights.

The system allows the user to leverage the Volumetric Clouds [39] system to enhance the simulation sky and lighting with an extra layer of realism. The performance drop is noticeable, albeit small, but it might be warranted in specific test cases to take advantage of it.

B. Weather and Seasonal Effects

CiThruS2 supports most common weather types, as visualized in Fig. 3. Fig. 3(a) serves as a clear day reference

and Fig. 3(b) presents the same location at night. The inclusion of weather effects severely changes the look and feel of the environment with Fig. 3(c) showing rainy conditions and Fig. 3(d) showing winter conditions.

The rain streaks were implemented using a post-processing material [40] which imitates rain streaks or splashes depending on the direction of the camera. Raindrop ripples and streaks also appear on selected objects, such as roads, most of the traffic control objects, and windows on cars and buildings. A particle system is responsible for creating actual rain drops to fall from above the camera view. The system can easily simulate heavy, dense rain with the addition of extra planar layers oriented towards the camera.

A realistic snow effect was implemented similarly to the rain; a particle system is 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 roofs of buildings. The depth of snow on the ground was imitated using a vertex displacement shader. In addition, a lens freezing effect was made. It is implemented similarly to the rain streaks effect and is also present on windows.

Wind moves foliage and trees globally and responds properly to the specified direction. This is especially visible during the autumn season, when leaves fall and are scattered according to the wind movement.

C. Day and Night Cycle

The implemented day-and-night cycle accurately reproduces the passage of time by positioning the sun in the sky and specifying the hour of the day. Their control is given to the user. The system also manages the changing color and light quality as the day progresses.

D. Natural Lights

The environment is lit with a single directional light during the day and night, with color and intensity dependent on the time of day. A skylight provides ambient lighting by image-based lighting techniques [41] and its color contribution is updated in real-time, allowing for realistically changing ambient lighting in the environment.

E. Additional Lighting and Effects

CiThruS2 currently uses the default render pipeline for Unreal Engine 4, the deferred shading pipeline [42]. It was chosen over the other common shading pipeline, the forward renderer [43], because deferred shading heavily reduces the performance impact of lights. Thanks to it, the environment can be equipped with a multitude of lights contributing to the scene illumination by removing their shadow-casting based on distance from the camera. This design choice is impactful during nighttime, with all the urban street, house, and traffic lighting turned on.

The system also supports several screen-space effects, such as screen-space global illumination [44] and screen-space reflections [45]. Despite the utility of these techniques being limited, as they are only affected by objects currently visible on the screen, it makes them more efficient while enhancing the rendered visuals.

VI. Performance Analysis

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

The system can reach real-time performance on lower-tier hardware thanks to a multitude of optimization techniques, the most relevant of which are listed in Table 2. The overall simulation speed can be further improved by disabling visual effects and using more drastic optimization techniques.

The CiThruS2 environment was also benchmarked on a more powerful AMD Radeon RX 6900 XT GPU that increased the rendering speed beyond 90 FPS. Table 3 reports the 4K performance results of our environment alongside the widely used FlightGear [9] and AirSim [13] frameworks on a desktop computer equipped with an Intel Core i7-5960X CPU, AMD Radeon 6900 XT GPU, and 32 GB of RAM.

Standalone release builds of the environments were used, and the selected scenes were *Keflavik* for FlightGear and *AirSimNH* for AirSim. As it was not possible to replicate the exact same conditions in each environment due to implementation differences, frame rates were measured in a few different common scenarios to obtain average values. The rendering speed of our CiThruS2 environment was observed to be on a comparable level to those of the FlightGear and AirSim platforms, while also offering equal or higher visual fidelity.

VII. Applications

CiThruS2 provides easy access to a sizeable environment with many distinct natural and urban zones and points of interest. The realistic graphics coupled with real-time performance and a plethora of user adjustable settings lead to CiThruS2 being a potential choice in many applications that assist in testing and training UAVs.

The simulation environment can serve as a virtual testbench for various UAV test cases, such as remote controlling a drone, gathering aerial footage from an urban environment, or landing in a crowded area. Our simulation framework is most geared towards:

1) Carrying out virtual BVLOS operations by remotely controlling a virtual drone in the environment through a low latency video stream sent from the video camera(s) of the drone. Video footage can also be used to gather information about the surroundings of the drone for better situational awareness.

TABLE II. OPTIMIZATION TECHNIQUES USED IN CITHRUS2

Feature	Optimization technique
All geometry	Minimal number of triangles, Level-Of-Detail meshes for each object, distance culling and cull distance volumes [46], far plane clipping, proper occlusion culling.
Buildings	The same master material for all buildings. Low texture resolution with tiling.
Foliage	The intensive usage of impostors [35].
Materials	The usage of material layers [47] system and dynamic layer management for buildings. The usage of proper material instancing to reduce draw calls.
Artificial lights	Shadow contribution only at short distances.
Traffic	Balanced performance with proper instancing, reduced physics and collision interaction and reduced computation in occluded areas.
Post processing	Introduction of DLSS [48] plugin to upscale the image.

2) Testing and validation of vision-based onboard perception algorithms in real-time SIL and HIL simulations can aid in the design of vision sensors, algorithms, and embedded systems. Simulating the design-under-test in the virtual environment before integrating it in a real-life drone can improve performance, reliability, and robustness of the implementation and thus increase the flexibility of development.
3) Simulate the feasibility of different Internet-of-Drones setups for environment perception, visual data collection, and other imaging applications. Virtual environment supports iterative design approach that seeks for optimal distribution of compute nodes at device and edge layers as well as efficient communication between them.
4) Visualize diverse autonomous transportation, plan cargo delivery routes, and smart city control flows. A virtual environment allows for simple and accurate development and iteration of such use cases, while optimizing time efficiency, costs and viability.
5) Collecting ground truth and base visual data used in calibrating neural networks for specific uses, including classification, object detection, and image segmentation in aerial drone vision applications. Objects of interest may include vehicles or pedestrians, traffic signs, road lanes, license plates, traffic congestions, balconies, cargo drop-off points, etc. Further variation can be added to the acquired data set by applying different conditions such as changes in seasons, weather, or time of day.

VIII. CONCLUSION

This paper presented the latest version of our an open-source CiThruS2 simulation framework that makes it possible to prototype and test UAVs in a realistic, high-fidelity virtual environment. The simulation scene was modeled after real-world city infrastructure, and it supports simulations of drones and drone swarms under a range of time-of-day, weather, and lighting conditions. This virtual environment can be used as a testing platform for a wide range of UAV flight missions.

TABLE III. PERFORMANCE RESULTS ON AN AMD RADEON RX 6900 XT

Platform	Scene	Average frame rate range
FlightGear [9]	Keflavik	62-80 FPS
AirSim [13]	AirSimNH	36-94 FPS
Ours (CiThruS2)	Hervanta	52-100 FPS

Our simulator is built on Unreal Engine 4 and has been optimized for real-time simulations. It can achieve real-time (60 FPS) 4K rendering speed of a 3D city scene on a consumer-grade GPU like the NVIDIA GTX 1060. The unique features of the environment make it a useful tool for design and validation stages, where real-time performance and photorealism are key considerations. The open-source nature also increases the accessibility for potential users and as such fosters development and research of UAV technologies and services.

REFERENCES

[1] IEEE, IEEE Std 1936.1-2021, IEEE Standard for Drone Applications Framework, Dec. 2021.

[2] H. Shakhatreh et al., "Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges," *IEEE Access*, vol. 7, 2019, pp. 48572-48634.

[3] O. Burkacky, H. Deichmann, and J. P. Stein, "Automotive software and electronics 2030: mapping the sector's future landscape," McKinsey & Company Inc., July 2019.

[4] Y. B. Sebbane, "Smart autonomous aircraft: Flight control and planning for UAV," Boca Raton: CRC Press, 2015.

[5] A. Bajaj, B. Philips, E. Lyons, D. Westbrook, and M. Zink, "Determining and communicating weather risk in the new drone economy," *in proc. IEEE Veh. Technol. Conf.,* Victoria, British Columbia, Canada, Feb. 2020.

[6] DJI, "DJI simulator site," 2019, [Online]. Available: https://www.dji.com/pl/simulator.

[7] Horizon Hobby, "RealFlight simulator site," 2020, [Online]. Available: https://www.realflight.com/.

[8] "droneSim Pro main site," 2015, [Online]. Available: https://www.dronesimpro.com/.

[9] FlightGear, "Flightgear main site," 1997, [Online]. Available: https://www.flightgear.org/.

[10] "Gazebo main site," 2012, [Online]. Available: https://gazebosim.org/home.

[11] "jMAVSim Github," 2013, [Online]. Available: https://github.com/PX4/jMAVSim.

[12] Microsoft, "AirSim documentation site," 2017, [Online]. Available: https://microsoft.github.io/AirSim/.

[13] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-fidelity visual and physical simulation for autonomous vehicles," in: M. Hutter and R. Siegwart. (eds) "Field and Service Robotics." *Springer Proceedings in Advanced Robotics*, vol 5, 2018.

[14] Y. Yang, W. Meng, H. Li, R. Lu, and M. Fu, "A digital twin platform for multi-rotor UAV," *in proc. Chinese Control Conf.*, Shanghai, China, Jul. 2021.

[15] S. Chen, W. Zhou, A.-S. Yang, H. Chen, B. Li, and C.-Y. Wen, "An end-to-end UAV simulation platform for visual SLAM and navigation," *Aerospace 2022*, vol. 9, no. 48, Jan. 2022, pp. 1-16.

[16] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. de Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. J. S. Tan, O. G. Younis, "Gymnasium," 2023, [Online]. Available: https://zenodo.org/record/8127025.

[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," [Online]. Available: https://naturemanufacture.com/windridge-city/.

[19] E. Gałązka, T. T. Niemirepo, and J. Vanne, "CiThruS2: Open-source photorealistic 3D framework for driving and traffic simulation in real time," *in Proc. to IEEE Int. Conf. Intell. Transp. Syst.,* Indianapolis, Indiana, USA, Sept. 2021.

[20] M. Pharr, W. Jakob, and G. Humphreys, "Physically based rendering: from theory to implementation," Sept., 2016.

[21] E. Politi, I. Varlamis, K. Tserpes, M. Larsen, and G. Dimitrakopoulos, "The future of safe BVLOS drone operations with respect to system and service engineering," *IEEE Int. Conf. Service-Oriented Syst. Eng.*, Newark, California, USA, Aug. 2022.

[22] L. Abualigah, A. Diabat, P. Sumari, and A. H. Gandomi, "Applications, deployments, and integration of internet of drones (IoD): a review," *IEEE Sensors J.*, vol. 21, no. 22, Nov. 2021. pp. 25532-25546.

[23] 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.

[24] "MakeHuman" [Online]. Available: http://www.makehumancommunity.org/.

[25] L. Meier, "MAVLink," [Online]. Available: https://mavlink.io.

[26] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith, and M. Khalgui, "Micro air vehicle link (MAVlink) in a nutshell: a survey," *IEEE Access,* vol. 7, Jun. 2019, pp. 87658-87680.

[27] M. Viitanen, A. Koivula, A. Lemmetti, A. Ylä-Outinen, J. Vanne, and T. D. Hämäläinen, "Kvazaar: Open-source HEVC/H.265 encoder," *in Proc. ACM Int. Conf. Multimedia*, Amsterdam, The Netherlands, Oct. 2016.

[28] A. Altonen, J. Räsänen, J. Laitinen, M. Viitanen, and J. Vanne, "Open-source RTP library for high-speed 4K HEVC video streaming," *in Proc. IEEE Int. Workshop on Multimedia Signal Processing*, Tampere, Finland, Sept. 2020.

[29] W. Hamidouche, M. Raulet, and O. Déforges, "Real time SHVC decoder: implementation and complexity analysis," *in Proc. IEEE Int. Conf. on Image Process.,* Paris, France, Oct. 2014.

[30] N. Greene, "Environment mapping and other applications of world projections," *IEEE Comput. Graph. Appl.*, vol. 6, no. 11, Nov. 1986, pp. 21-29.

[31] "OpenStreetMap," [Online]. Available: www.openstreetmap.org.

[32] "National land survey of Finland," [Online]. Available: https://www.maanmittauslaitos.fi/en.

[33] P. Kudela, M. Palčák, K. Zábovská, and B. Bučko, "Integration of photogrammetry within laser scanning approach," *in proc. Int. Conv. on Inf., Commun. Eletr. Technol.,* Opatija, Croatia, Sept. 2020.

[34] Nvidia, "GPU Gems 3," Part IV, Chapter 21, Aug. 2007.

[35] R. Brucks, "Impostor Baker plugin," [Online]. Available: https://github.com/ictusbrucks/ImpostorBaker.

[36] "Unreal Engine 4," [Online]. Available: https://www.unrealengine.com/en-US/.

[37] U. E. 4. Documentation, "Volumetric fog," [Online]. Available: https://docs.unrealengine.com/4.27/en-US/BuildingWorlds/FogEffects/VolumetricFog/.

[38] U. E. 4. Documentation, "Light shafts," [Online]. Available: https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/LightShafts/index.html.

[39] U. E. 4. Documentation, "Volumetric clouds," [Online]. Available: https://docs.unrealengine.com/4.26/en-US/BuildingWorlds/LightingAndShadows/VolumetricClouds/.

[40] U. E. 4. Documentation, "Post process materials," [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/PostProcessEffects/PostProcessMaterials/index.html.

[41] Nvidia, "GPU Gems," Part III, Chapter 19, Mar. 2004.

[42] U. E. 4. Documentation, "Rendering overview," [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/Overview/index.html.

[43] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs, "A sorting classification of parallel rendering," *in IEEE Computer Graphics and Applications,* July 1994, vol. 14, no. 4, pp. 23-32.

[44] U. E. 4. Documentation, "Screen Space Global Illumination," [Online]. Available: https://docs.unrealengine.com/en-US/BuildingWorlds/LightingAndShadows/ScreenSpaceGlobalIllumination/index.html.

[45] U. E. 4. Documentation, "Screen Space Reflections," [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/PostProcessEffects/ScreenSpaceReflection/index.html.

[46] U. E. 4. Documentation, "Cull distance volume," [Online]. Available: https://docs.unrealengine.com/en-US/RenderingAndGraphics/VisibilityCulling/CullDistanceVolume/index.html.

[47] U. E. 4. Documentation, "Material layers," [Online]. Available: https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Materials/MaterialLayers/.

[48] Nvidia, "Nvidia Deep Learning Super Sampling," [Online]. Available: https://developer.nvidia.com/rtx/dlss.