AirTaxiSim: A Simulator for Autonomous Air Taxis

Ayoosh Bansal*† and Mikael Yeghiazaryan*‡
University of Illinois Urbana-Champaign, Urbana, Illinois, 61801

Hyung-Jin Yoon*§
Tennessee Tech University, Cookeville, Tennessee, 38505

Duo Wang¶‖
University of Nevada, Reno, Nevada, 89557

Ashik E Rasul**
Tennessee Tech University, Cookeville, Tennessee, 38505

Chuyuan Tao††
University of Illinois Urbana-Champaign, Urbana, Illinois, 61801

Yang Zhao‡‡
Northeastern University, Boston, Massachusetts, 02115

Tianyi Zhu§§
California Institute of Technology, Pasadena, California, 91106

Oswin So¶¶ and Chuchu Fan***
Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139

Petros Voulgaris†††
University of Nevada, Reno, Nevada, 89557

Naira Hovakimyan‡‡‡ and Lui Sha§§§
University of Illinois Urbana-Champaign, Urbana, Illinois, 61801

The rapid advancements in air mobility vehicles is paving the way for air taxis to become a viable mode of public transportation. The next technological frontier for air taxis is fully autonomous operation. Developing safe and efficient autonomous control for air taxis presents greater challenges than for ground vehicles due to the inherent instability of aerial vehicles. Therefore, simulation solutions for autonomous air taxis will play a crucial role in accelerating their development and eventual safe deployment.

This paper introduces AirTaxiSim, an end to end simulation framework for autonomous air taxis. AirTaxiSim is designed to model and analyze the complexities of autonomous air

*Ayoosh Bansal, Mikael Yeghiazaryan, and Hyung-Jin Yoon contributed equally to this work.
†Postdoctoral Research Associate, Computer Science, 201 North Goodwin Avenue, Urbana, IL 61801, USA.
‡Research Assistant, Mechanical Science and Engineering, 1206 W. Green St. MC 244, Urbana, IL 61801, USA.
§Assistant Professor, Mechanical Engineering, 115 W. 10^{th} St, Cookeville, TN 38505, USA.
¶Postdoctoral Research Associate, Mechanical Engineering Department, 1664 N. Virginia Street, Reno, NV 89557
‖Visiting Scholar at UIUC, Mechanical Science & Engineering Department, 105 S Mathews Ave, Urbana, IL 61801
**Graduate Student, Mechanical Engineering Department, 1 William L Jones Dr, Cookeville, TN 38505.
††Graduate Student, 1206 W. Green St. MC 244, Urbana, IL 61801, USA.
‡‡Research Assistant, Department of Mechanical and Industrial Engineering, 360 Huntington Avenue, Boston, Massachusetts, 02115
§§Research Assistant, Electrical Engineering, 1200 East California Boulevard Pasadena, California, 91125
¶¶Graduate Student, Department of Aeronautics and Astronautics, 125 Massachusetts Ave, Cambridge, Massachusetts, 02139
***Associate Professor, Department of Aeronautics and Astronautics, 125 Massachusetts Ave, Cambridge, Massachusetts, 02139
†††Professor, Mechanical Engineering Department, 1664 N. Virginia Street, Reno, NV 89557
‡‡‡W. Grafton and Lillian B. Wilkins Professor, Mechanical Science and Engineering, 1206 W. Green St. MC 244, Urbana, IL 61801, USA.
§§§Donald B. Gillies Chair in Computer Science, Computer Science, 201 North Goodwin Avenue, Urbana, IL 61801, USA.

taxi operations in dynamic and cluttered urban environments. AirTaxiSim integrates high-fidelity physical models of vertical take-off and landing air vehicles in photo-realistic urban environments. The primary purpose of AirTaxiSim is to evaluate the safety, performance, and efficiency of autonomous air taxi services, across a variety of scenarios, including dangerous edge cases. AirTaxiSim also provides methods for generating datasets and establishing benchmarks for autonomous air taxis. This paper describes the simulator's construction, functionalities, and some of the use cases, providing critical information to facilitate its use in advancing autonomy in aerial vehicles. https://github.com/CPS-IL/airtaxisim

I. Introduction

Autonomous air taxis are poised to revolutionize urban transportation, providing a convenient and efficient mobility solution [1, 2]. A major challenge in the development of autonomous air taxis are the risks and costs of real world validation using life scale vehicles. This challenge is further exacerbated by the natural instability of aerial vehicles, as compared to ground vehicles. Therefore, simulation frameworks will play a pivotal role in development of autonomous solutions for air taxis [3–5]. The key factor in the utility of such frameworks is their ability to emulate real world conditions with high fidelity. Prior simulation solutions for autonomous vehicles address important concerns for air taxis [6–10], however, the existing solutions do not cover the wide range of requirements for end to end autonomy. Leveraging a photorealistic environment simulator built for ground autonomous vehicles [11], this work introduces AirTaxiSim for autonomous air taxis operating in high fidelity urban environments. An overview of the simulation framework is presented in Figure 1.

CARLA simulator [11] provides high fidelity photorealistic environment and physics simulation for ground vehicles. However, CARLA lacks support for aerial vehicles. To overcome this limitation, AirTaxiSim integrates high fidelity physics models for aerial vehicles within the CARLA environment. This includes a full-scale hybrid fixed-wing vehicle with vertical take-off and landing (VTOL) capabilities. This vehicle model is adapted from a simulator developed by National Aeronautics and Space Administration Langley Research Center [12]. Another vehicle supported is MiniHawk-VTOL [13, 14], a small tiltrotor aerial vehicle with VTOL capabilities. Therefore, AirTaxiSim natively supports the two primary use cases for such a simulation framework, *i.e.*, high fidelity simulation for a life scale vehicle, and a practical option for sim-to-real conversion by using Minihawk. AirTaxiSim is designed to be extensible and supports other aerial vehicle models with minimal integration overheads.

Leveraging docker containers, AirTaxiSim is designed to be highly modular. The modularized components use Robot Operating System (ROS) Noetic [15] as the middleware to communicate. Combining the flexibilities provided by docker containers, ROS, and a custom hierarchical configuration layer, AirTaxiSim allows easy selection between redundant components, *e.g.*, selecting one of different path planning algorithm implementations. Therefore, AirTaxiSim is built for easy integration of novel solutions to specific tasks to validate and benchmark specific solutions, operating as part of the overall end to end system. Furthermore, simulation techniques provide perfect alternatives for certain tasks, facilitating the validation of individual components without interference from errors in other components. The configuration layer allows for easy definition of specific evaluation scenarios, including various initial states of the vehicle, infrastructure, environmental conditions, and ground plus aerial traffic.

This paper presents multiple case studies showcasing some of the ways the simulator can be used in advancing autonomy solutions for air taxis (Section IV). These case studies include training for machine learning solutions, validation of safety solutions, and performance benchmarking. AirTaxiSim enables experimentation and validation of autonomy solutions for autonomous air taxis and facilitate research in robust and scalable solutions for urban air mobility. This is especially suited for validation of solutions to safety challenges in learning-enabled autonomy for aerial vehicles [16, 17]. Preliminary versions of AirTaxiSim have been used to for various purposes in prior works, including validation of safety solutions [18], generate synthetic data for model training [19], and integration with verification framework [20]. By providing a flexible high-fidelity platform, AirTaxiSim supports the advancement of autonomy in air taxi systems, accelerating the transition to safe and efficient real-world deployment.

The key **contributions** of this work are:

- AirTaxiSim, a software in the loop simulation framework for autonomous air taxis.
- Functional and implementation descriptions, providing guidance for use and customization of this simulator for autonomous air taxi research.
- Case studies presenting some of the of use cases supported by AirTaxiSim, including dataset generation and benchmarking.

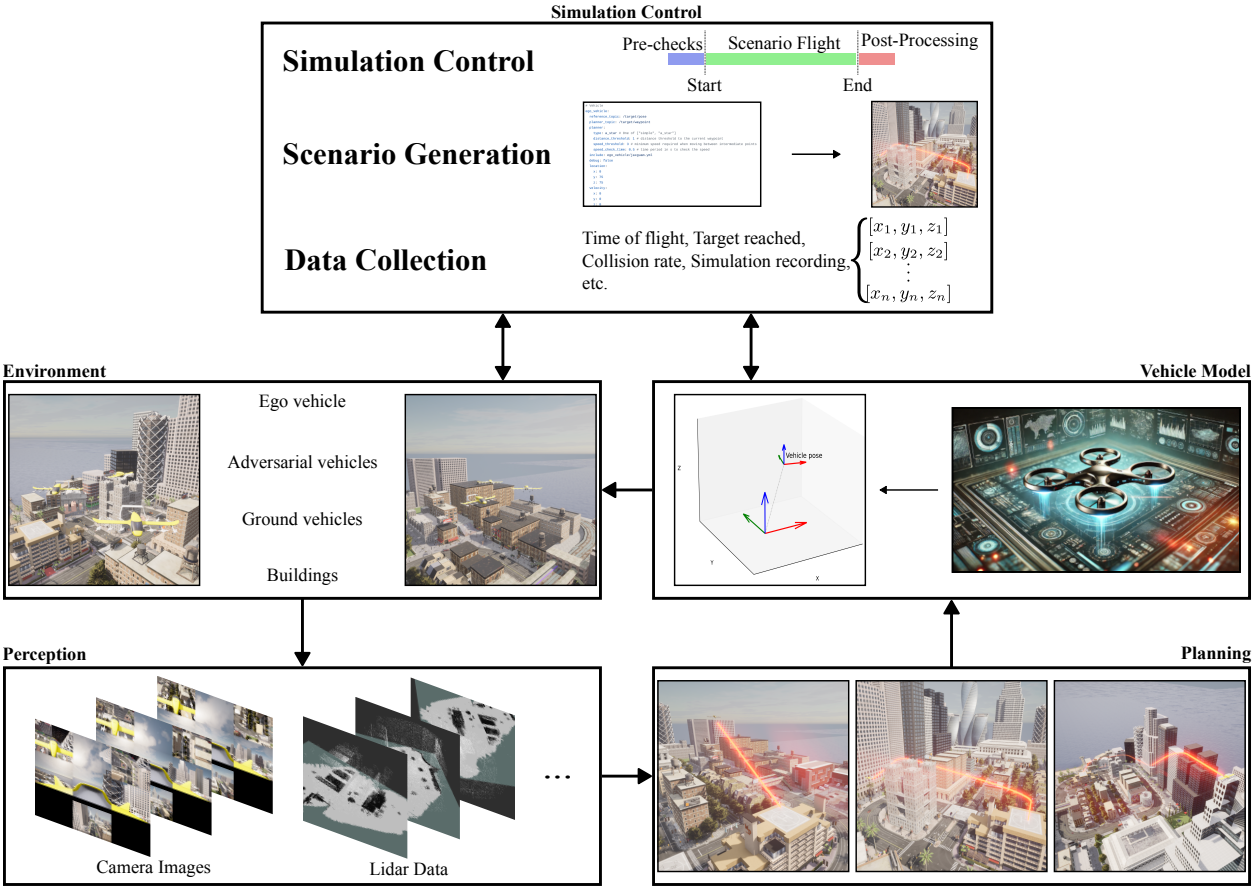

Fig. 1 AirTaxiSim Overview: a) Environment includes elements like the ego vehicle, adversarial vehicles, ground vehicles, and buildings; b) Perception processes sensor data; c) Planning generates paths for the ego vehicle; d) Vehicle Model simulates ego vehicle physics; e) Simulation Control manages the simulation and collects data for safety and performance metrics.

II. Related Work

In this section we discuss the prevalent simulation solutions for autonomous vehicles and aerial vehicles. To the best of our knowledge, AirTaxiSim is the first open-source simulator for autonomous air taxis in realistic urban environments with end to end autonomy support, providing a valuable resource for autonomous air taxi research and development.

A. Autonomous Ground Vehicle Simulators

Autonomous ground vehicle simulators play a critical role in the development and testing of autonomous driving technologies, offering a safe and cost-effective environment for algorithm validation and system integration. Several well-established simulators have been developed to cater to the diverse needs of autonomous vehicle research.

CARLA [11] is a widely-used simulator for autonomous ground vehicles, providing a photorealistic, high-fidelity environment with detailed urban settings and traffic dynamics. It supports a range of sensors for testing autonomous driving systems, including lidar, radar, and cameras. Our work builds on CARLA by integrating aerial vehicle models into its simulation framework, allowing us to extend its capabilities to urban air mobility applications. LGSVL Simulator [21] provided similar functionality as CARLA, however, LGSVL Simulator was deprecated in 2022. Therefore, CARLA is used as the environment simulator in this work.

Autoware [22] is the leading open-source research platform for autonomous vehicles, integrating various sensors and algorithms for perception, planning and control. It is often used with CARLA or LGSVL Simulator, which provide a high-fidelity environment for testing vehicle dynamics, sensor simulations, and hardware-in-the-loop systems. This integration allows for the validation of autonomous driving systems under complex, realistic conditions.

Baidu Apollo [23] offers a comprehensive ecosystem for autonomous driving, including a powerful simulation system. The platform supports the testing of vehicle software in virtual environments, simulating diverse scenarios such as urban traffic and adverse weather. Apollo's simulation tools are particularly useful for validating machine learning models related to path planning, perception, and decision-making.

While simulators like Autoware [22], Baidu Apollo [23], and CARLA [11] are highly effective for ground vehicle simulation, they focus on ground-based systems, which limits their use for multi-modal transportation systems like air taxis. This gap highlights the need for simulators that can integrate both ground and aerial vehicle dynamics, enabling the development of urban air mobility solutions.

B. Aerial Vehicle Simulators

Prior aerial vehicle simulators have primarily focused on the physics of the vehicle itself and the direct effects of environmental factors, such as wind, on vehicle dynamics. While these simulators are valuable for understanding basic flight mechanics, autonomous air taxis operating in cluttered urban environments face additional challenges, including obstacle detection and avoidance, complex traffic scenarios, and real-time decision-making in dynamic conditions.

AirSim [24] is a widely-used simulator for testing aerial vehicles, particularly drones, in realistic environments. While it offers high-fidelity simulation for aerial vehicle dynamics, it is primarily designed for open or rural environments, with limited support for simulating complex, dynamic urban traffic. AirSim does not natively integrate urban traffic or multi-modal interactions, which are critical for urban air mobility (UAM) applications like air taxis. In contrast, our work builds on a simulator designed for urban environments (e.g., CARLA), incorporating both aerial and ground vehicle dynamics to enable testing in dense, urban traffic scenarios, which are essential for the safe operation of autonomous air taxis in real-world settings.

X-Plane [25], another popular simulator, provides a comprehensive simulation platform for both aircraft and unmanned aerial systems (UAS). It offers a high level of fidelity in terms of aerodynamics, aircraft systems, and environmental interactions. X-Plane is often used for flight training and vehicle design, but it also serves as a valuable tool for testing autonomous systems, especially in structured environments. However, like many other simulators, X-Plane focuses primarily on vehicle dynamics and lacks built-in support for multi-agent simulations and complex urban environments.

Recent studies have enhanced aerial simulators to better address the complexities of urban air mobility (UAM). For instance, Lu *et al.* [6] developed a multi-agent simulation to study the dynamics of autonomous taxis in urban settings, analyzing travel economics and environmental impacts in the context of UAM systems. Liu *et al.* designed a simulation for autonomous aircraft maintenance scheduling [7], while Naser *et al.* focused on the efficiency of air taxis in comparison to traditional taxicabs [8]. These studies highlight the growing interests in simulating air taxi operations in complex urban environments.

Additionally, Barbarino *et al.* [9] presented high-fidelity aeroacoustic simulations of VTOL aircraft operating in urban air mobility scenarios, addressing challenges related to noise pollution and flight efficiency. Similarly, Martín-Lammerding *et al.* [10] explored high-density air traffic scenarios with a multi-UAS simulator, emphasizing the need for coordination and collision avoidance in crowded airspace. Altun *et al.* [26] developed a comprehensive flight testing and simulation infrastructure for advanced air mobility (AAM), focusing on airspace management and vehicle performance, whereas our work extends this by integrating both aerial and ground vehicle dynamics in high-fidelity urban environments, enabling the simulation of complex multi-modal transportation scenarios.

In summary, while existing aerial vehicle simulators provide valuable insights into the dynamics and performance of autonomous aircraft, they often lack the necessary capabilities to model complex urban environments within photorealistic environments. This work builds upon these simulators by incorporating both aerial and ground vehicle dynamics, addressing the unique challenges of urban air mobility through high-fidelity simulations that enable the testing of multi-modal transportation systems in dense, dynamic traffic scenarios. This integrated approach provides a critical tool for testing the safety and efficiency of autonomous air taxi systems in urban settings.

C. Urban Aerial Datasets

Several datasets featuring overhead aerial imagery captured by drones provide valuable resources for developing perception systems for aerial vehicles operating in urban environments. For example, the DOTA dataset [27] offers images of urban settings with bounding box annotations for various vehicle categories. Similarly, Sun et al. [28] introduce a dataset containing aerial images of urban areas, which is useful for object detection and classification tasks. However, these datasets are limited in scope for developing fully autonomous air taxi systems, as they primarily focus

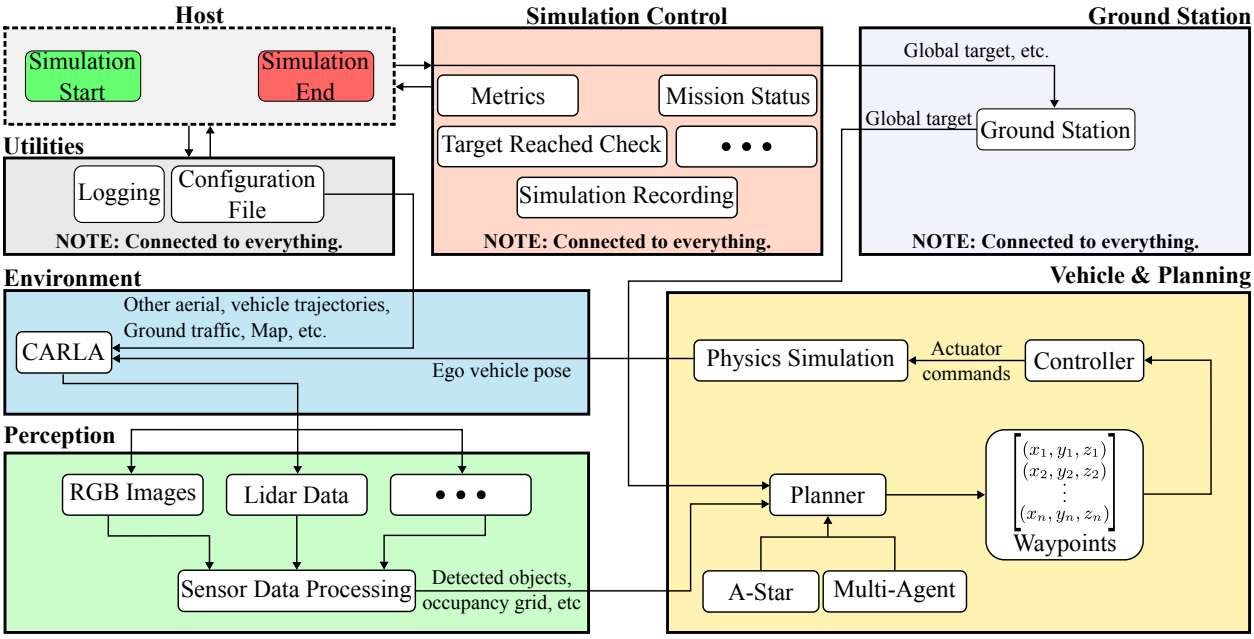

Fig. 2 Detailed system architecture of the air taxi simulator, showcasing key components and their interactions. While the three containers at the top connect to multiple other modules, these connections are omitted for clarity to avoid overcrowding the figure.

on perception tasks and do not support other critical components such as planning and control. Moreover, these datasets do not fully capture the complexity of urban environments that autonomous air taxis will face. For instance, they lack the presence of other aerial vehicles, which is essential for autonomous air taxi operation. Additionally, a robust air taxi system must be trained on data that includes failure scenarios, such as collisions, to recognize and avoid such hazards. Such unsafe scenarios are vastly underrepresented in real-world datasets, including those for ground vehicles [29, 30].

Numerous large-scale datasets are available for autonomous ground vehicles, such as the KITTI dataset [31] and the Waymo Open Dataset [32]. However, these datasets are tailored for ground vehicles and thus do not include critical information needed for air taxis, such as wind velocity, overhead viewpoints, or vertical navigation data.

While existing real-world datasets provide valuable resources that can support some aspects of air taxi research, they fall short in addressing the comprehensive needs of air taxi systems, especially for dense urban environments. Furthermore, the generation of synthetic datasets specifically designed for air taxi research — capable of simulating the unique challenges of urban air mobility — remains an under explored area.

Although various simulation tools exist for autonomous ground vehicles, there is a significant gap in simulation tools for air taxis. Our work addresses this gap by introducing a simulation-based approach to generate synthetic data to support the development of end-to-end air taxi systems.

III. Simulator Description

This section describes the major components of AirTaxiSim, including their functionality, implementation, and design decisions. The system architecture of AirTaxiSim is illustrated in Figure 2.

A. Infrastructure

AirTaxiSim is backed by a custom software infrastructure, built to leverage popular software frameworks. This infrastructure, and the use of popular frameworks, facilitates ease of use, rapid development, modularity, flexibility, configuration, and robustness of simulation components.

1. Host Scripts

AirTaxiSim is launched by running the main script on a host machine. This script parses and loads the configuration files (Section III.A.2) that define the simulation run, including which simulator components or services to run. A set of shared utilities are used by the host script and made available to each containerized service (Section III.A.3). The utilities provide essential functionalities to support the simulation framework. This includes container services management, configuration management, logging, and containers to host communication.

Based on configuration options, the host scripts set up containers for each service, including their file systems and any one-time setups, including builds for any ROS packages. From that point the host scripts start each service, and in conjunction with a simulator control service (Section III.A.4), monitors and controls the simulation run. The host scripts also aggregate and analyze final results. All these behaviors are configurable using the custom configuration framework, described in Section III.A.2.

To minimize software dependencies on the host machine, the host scripts primarily use standard python modules, with the only exception being a third party logging package, loguru [33]. Of special note is the fact that the host machine does not need to install ROS, which significantly improves the portability of this simulation framework.

2. Configuration Files

Configuration files serve as the interface for defining the simulation scenarios the user wishes to run and select the automation components for these simulations. The simulator natively supports a wide range of options, with support to define custom scenarios to fit novel use cases. This configuration plane is designed to be extensible, allowing the introduction of new options as needed to support diverse tasks and emergent use cases. The hierarchical structure of the configuration plane, enabled by an include mechanism, further enhances the modularization and configurability of AirTaxiSim. The current configurable options include

- Containerized services to include in a simulation run, their one time setup and run commands
- Data flows, *e.g.*, to select between different trajectory sources to control the ego vehicle
- The ego vehicle's* initial state
- Simulation target behavior and goal, *e.g.*, landing target
- Initial positions and motion patterns for other aerial vehicles and obstacles
- CARLA configuration
- Ground traffic levels
- Weather conditions

This level of granularity offers users the flexibility to create a vast number of unique simulation scenarios, accommodating diverse needs. The combinatorial explosion of possible configurations ensures the simulator's utilities across a wide range of applications and use cases. Below is an example snippet from a configuration file:

```yaml
# Test
map: Town02_Opt
spectator_follows_ego_vehicle: false

# Vehicle
ego_vehicle:
  reference_topic: /target/pose
  planner_topic: /target/waypoint
  planner:
    type: a_star # One of ["simple", "a_star"]
    distance_threshold: 1 # distance threshold to the current waypoint
    speed_threshold: 3 # minimum speed required when moving between intermediate points
    speed_check_time: 0.5 # time period in s to check the speed
  include: ego_vehicle/jaxguam.yml
  debug: false
  location:
    x: 0
    y: 75
    z: 75
  velocity:
    x: 0
    y: 0
    z: 0
```

*Ego vehicle refers to the primary vehicle that is autonomously controlled and observed for the simulation.

As shown, the configuration files are designed to be intuitive, allowing users to easily set up and modify simulation parameters without requiring deep familiarity and technical expertise. This accessibility ensures that researchers and practitioners alike can efficiently operate the simulator and explore various customizations. Also of note is line 14 in the snippet above. It shows an include statement, which imports configuration options from the included file. Such a hierarchical structure improves the readability of the configuration files. More importantly, it allows selection of features at higher levels of granularity. For example, by modifying line 14 above from `include: ego_vehicle/jaxguam.yml` to `include: ego_vehicle/minihawk.yml`, the simulation would use the minihawk vehicle. The vehicle specific configuration file includes all options specific to the vehicle as well as corresponding defaults.

3. Modular Services

All simulator components, with the exception of host scripts, are encapsulated in docker containers [34] and run as docker compose services [35]. A service here is a docker container that fulfills a specific functionality within the simulation framework. Most services are a ROS package with any accompanying algorithm and implementations, *e.g.*, YOLOv5, A* planner, and vehicle models, however, this is not a requirement. The services communicate over network sockets, with ROS providing a publisher subscriber model built over the network sockets. Any inter-service communication that is not a ROS topic, needs to define its own protocol. An example of this in current AirTaxiSim is the CARLA control using CARLA Python API [36]. Encapsulating the services in containers facilitates the use of services with differing dependencies and requirements within the same simulation run and simplifies the inclusion of new solutions with their specific dependencies. The following services are included in the base simulator:

- Simulation Control — monitors the simulation, records the simulation, coordinates with host scripts, etc.
- Ground Station — represents the physical ground station as a separate container, sets the global mission target, etc.
- Environment — simulates the environment around the vehicle using CARLA.
- Perception — container responsible for retrieving sensor data.
- Vehicle & Planning — simulates the vehicle physics and performs path planning.

The modular architecture of the simulator enables fault injection at various stages of the simulation pipeline. Users can easily define and customize a wide range of faults, such as perturbations in actuator commands or the injection of noise into sensor data (*e.g.*,, camera or LiDAR). Rather than providing a fixed set of predefined faults, the simulator is designed to be transparent and extensible, allowing users to implement fault models that suit their specific use cases. Detailed examples and guidance for implementing custom faults are provided in the simulator's documentation.

4. Simulation Control

Simulation control is a core containerized service that coordinates with the host scripts to manage the execution of the simulation. It includes a ROS package that monitors critical mission metrics, such as whether the global target has been reached or if a collision has occurred.

Reaching the global target is determined by evaluating data from various ROS topics that publish the simulation states. These values are compared against the global target specified in the configuration file for the current simulation. For example, if the simulation target is successful landing, and the ego vehicle reaches that target, this module records metrics like time to completion, landing velocity, landing accuracy, *etc.* and communicates to the host scripts that the simulation has completed. The host scripts then either terminate the simulation and aggregate results, or trigger the next simulation run as dictated by configuration options.

Another key feature of the simulation control service is its ability to record simulation runs as rosbags [37]. This feature can be toggled in the configuration file, allowing users to decide whether to record the simulation and which topics to record. When enabled, a rosbag file is generated that captures all data published to selected ROS topics.

B. Environment

The simulation environment is built on a customized version of CARLA, providing air taxi specific assets within CARLA's high-fidelity representation of urban settings. AirTaxiSim adds landing pads and 3D animation models for air taxis to CARLA. CARLA provides a rich set of features to emulate urban environments, including detailed road networks, dynamic traffic, and realistic urban obstacles such as pedestrians, buildings, and other vehicles. CARLA also provides a flexible infrastructure for defining and controlling objects, that AirTaxiSim leverages to emulate cluttered environments for air taxis. The environment integrates with the vehicle model, as described in Section III.C.

The environment service is a core component of our simulation pipeline, responsible for initiating the CARLA

simulation and configuring its key elements. It spawns the ego vehicle, additional aerial vehicles, and generates ground traffic according to specifications in the configuration file. Parameters such as the simulation map, the ego vehicle's starting position, and the trajectories and starting points of adversarial vehicles are retrieved from the configuration file and set accordingly.

CARLA emulates sensors for the vehicle, including cameras and LiDAR. The sensors are capable of emulating effects of weather conditions and noise, as supported natively by CARLA. CARLA generates the output for each sensor attached to the ego vehicle in the simulation. This sensor data is then published to corresponding ROS topics.

Additionally, leveraging the recording capabilities of our simulator, as detailed in Section III.A, we enable scenarios involving multiple aerial vehicles, each backed by its own physical engine. This is achieved by pre-recording the trajectory of an aerial vehicle and then replaying it in subsequent simulation runs, where the recorded path and vehicle model are integrated into the environment. This approach allows multiple complex aerial vehicles to operate simultaneously, increasing the difficulty of the aerial navigation task for the ego vehicle.

AirTaxiSim, therefore, maintains all rich features provided by CARLA, while enhancing it with the assets required for air taxi simulation. Current version of AirTaxiSim is based on CARLA 0.9.15. Work is underway to support Carla 0.10.0, which is based on Unreal Engine 5.5 [38]. This update will significantly improve the visual fidelity of the simulated environment, reducing the sim-to-real gap.

C. Vehicle Model

The aerial ego vehicle in AirTaxiSim utilize dedicated physics engines to compute the dynamics of the vehicle, simulating the physical model of the ego vehicle with high fidelity. Given reference trajectory from a path planning module, the vehicle's pose is computed by a high fidelity vehicle model, transformed into CARLA's coordinate frame, and communicated to CARLA. Within CARLA the ego vehicle is defined as an object without physics or gravity simulation, allowing the external vehicle model to precisely control the ego vehicle's behavior. Within CARLA, the ego vehicle essentially teleports with each pose update from the external vehicle model. CARLA and the vehicle model are synchronized to ensure a seamless simulation loop. The vehicle model updates the vehicle pose at the same or higher frequency than CARLA updates to the environment, avoiding any lags and minimizing jitter.

AirTaxiSim includes models for two aerial vehicles.

- The first is a life scale air taxi, adapted from the Generic Urban Air Mobility (GUAM) vehicle simulator developed by National Aeronautics and Space Administration Langley Research Center [12, 39]. We converted the original continuous time Simulink model to a time-stepped model implemented in Python using JAX [40]. This converted model was evaluated to ensure that any quantization error is minimal and there is no significant loss in fidelity [41]. The converted model has significantly higher computational performance to the extent that it can support real-time execution. It enables use cases where multiple air taxis, controlled by independent instances of the vehicle model, operate simultaneously within a simulation run. This converted model is also available for independent use [42].
- The second aerial vehicle supported is Minihawk VTOL [13, 14] which utilizes Gazebo simulation [43]. Minihawk can be 3D printed and assembled to support autonomous control [13]. Therefore, Minihawk provides a highly accessible avenue for sim-to-real validation within workflows for research and development of autonomy solutions for VTOL vehicles.

Following the examples of these aerial vehicle models users can integrate their own vehicles. AirTaxiSim configuration framework is designed to support such flexibilities, as described in Section III.A.2.

Some physics engines for aerial vehicles support interaction with environmental factors, such as wind disturbances and ground effect modeling. While the current version of AirTaxiSim does not include these capabilities for the two supported vehicle models, support for environmental interaction is planned for a future release.

The low-level control for the vehicle is integrated within the vehicle models, *e.g.*, PID Control [44, 45] or $\mathcal{L}_1$ Adaptive Control [46, 47]. This design choice is governed by two factors. First, the low-level control loop runs at a much higher frequency (*e.g.*, 400 Hz) compared to other autonomy components (typically 10 Hz). Second, the low-level control implementation and paramterization is highly specialized to the vehicle models based on different reference vehicles and specific designs [39]. Therefore, while flexibility and modularity have been prioritized across most system components, for low-level control, we have opted for a more integrated approach by embedding it as a sub-module of vehicle models, due to the two key considerations outlined in this paragraph.

The high fidelity vehicle simulators combined with 3D models for these vehicles within CARLA provide a realistic simulation of aerial vehicles within the CARLA environment.

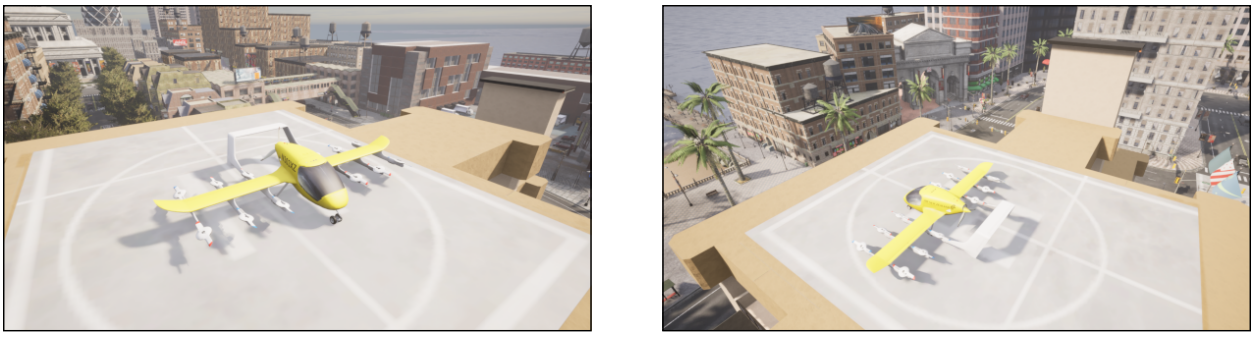

Fig. 3 Simulation example of an air taxi resting on a vertipad, illustrating a key operational scenario for urban air mobility.

D. Autonomy

The vehicles' autonomy stack is comprised of perception and planning components. Perception modules process sensor data, including RGB images and LiDAR-generated point clouds, which may originate from multiple sensors operating concurrently, extracting relevant information for the planning module. The planning module combines this processed data with additional inputs, such as the ego vehicle's current pose, to compute an optimal path toward the target. For instance, the perception module can provide data like an occupancy grid map or landing zone detection, which the planning module uses to navigate. The components are designed to be modular and therefore easily replaceable by alternate implementations. This modular approach supports flexibility, allowing users to test and integrate custom algorithms for perception and planning. All existing autonomy modules communicate using ROS topics, leveraging the standardized framework for data exchange.

Aiding the research and development of perception, planning, and control modules for autonomous air taxis is the central goal of AirTaxiSim. Therefore, while the simulator includes baseline solutions for each, its modular structure facilitates selection between alternative implementations. The baseline perception includes 3D obstacle detection using camera and LiDAR sensor data. Yolov5 [48] for landing pad detection and Depth Clustering [49, 50] for verifiable obstacle detection [18, 51, 52] are included. The baseline path planning implements obstacle avoidance using A* algorithm [53] and RRT (rapidly exploring random trees).

As example, we provide a baseline autonomy stack comprising perception and planning components, centered around octomap generation from point cloud data captured by multiple lidar sensors. The point clouds are first merged through a dedicated node and then passed to the octomap package to construct a 3D occupancy grid map. This map is used by an RRT-based planner to generate a path to the target location. The planner supports dynamic replanning: if updated occupancy data indicates a potential collision along the current path, the planner automatically recomputes a new path using the updated information.

Additionally, as an ongoing long-term goal we will incorporate research solutions developed using AirTaxiSim within the public repository to promote continuous collaborative research.

IV. Case Studies

In this section we showcase some of the use cases and scenarios that are supported by AirTaxiSim, demonstrating its versatility and relevance to real-world urban air mobility challenges. These scenarios are designed to reflect realistic situations that may arise when deploying air taxis in urban environments.

A. Landing on a Vertipad

A fundamental safety-critical requirement for any air taxi is the ability to land safely and accurately on a designated vertipad. This scenario reflects the envisioned operation of urban air taxi hubs, often referred to as *vertiports*. To illustrate this capability, we simulate an air taxi starting in flight and successfully landing on a vertipad, as shown in Figure 3. The landing maneuver includes communication with the ground station to determine the landing target, trajectory planning, low level control of the rotors of the vehicle and high fidelity simulation of vehicle pose under the applied rotor control and environmental conditions.

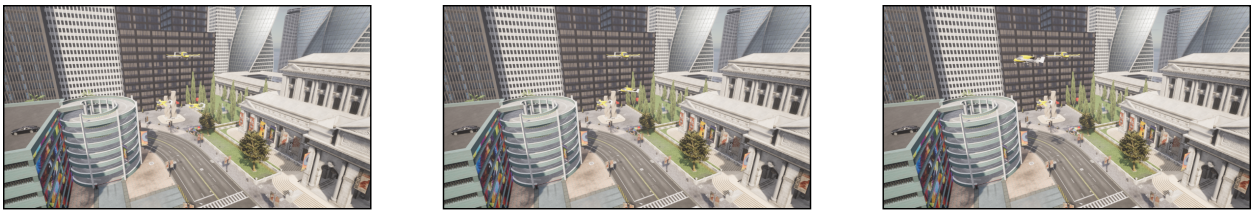

Fig. 4 Simulation of an air taxi navigating a cluttered urban environment, surrounded by various obstacles such as other air taxis, buildings, and trees.

B. Operating in a Cluttered Environment

A critical challenge for urban air taxis is operating safely in highly cluttered environments. Such scenarios can involve a dense array of obstacles, including other air taxis, buildings, trees, and other infrastructure typically found in urban settings. The ability to navigate effectively in these situations is essential for ensuring mission success and passenger safety. Our simulation pipeline allows the ego vehicle to be surrounded by various types of obstacles and evaluates its ability to safely navigate out of the cluttered zone to complete its mission. This scenario illustrates the importance of robust perception, planning, and collision avoidance systems. Figure 4 shows an example of an air taxi successfully navigating through a cluttered environment, demonstrating the effectiveness of its autonomy module in maintaining safety and efficiency. A similar scenario was used to validate safety solutions in a prior work [18].

C. Landing on the Ground

In certain scenarios, an air taxi may need to perform an emergency landing. These situations can arise when critical systems, such as the GPS or vision modules, fail due to errors or damage. Such failures necessitate a prompt and safe landing to ensure the safety of passengers, pedestrians, and surrounding vehicles. Unlike controlled rooftop landings, emergency landings in urban environments are more likely to occur on flat ground. However, these areas are typically shared with ground vehicles, introducing unique challenges that require precise coordination. To simulate this scenario, we model the interaction between air taxis and ground vehicles during an emergency landing. Ground vehicles halt their motion when the air taxi descends below a critical altitude, creating a safe buffer zone. Simultaneously, the air taxi modulates its descent rate to provide adequate time for nearby vehicles to react. This simulation highlights the importance of responsive systems for both aerial and ground vehicles in effectively managing unexpected events.

Notably, this emergent interaction is a natural outcome of our simulator's internal structure, rooted in the CARLA framework. Although CARLA was initially designed for ground vehicles, its adaptability allows us to explore aerial vehicle scenarios, showcasing the versatility and robustness of our simulation pipeline.

D. Training of Machine Learning Solutions

AirTaxiSim supports the generation of synthetic training data for learning-based solutions in aerial vehicles. This is especially useful for scenarios that are infeasible to replicate in the real world, such as environments cluttered with flying obstacles or hazardous conditions. While we do not release a dataset as part of this work, AirTaxiSim includes tools to extract data from both predefined and user-defined scenarios. Therefore, AirTaxiSim can be used to generate datasets for training and refinement of learning-based solutions.

However, AirTaxiSim can go beyond typical supervised learning to enable a fully automated active learning paradigm. Active learning involves dynamic adaptation of the learning data, based on external heuristics [54]. The ability of AirTaxiSim to support automated active learning was demonstrated in a recent work [19].

An example reinforcement learning pipeline for an end-to-end flying agent is also available.

E. Metrics and Benchmarks

AirTaxiSim currently collects metrics such as collisions, time to completion, maximum velocity, and landing velocity. Its architecture has been configured to support flexible collection of additional task- and algorithm-specific metrics. Although benchmark definitions are still under development, the simulator's architecture already supports flexible integration of user-defined metrics for evaluation.

V. Conclusion and Future Work

This paper presents a high-fidelity simulator for autonomous aerial vehicles operating in urban environments. The simulator integrates key components of the autonomy pipeline — including perception, planning, control, and a photorealistic environment renderer based on CARLA. AirTaxiSim is designed to be highly customizable and adaptable to support diverse scenarios and use cases. This adaptability includes ease of incorporating custom physics engines for new aerial vehicle models. This simulator is designed to support research and development in urban aerial autonomy, especially in light of recent advances in eVTOL technology and the growing interest in urban air mobility.

Future development of AirTaxiSim is expected to be collaborative and open-source, driven by the requirements to support specific research efforts. This model ensures that in addition to supporting short terms research goals, each development effort enhances the simulation framework with new support and capabilities. We encourage the research community and practitioners to collaborate with us by sharing their inputs and enhancements to AirTaxiSim.

Major planned enhancements include

- Upgrade Carla version to 0.10.0 that is built on Unreal Engine 5.5 [38]. This upgrade will vastly improve the visual fidelity of the simulated environment.
- Some physics engines for aerial vehicles support interaction with environmental factors, such as wind disturbances and ground effect modeling. While the current version of AirTaxiSim does not include these capabilities for the two supported vehicle models, support for environmental interaction is planned for a future release.
- Extend the scenario configuration to support existing scenario definition languages, *e.g.*, Scenic [55].

Acknowledgments

This material is based upon work supported by the National Aeronautics and Space Administration (NASA) under the cooperative agreement 80NSSC20M0229 and University Leadership Initiative grant no. 80NSSC22M0070, and the National Science Foundation (NSF) under grant no. CNS 1932529 and ECCS 2311085. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

We also thank Michael Acheson from NASA for sharing GUAM simulator [12].

References

[1] Ward, K. A., Winter, S. R., Cross, D. S., Robbins, J. M., Mehta, R., Doherty, S., and Rice, S., "Safety systems, culture, and willingness to fly in autonomous air taxis: A multi-study and mediation analysis," *Journal of Air Transport Management*, Vol. 91, 2021, p. 101975.

[2] LLC, W. A., "Wisk Aero," https://wisk.aero/, 2024.

[3] Fadaie, J., "The state of modeling, simulation, and data utilization within industry: An autonomous vehicles perspective," *arXiv preprint arXiv:1910.06075*, 2019.

[4] Garrow, L. A., German, B. J., and Leonard, C. E., "Urban air mobility: A comprehensive review and comparative analysis with autonomous and electric ground transportation for informing future research," *Transportation Research Part C: Emerging Technologies*, Vol. 132, 2021, p. 103377.

[5] Mishra, S., and Palanisamy, P., "Autonomous advanced aerial mobility—An end-to-end autonomy framework for UAVs and beyond," *IEEE Access*, Vol. 11, 2023, pp. 136318–136349.

[6] Lu, M., Taiebat, M., Xu, M., and Hsu, S.-C., "Multiagent spatial simulation of autonomous taxis for urban commute: Travel economics and environmental impacts," *Journal of Urban Planning and Development*, Vol. 144, No. 4, 2018, p. 04018033.

[7] Liu, Y., Wang, T., Zhang, H., Cheutet, V., and Shen, G., "The design and simulation of an autonomous system for aircraft maintenance scheduling," *Computers & industrial engineering*, Vol. 137, 2019, p. 106041.

[8] Naser, F., Peinecke, N., and Schuchardt, B. I., "Air taxis vs. taxicabs: A simulation study on the efficiency of UAM," *AIAA Aviation 2021 Forum*, 2021, p. 3202.

[9] Barbarino, M., Petrosino, F., and Visingardi, A., "A high-fidelity aeroacoustic simulation of a VTOL aircraft in an urban air mobility scenario," *Aerospace Science and Technology*, Vol. 125, 2022, p. 107104.

[10] Martın-Lammerding, D., Astrain, J., and Córdoba, A., "A multi-UAS simulator for high density air traffic scenarios," *Proceedings of the VEHICULAR*, 2022.

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

[12] Acheson, M. J., Cook, J. W., and Simmons, B. M., "Generic Urban Air Mobility (GUAM) Simulation v1.1," https://github.com/nasa/Generic-Urban-Air-Mobility-GUAM, 2024.

[13] Carlson, S., "MiniHawk-VTOL," https://github.com/StephenCarlson/MiniHawk-VTOL, 2022.

[14] RoboWork, "Aerial Robotics," https://github.com/robowork/aerial_robotics, 2023. Accessed: 2024-11-18.

[15] Open Robotics, "ROS Noetic," https://wiki.ros.org/noetic, 2020.

[16] Singh, V., Hari, S. K. S., Tsai, T., and Pitale, M., "Simulation driven design and test for safety of ai based autonomous vehicles," *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 122–128.

[17] Bansal, A., and Sha, L., "Towards Certifiable Safety in Learning-Enabled Autonomous Systems: A Perspective," 2025.

[18] Bansal, A., Zhao, Y., Zhu, J., Cheng, S., Gu, Y., Yoon, H. J., Kim, H., Hovakimyan, N., and Sha, L. R., "Synergistic perception and control simplex for verifiable safe vertical landing," *AIAA Scitech 2024 Forum*, 2024, p. 1167.

[19] Rasul, A., Tasnim, H., Yoon, H.-J., Bansal, A., Wang, D., Hovakimyan, N., Sha, L., and Voulgaris, P., "Bayesian Data Augmentation and Training for Perception DNN in Autonomous Aerial Vehicles," *AIAA SciTech 2025 Forum*, 2025, p. 0933.

[20] Bansal, A., Wang, D., Yeghiazaryan, M., Li, Y., Tao, C., Yoon, H.-J., Arora, P., Papachristos, C., Voulgaris, P., Mitra, S., et al., "Verification and Validation of a Vision-Based Landing System for Autonomous VTOL Air Taxis," *AIAA SCITECH 2025 Forum*, 2025, p. 1322.

[21] Rong, G., Shin, B. H., Tabatabaee, H., Lu, Q., Lemke, S., Možeiko, M., Boise, E., Uhm, G., Gerow, M., Mehta, S., et al., "LGSVL Simulator: A High Fidelity Simulator for Autonomous Driving," *2020 IEEE 23rd International conference on intelligent transportation systems (ITSC)*, IEEE, 2020, pp. 1–6.

[22] Foundation, A., "Autoware: The open-source software for autonomous driving," https://www.autoware.org/, 2024. Accessed: 2024-11-18.

[23] Baidu, "Apollo," https://apollo.auto/, 2024. Accessed: 2024-11-18.

[24] Shah, S., Dey, D., Lovett, C., and Kapoor, A., "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," *Field and Service Robotics: Results of the 11th International Conference*, Springer, 2018, pp. 621–635.

[25] Laminar Research, "X-Plane," https://www.x-plane.com, 2023. Accessed: 2024-11-18.

[26] Altun, A. T., Hasanzade, M., Saldiran, E., Guner, G., Uzun, M., Fremond, R., Tang, Y., Bhundoo, P., Su, Y., Xu, Y., et al., "The development of an advanced air mobility flight testing and simulation infrastructure," *Aerospace*, Vol. 10, No. 8, 2023, p. 712.

[27] Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., and Zhang, L., "DOTA: A large-scale dataset for object detection in aerial images," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3974–3983.

[28] Sun, H., Guo, J., Meng, Z., Zhang, T., Fang, J., Lin, Y., and Yu, H., "EVD4UAV: An Altitude-Sensitive Benchmark to Evade Vehicle Detection in UAV," *arXiv preprint arXiv:2403.05422*, 2024.

[29] Bansal, A., Singh, J., Verucchi, M., Caccamo, M., and Sha, L., "Risk ranked recall: Collision safety metric for object detection systems in autonomous vehicles," *2021 10th Mediterranean Conference on Embedded Computing (MECO)*, IEEE, 2021, pp. 1–4.

[30] Marathe, A., Ramanan, D., Walambe, R., and Kotecha, K., "WEDGE: A multi-weather autonomous driving dataset built from generative vision-language models," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 3318–3327.

[31] Geiger, A., Lenz, P., and Urtasun, R., "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite," *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[32] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhang, Y., Shlens, J., Chen, Z., and Anguelov, D., "Scalability in Perception for Autonomous Driving: Waymo Open Dataset," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[33] Delgan, "Delgan/loguru: Python logging made (stupidly) simple," https://github.com/Delgan/loguru, 2017.

[34] Docker Inc., "Docker," https://www.docker.com/, 2013.

[35] Docker Inc., "Docker Compose," https://docs.docker.com/compose/, 2014.

[36] CARLA Team, "Python API reference," https://carla.readthedocs.io/en/latest/python_api/, 2025.

[37] Open Robotics, "Rosbag," https://wiki.ros.org/rosbag, 2020.

[38] Rowe, M., "CARLA 0.10.0 Release with Unreal Engine 5.5!" https://carla.org/2024/12/19/release-0.10.0/, 2024.

[39] National Aeronautics and Space Administration (NASA), "NASA Urban Air Mobility (UAM) Reference Vehicles," https://sacd.larc.nasa.gov/uam-refs/, 2024.

[40] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q., "JAX: composable transformations of Python+NumPy programs," , 2018. URL http://github.com/jax-ml/jax.

[41] Fan, C., "Efficiently predicting and repairing failure modes via sampling," https://uofi.app.box.com/s/ey543nfa836u49pjasmvxj3ss7c772vp/file/1451288867656, 2023.

[42] So, O., "JAX-GUAM," https://github.com/oswinso/jax_guam, 2024. Accessed: 2024-11-18.

[43] Koenig, N. and Howard, A., "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, Vol. 3, 2004, pp. 2149–2154 vol.3. https://doi.org/10.1109/IROS.2004.1389727.

[44] Bennett, S., "Development of the PID controller," *IEEE Control Systems Magazine*, Vol. 13, No. 6, 1993, pp. 58–62.

[45] Bauersfeld, L., and Ducard, G., "Fused-PID control for tilt-rotor VTOL aircraft," *2020 28th Mediterranean Conference on Control and Automation (MED)*, IEEE, 2020, pp. 703–708.

[46] Cao, C., and Hovakimyan, N., "Design and analysis of a novel l1 adaptive controller, part i: control signal and asymptotic stability," *2006 American Control Conference*, IEEE, 2006, pp. 3397–3402.

[47] Kaminer, I., Pascoal, A., Xargay, E., Hovakimyan, N., Cao, C., and Dobrokhodov, V., "Path following for small unmanned aerial vehicles using L1 adaptive augmentation of commercial autopilots," *Journal of guidance, control, and dynamics*, Vol. 33, No. 2, 2010, pp. 550–564.

[48] Zhang, Y., Guo, Z., Wu, J., Tian, Y., Tang, H., and Guo, X., "Real-time vehicle detection based on improved yolo v5," *Sustainability*, Vol. 14, No. 19, 2022, p. 12274.

[49] Bogoslavskyi, I., and Stachniss, C., "Fast Range Image-Based Segmentation of Sparse 3D Laser Scans for Online Operation," *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2016. URL http://www.ipb.uni-bonn.de/pdfs/bogoslavskyi16iros.pdf.

[50] Bogoslavskyi, I., and Stachniss, C., "Efficient Online Segmentation for Sparse 3D Laser Scans," *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 2017, pp. 1–12. URL https://link.springer.com/article/10.1007%2Fs41064-016-0003-y.

[51] Bansal, A., Kim, H., Yu, S., Li, B., Hovakimyan, N., Caccamo, M., and Sha, L., "Verifiable obstacle detection," *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)*, IEEE, 2022, pp. 61–72.

[52] Bansal, A., Kim, H., Yu, S., Li, B., Hovakimyan, N., Caccamo, M., and Sha, L., "Perception simplex: Verifiable collision avoidance in autonomous vehicles amidst obstacle detection faults," *Software Testing, Verification and Reliability*, Vol. 34, No. 6, 2024, p. e1879.

[53] Persson, S. M., and Sharf, I., "Sampling-based A* algorithm for robot path-planning," *The International Journal of Robotics Research*, Vol. 33, No. 13, 2014, pp. 1683–1708.

[54] Settles, B., "Active learning literature survey," 2009.

[55] Vin, E., Kashiwa, S., Rhea, M., Fremont, D. J., Kim, E., Dreossi, T., Ghosh, S., Yue, X., Sangiovanni-Vincentelli, A. L., and Seshia, S. A., "3D Environment Modeling for Falsification and Beyond with Scenic 3.0," *International Conference on Computer Aided Verification*, Springer, 2023, pp. 253–265.