IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.Doi Number

Agri-fly: simulator for uncrewed aerial vehicle flight in agricultural environments

JIAMING ZHA[1], TEAYA YANG[1], and MARK W. MUELLER[1]
[1]High Performance Robotics Laboratory (HiPeRLab), UC Berkeley, Berkeley, CA 94720, USA

Corresponding author: Jiaming Zha (e-mail: jiaming_zha@berkeley.edu).

This work was supported by the Agriculture and Food Research Initiative (AFRI) Competitive Grant no. 2020-67021-32855/project accession no. 1024262 from the USDA National Institute of Food and Agriculture.

ABSTRACT We present Agri-fly, a simulator for Uncrewed Aerial Vehicles (UAVs) with a focus on agricultural applications. In addition to general flight simulator features, such as dynamics modeling and sensor synthesis, Agri-fly comes with an autonomous stack for under-the-canopy flight and tools to create high-fidelity agricultural landscapes. By leveraging detailed plant models and tools for scene customization, Agri-fly excels at generating realistic agricultural environments for UAV operations. This capability facilitates in-depth analysis of the visual and spatial relationships between UAVs and plants, creating avenues for refining operational strategies in agricultural environments. The source code of Agri-fly and detailed usage instructions are available at: **github.com/muellerlab/agri-fly**.

INDEX TERMS Uncrewed Aerial Vehicles (UAVs), Robotics, Simulation, Agricultural Application

I. INTRODUCTION

In recent years, the application of UAVs has substantially boosted agricultural production. UAVs have improved processes such as plant health monitoring, disease detection, and pesticide spraying. These advancements have led to responsive management of plant health and optimized application of chemicals. A survey on recent developments in UAV technologies in the field of agriculture can be found in [1].

While most current UAV applications are limited to above-the-canopy (or above-the-treetops) operations, there is a rising demand for intricate under-the-canopy tasks, such as close-up plant monitoring, autonomous pollination, and yield estimation. However, developing these capabilities poses numerous challenges. For instance, wind-moved thin branches complicate obstacle detection and avoidance, whereas constant growth of plants makes previous spatial maps obsolete rapidly, necessitating continuous mapping updates. As a result, reliable under-the-canopy flight is hard to achieve, and frequent crashes during testing can cause significant delays in the development process. Given these challenges, the need for a risk-free simulation tool that provides realistic agricultural environments for autonomous flight validation becomes clear.

In this paper, we introduce "Agri-fly", an open-source UAV simulator designed for autonomous operation in agricultural scenes. Agri-fly combines generalized UAV simulation features, such as flight dynamics modeling and autonomous flight controllers, with software tools to create, manage, and visualize high-fidelity agricultural scenes. This enables users to test and refine autonomous flight strategies in virtual environments mirroring real-world agricultural landscapes. Additionally, Agri-fly can help validate flight missions, like visual plant health scanning, that depend on accurately understanding the spatial and visual relationships between UAVs and plants. To achieve this, we have developed a customized flight library responsible for dynamics simulation and autonomous flight control, as well as a toolset in Unity [2] to populate and manage agricultural worlds with high-fidelity plant models created by the Helios 3D plant modeling framework [3]. In addition, we connect the flight library and the agricultural world in Unity with an information bridge based on AirSim's AirLib [4]. An example scene simulated in Agri-fly is shown in Fig. 1.

In the following subsections, we will briefly overview existing UAV simulators and agricultural robotics simulators in the literature, and discuss what Agri-fly can achieve by bridging the gap between the two groups.

A. UAV SIMULATORS IN THE LITERATURE

The literature features a variety of UAV simulators, each designed for specific use cases and emphases. Here, we provide a summary of popular UAV simulators and their key characteristics. We begin with AirSim, which partially serves as the foundation for this work, and then explore other influential simulators.

FIGURE 1. An example agricultural scene in Agri-fly: an aerial vehicle flies autonomously in a simulated orchard built with high-fidelity almond models. Top: simulated scene generated with high-fidelity plant models. Bottom: an experimental flight scene in an almond orchard.

AirSim is an open-source simulator for autonomous vehicle operation studies [4]. It has various versions and releases. The primary version leverages the Unreal Engine [5] for rendering, and it has a dedicated vehicle physics engine paired with an environment manager. Designed with vision-based machine learning research in mind, it emphasizes delivering high-quality visuals to reduce the sim-to-real gap. Based on the AirSim platform, many UAV simulators with specific expertise are developed, such as [6] for visual and language navigation, [7] for software/hardware-in-the-loop system validation, and [8] for UAV-based landslide monitoring. Two years post its original debut, an experimental version employing Unity was created. It marked a great attempt to make AirSim more accessible, given Unity's lower hardware requirement and greater adaptability. The Agri-fly incorporates the Unity wrapper and bridge code from AirSim's Unity version to link different parts of the simulator.

Flightmare [9] is a flight simulator that features a rendering engine and a synthetic sensor suite built within the Unity framework. Its primary advantage is adaptability, allowing users to select different configurations based on their hardware setup to achieve the desired trade-off between simulation speed and precision.

FlightGoggles [10], similar to Flightmare, employs a

FIGURE 2. A close-up shot of the under-the-canopy autonomous flight experiment in an almond orchard, validating the autonomous stack used in the Agri-fly simulator.

Unity-based rendering and a sensor synthesis suite. It comes with a custom physics engine and offers VR-based first-person flight visualization. In addition, it can interact with real vehicles in a motion capture lab. It is designed to support research on vision-based control and odometry algorithms.

RotorS [11] is a Gazebo-based [12] flight simulator with accurate physics simulation for quadcopters and helicopters. Its rendering engine is rooted in OpenGL, yielding less vivid, but more computationally efficient visuals than FlightGoggles and Flightmare.

B. SIMULATORS FOR ROBOTS IN AGRICULTURAL ENVIRONMENTS

Navigating agricultural environments is complex for robots, and various simulators are created to assist in such endeavors. A comprehensive review of existing agricultural robotics simulators is detailed in [13]. Here, we provide a simplified overview of the work with goals similar to those of Agri-fly.

A robotic vegetable harvesting simulator is presented in [14]. This work introduces a fully simulated environment in V-REP [15] for experimenting with sensors and manipulators for sweet pepper harvesting. Meanwhile, a Gazebo-based simulator for fruit mapping, which helps validate a method for creating accurate fruit maps using Simultaneous Localization And Mapping (SLAM) is proposed in [16]. Moreover, a simulation for a robot fleet in precision agricultural environments is proposed in [17]. The tool provides a dual-interface system for configuring the crop field, robot fleet, and associated sensors and actuators, and thus facilitates the assessment of precision farming fleet strategies in a 3D simulated world.

C. AGRI-FLY: UAV SIMULATOR FOR AGRICULTURAL TASKS

From the review above, we can observe that existing UAV simulators prioritize flight functionalities and visualization rendering but place less emphasis on the surrounding environments in which they operate. In contrast, for agricultural robots, interaction (through image capture or physical contact) is paramount, and determining the spatial relationship between the robots and plants is critical. As a result, a virtual environment that closely resembles real-world agricultural scenarios is essential. To bridge the gap between general UAV simulators and agricultural simulators, we created Agri-fly, a flight simulator tailored for UAV operations in agricultural

environments. Like Flightmare and FlightGoggles, Agri-fly uses Unity for rendering and synthetic visual sensing. In addition, we developed a custom C++ flight library for fast and high-fidelity quadcopter physics simulation, state estimation, and autonomous flight control.

Agri-fly's advantages are threefold:

1) Autonomous under-the-canopy flight pipeline

Agri-fly features an autonomous flight stack based on the work in [18] for under-the-canopy flights, which we have experimentally validated in an orchard (see Fig. 2 and the attached video). The stack includes a motion planner that senses the surrounding environment with a depth camera and generates dynamically feasible trajectories that avoid collisions with obstacles. This frees users, particularly those who focus on agricultural productivity rather than specific flying robotic technicalities, by eliminating the need to develop autonomous flight control systems before validating robotic missions in agricultural environments.

2) Agricultural model-realism

Existing UAV simulators prioritize vision-based machine learning applications, focusing heavily on improving visual rendering quality, while placing less emphasis on the specific models within the scene. Meanwhile, recent robotics technology advancements have demonstrated the potential of using depth cameras for robust outcomes in robotics planning and control [19], [20]. Depth cameras are less susceptible to color and light interference. However, their effectiveness heavily relies on the shape and spatial relationships of the objects they detect. Consequently, using models that closely resemble real-world items is advantageous for bridging the sim-to-real gap. Agri-fly features an extensive collection of high-fidelity plant models developed with the Helios modeling framework [3], which applies the Weber and Penn algorithm [21] to generate trees with branch structures that closely mimic those found in nature. This enables Agri-fly to accurately replicate real-life shapes and spatial relationships in agricultural scenes, helping narrow the sim-to-real gap.

3) Various environments with tools for customization

Agri-fly comes with four pre-built examples of realistic agricultural environments: a fruit orchard, a vineyard, a greenhouse, and a vertical farming pod, covering a range of representative agricultural scenes. In addition, Agri-fly provides a comprehensive toolkit for the autonomous generation of detailed virtual environments, facilitating user-driven customization through the Unity editor to enhance the adaptability and versatility of the simulator. Additionally, the simulator records the positions and orientations of all agricultural components, including leaves, fruits, branches, and trunks. This feature provides users with accurate data to examine intricate spatial relationships, and helps create ground truth annotations for synthetic visual data generated in the simulator.

With these advantages, Agri-fly serves as a virtual UAV training ground for precision farming, enabling effective validation and rapid iteration. By generating high-quality agricultural scenes with realistic plant models, Agri-fly ensures that simulated flights build confidence for successful real-world operations. Additionally, Agri-fly supports mission planning for tasks that require precise spatial coordination between the UAV and crops. For example, with Agri-fly, users can accurately predict whether an agricultural target, such as a fruit, is visible to the UAV or obscured by foliage. Such spatial reasoning and practice would be challenging without scenes and plant models that accurately mirror real-life conditions.

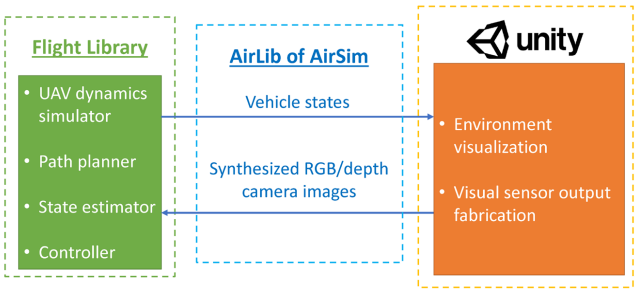


FIGURE 3. Structural diagram of the Agri-fly simulator. The simulator has three key components: the flight library in charge of dynamics simulation and autonomous flight, the Unity-based agricultural scene manager and visualizer, and the information bridge connecting the two components.

II. SIMULATOR STRUCTURE

Agri-fly is composed of three major components: a custom flight library, a Unity-based agricultural scene manager with visualizer, and an information bridge connecting them. Fig. 3 illustrates the structure of Agri-fly and the interactions between these key components.

The custom flight library we created, positioned on the left in the Fig. 3, includes a UAV flight dynamics simulator and a UAV autonomy stack for state estimation, path planning, and control. Modular in design and connected through ROS [22], this library offers flexibility for users to easily modify or interchange individual modules as necessary.

On the right side of the diagram, the agricultural scene manager and visualizer are responsible for creating and visualizing agricultural environments. This component is also responsible for generating synthetic visual sensor data, including RGB and depth images.

Centrally located in the diagram is the information bridge, which utilizes AirSim's AirLib and its Unity simulator wrapper. This tool integrates our flight library with Unity, ensuring synchronization between the UAV's state in the flight dynamics simulator and the Unity scene. It also helps pipe the visual sensor data to the autonomous motion planner.

For the remainder of this section, we will provide a detailed introduction to each component and highlight the features of Agri-fly related to these components.

A. FLIGHT LIBRARY

The flight library implements two key functionalities, dynamics simulation and computation for autonomous flight.

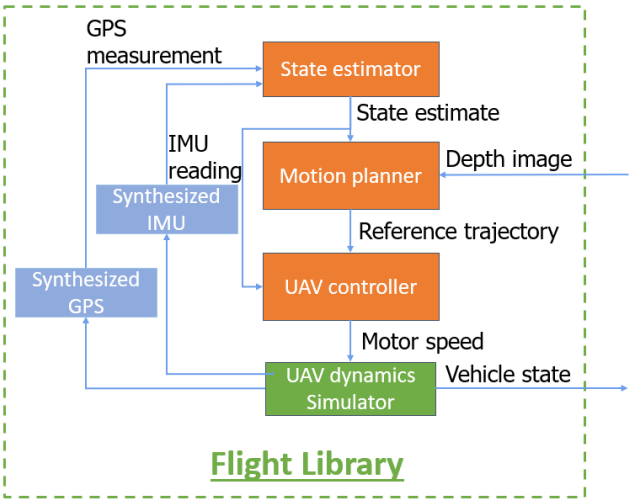


FIGURE 4. **Detailed structure of the flight library, which includes a flight dynamics simulator and an autonomous flight stack with a state estimator, a motion planner, and a flight controller.**

A structural block diagram of the flight library is shown in Fig. 4. Notably, each block in this diagram represents a single code module. The modular design of the flight library allows for easy editing or replacement of each block. Below is a brief description of the default modules of the flight library included in the released Agri-fly repository.

1) State estimator

The state estimator employs a Kalman Filter that utilizes synthetic GPS measurements. It follows a predictor-corrector structure, which predicts the vehicle states by integrating the vehicle's Inertial Measurment Unit (IMU) reading at a high frequency and then corrects the state using a virtual GPS measurement generated based on the simulated UAV state.

2) Collision avoidance planner

The flight library incorporates a depth-image-based collision avoidance motion planner, which is originally proposed in [20]. This planner detects obstacles from depth image, reformulates depth information into pyramids for quick collision checks, and subsequently generates a collision-free trajectory using a sampling-based method. An experimental validation of this planner in a forest environment can be found in [18].

3) Flight controller

The flight library employs a cascaded feedback controller to help the simulated UAV track the planned trajectory. A block diagram showcasing the structure is in Fig. 5. A position controller outputs desired total thrust and thrust direction, whereas an inner attitude controller computes desired torques. Finally, a thrust converter maps the total thrust and body torque commands to per-propeller thrust commands (i.e. motor speed commands).

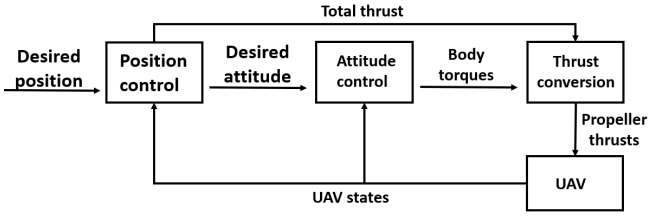


FIGURE 5. **The block diagram of the cascaded UAV controller implemented in the flight library.**

B. AIRLIB-BASED INFORMATION BRIDGE

The objective of the information bridge is to link the agricultural scene in Unity with the flight library. We decide to develop the information bridge based on AirSim's AirLib and its Unity wrapper rather than building it from scratch, as they contain features closely matching our requirements. This information bridge acts as a communication channel, synchronizing the Unity scene with state updates from the dynamics simulator and passing synthesized visual sensor readings to the flight autonomy stack within the flight library.

C. AGRICULTURAL SCENE MANAGER AND VISUALIZER

To create and visualize the scenes in which UAVs operate, we develop the scene manager and visualizer using the Unity platform. The system comprises three main components: a library of high-fidelity plant models, a management tool for placing and tracking these models, and rendering pipelines for visualizing the scenes and generating synthetic sensor outputs. These components are introduced in detail below:

1) High-fidelity plant model and scenes

To reproduce an agricultural scene resembling reality, we employ 3D models generated using the Helios 3D plant modeling framework [3]. The framework creates plant models using the Weber and Penn branching algorithm, resulting in canopies closely resembling what we observe in real trees. An example almond tree model produced with Helios, juxtaposing against a real almond tree can be viewed in Fig. 6. Beyond leaves and branches, we can also incorporate fruits into the models. This can be very helpful for tasks focusing on produce monitoring. In the Agri-fly repository, we have included pre-generated high-fidelity models of different types of plants, ranging from walnut trees to strawberry plants (see Fig. 7). These models can be easily imported into a Unity scene to create user-customized agricultural environment. In addition, Agri-fly comes with four pre-built scenes: an almond orchard, a vineyard, a mini greenhouse, and a vertical farm pod [23]. Users can directly use these scenes to test their missions and can easily modify them by rearranging the position of the models in the Unity editor. Figures of these pre-built scenes can be found in Fig. 8.

2) Agricultural scene management

Agri-fly includes a scene manager script that tracks the ground truth poses (positions and orientations) of all plants

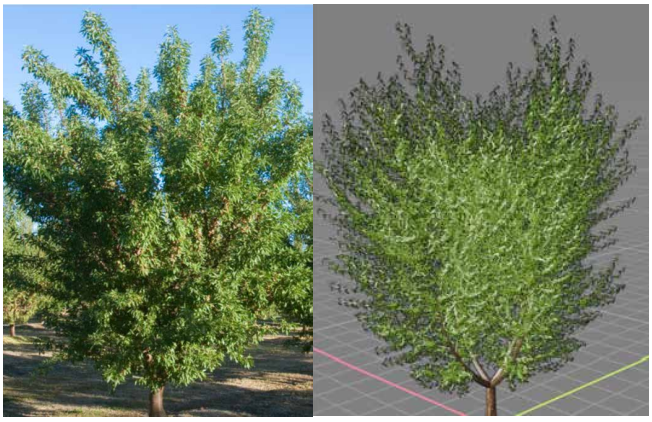


FIGURE 6. We use Helios to generate high-fidelity models that resemble plants in real life. Here we use the almond tree as an example. Left: photo of a real almond tree. Right: the almond tree model generated using Helios.

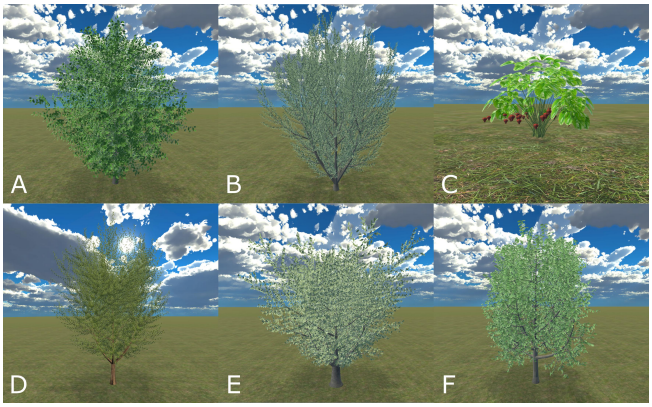


FIGURE 7. Agri-fly comes with a wide range of high-fidelity plant models that can be directly imported into agricultural scenes. Here we show a few as an example: (A) orange; (B) olive; (C) strawberry; (D) almond; (E) lemon; (F) walnut.

FIGURE 8. Agri-fly comes with four pre-built scenes: an orchard, a vineyard, a mini greenhouse, and a vertical farm pod, representing commonly-observed agricultural environments.

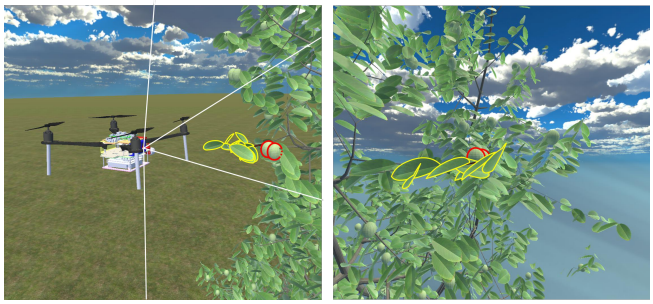


FIGURE 9. The granularity of the agricultural scene manager allows users to analyze the spatial relationships between the drone camera and individual elements, such as leaves and fruits. In this way, users can predict if a fruit will be occluded by leaves during an image scan.

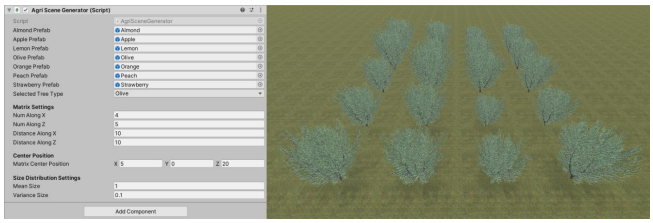


FIGURE 10. The scene generator allows users to specify the plant type, number of plants, lay out and size variance, and autonomously generate the agricultural scene for the user.

in the scene. More importantly, for the high-fidelity plant models utilized in Agri-fly, each model can be dissected into individual elementary components such as branches, leaves, and fruits. This level of granularity is useful for in-depth analysis of spatial relationships. For instance, for projects like plant scanning, this feature enables users to predict whether a fruit might be obscured by foliage, thereby informing drone positioning for data collection (see Fig. 9 for example).

In addition, we have created a scene generator to autonomously populate agricultural scenes. By specifying key parameters such as plant type, orchard dimensions, and the variance in plant sizes, users can quickly create an agricultural environment for UAV operation tests. An illustration of the autonomous agricultural scene generation process is shown in Fig. 10. It can also be used to generate less structured agricultural environments that UAVs have not encountered before, to test the robustness of flight algorithms.

3) Visualization and image capturing tool

The visualization and image capturing tool simulates a third-person camera tracking the UAVs, as well as the UAVs' onboard visual sensors, such as RGB and depth cameras. This allows the autonomous controller in Agri-fly to sense its surroundings and avoid obstacles during flight, and enables users to generate synthetic visual data for analysis or training. We tune the image synthesis pipeline so that it is representative of our real-life experimental vehicle in Fig. 2. The synthetic camera has the same field of view, resolution, focal length, and relative position with respect to the vehicle center. Meanwhile, users can freely position the

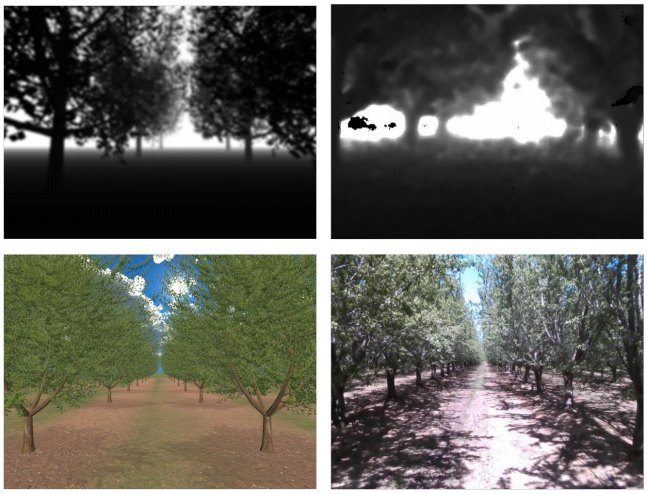


FIGURE 11. Left: generated RGB and depth image through the agri-fly visualizer. Right: RBG and depth image taken by actual sensors for a similar scene in real world.

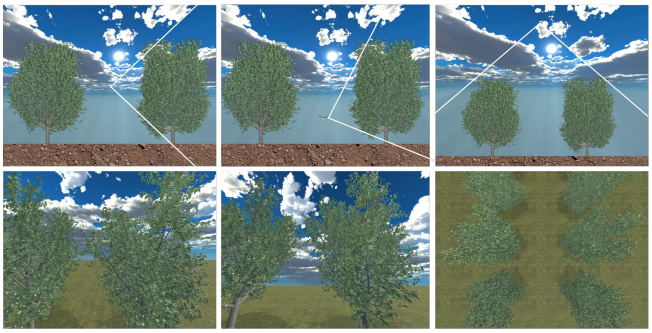


FIGURE 12. Users can easily change the setup of the camera and take pictures of the agricultural scene.

drone's camera within the scene and adjust camera settings. This flexibility enables users to collect images in the virtual agricultural environment they have created, facilitating the creation of customized image datasets tailored to specific agricultural scenarios (see Fig. 12). The scene and the camera footage are both rendered with the Universal Render Pipeline (URP) of the Unity Engine [24], which is used for its user-friendliness and low computational requirements. For image pipelines mimicking onboard RGB and depth cameras, we utilize the Unity ML-ImageSynthesis toolbox [25]. We adjust the rendering pipeline and add a blurring effect to make the synthesized output closely resemble real sensor captures. Fig. 11 presents a comparison between the RGB and depth images synthesized by Agri-fly and those captured by an Intel Real Sense D455 camera [26].

III. DEMONSTRATION EXAMPLE

In this section, we validate Agri-fly with an application example, demonstrating autonomous data collection missions within the simulator. This example highlights the ease of use and operational efficiency of Agri-fly.

First, we set up a walnut orchard environment using our scene manager. We select Helios walnut tree models and configure the orchard's size, tree size variance, and orientation. We then setup the UAV with a front RGB and depth camera for detect-and-avoidance purposes, along with two side cameras for data collection. The default autonomous under-the-canopy flight pipeline of Agri-fly is utilized for this demonstration. We configure a series of waypoints in the orchard via a JSON file, instructing the vehicle to follow these waypoints while autonomously avoiding obstacles such as tree trunks and branches. The vehicle successfully navigates through the orchard autonomously, collecting visual plant data along the way.

FIGURE 13. A sample processed frame from a synthetic onboard image captured during a simulated flight. The white boxes indicate the ground truth for the fruits visible to the camera in this frame. As Agri-fly has ground truth information of the pose of the fruits, we can accurately annotate them without manual labeling.

Fig. 13 shows an example of the image collected during the scanning flight. Thanks to Agri-fly's scene manager, we have the ground truth pose of all agricultural elements like leaves and fruits in the scene. As a result, we can annotate the position of these parts in the picture without manual labeling.

Fig. 14 shows a composite image of the whole simulated flight process, including a top-down shot of the orchard and the vehicle's trajectory marked with a red curve. Data collected in this flight can be post-processed to evaluate the vehicle's recording efficiency and used for other analysis such as yield estimation. In addition to the flight in the almond orchard, we have also simulated the autonomous flights in other pre-built scenes. Videos and footage from all simulated flights are attached to this paper and available at: **https://youtu.be/hMaTrmD2yTc**.

IV. CONCLUSION

In this paper, we introduce Agri-fly, a UAV simulator tailored specifically for agricultural scenarios. While it encompasses standard features of general UAV simulators—such as dynamics simulation, autonomous flight, and sensor data synthesis—Agri-fly places special emphasis on providing an

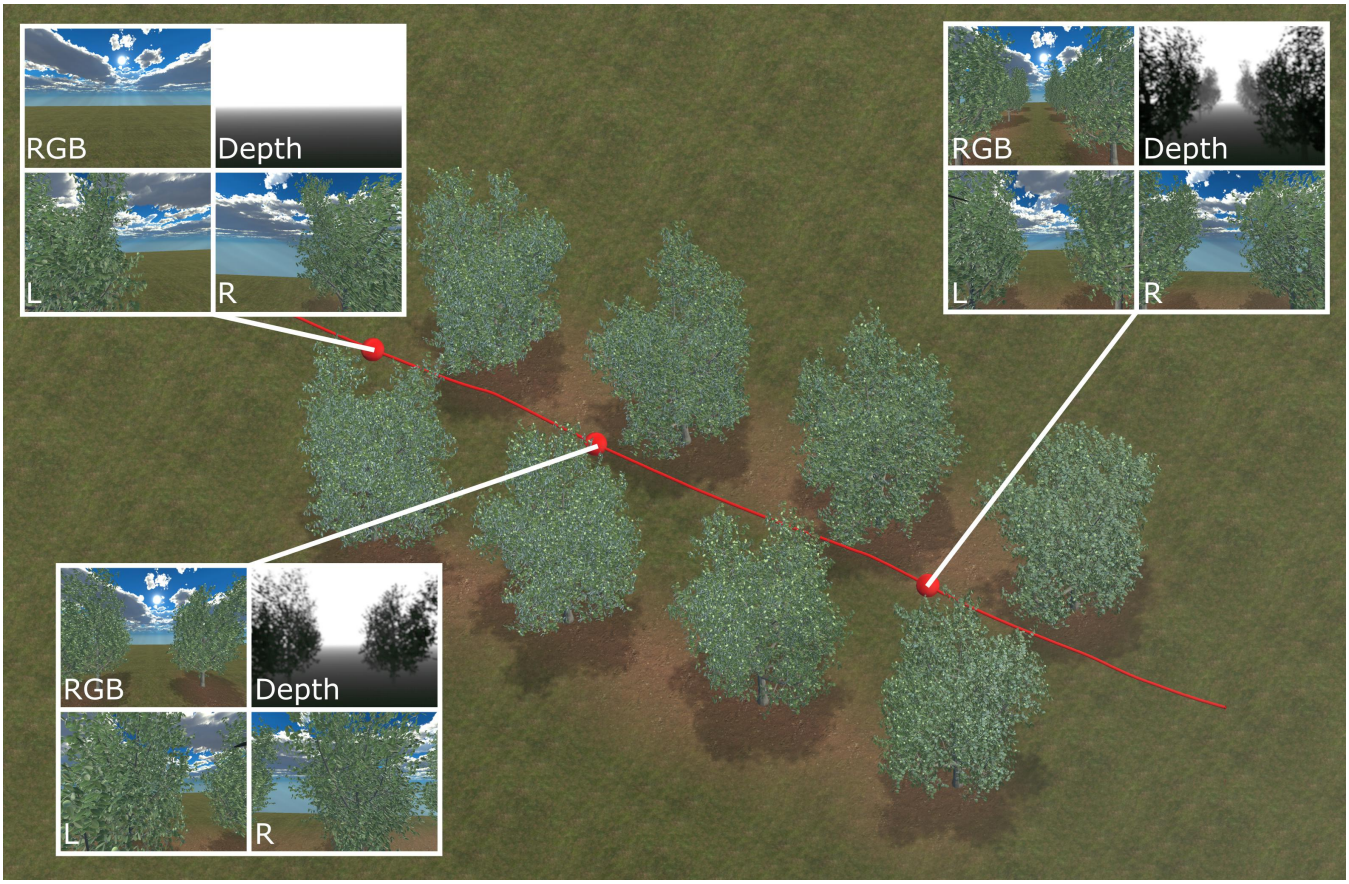


FIGURE 14. An example demonstrating Agri-fly's capabilities: the vehicle is commanded to autonomously fly between two lines of walnut trees while avoiding tree branches and collecting image data. The red line indicates the vehicle's flight path (from right to left). We show image captures from the onboard depth and RGB cameras at the beginning, middle, and end of the flight.

accurate and realistic portrayal of agricultural environments. Leveraging high-fidelity plant models generated through Helios, along with scripts for autonomous scene creation and management, Agri-fly can swiftly create agricultural landscapes for UAV operations. This capability facilitates studies of the interactions between the UAV and its agricultural surroundings, making it well-suited for optimizing operational strategies, especially when the spatial relationships between the UAV and plants are critical.

ACKNOWLEDGEMENTS

This work was supported by the Agriculture and Food Research Initiative (AFRI) Competitive Grant no. 2020-67021-32855/project accession no. 1024262 from the USDA National Institute of Food and Agriculture. The authors would like to thank Brian Bailey for his advice and help in agricultural plant modeling; and Joseph Birtman, Gyu Bae, and Madeline Bumpus for help in the development of Agri-fly.

REFERENCES

[1] J. del Cerro, C. Cruz Ulloa, A. Barrientos, and J. de León Rivas, "Unmanned aerial vehicles in agriculture: A survey," *Agronomy*, vol. 11, no. 2, p. 203, 2021.

[2] Unity Technologies, *Unity*, 2022. [Online]. Available: https://unity.com/

[3] B. N. Bailey, "Helios: A scalable 3d plant and environmental biophysical modeling framework," *Frontiers in Plant Science*, vol. 10, p. 1185, 2019.

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

[5] Epic Games, *Unreal Engine*, 2022. [Online]. Available: https://www.unrealengine.com

[6] S. Liu, H. Zhang, Y. Qi, P. Wang, Y. Zhang, and Q. Wu, "Aerialvln: Vision-and-language navigation for uavs," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 15 384–15 394.

[7] J. Giovagnola, J. B. M. Megías, M. M. Fernández, M. P. Cuéllar, and D. P. M. Santos, "Airloop: A simulation framework for testing of uav services," *IEEE Access*, vol. 11, pp. 23 309–23 325, 2023.

[8] D. Xie, R. Hu, C. Wang, C. Zhu, H. Xu, and Q. Li, "A simulation framework of unmanned aerial vehicles route planning design and validation for landslide monitoring," *Remote Sensing*, vol. 15, no. 24, p. 5758, 2023.

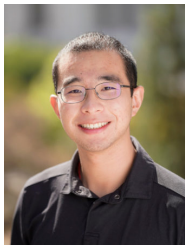
[9] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*. PMLR, 2021, pp. 1147–1157.

[10] W. Guerra, E. Tal, V. Murali, G. Ryou, and S. Karaman, "Flightgoggles: Photorealistic sensor simulation for perception-driven robotics using photogrammetry and virtual reality," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 6941–6948.

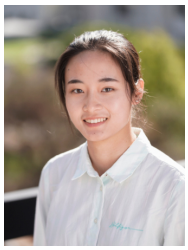
[11] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, "Rotors—a modular gazebo mav simulator framework," *Robot Operating System (ROS) The Complete Reference (Volume 1)*, pp. 595–625, 2016.

[12] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ international conference*

on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566), vol. 3. IEEE, 2004, pp. 2149–2154.
[13] R. R Shamshiri, I. A. Hameed, L. Pitonakova, C. Weltzien, S. K. Balasundram, I. J Yule, T. E. Grift, and G. Chowdhary, "Simulation software and virtual environments for acceleration of agricultural robotics: Features highlights and performance comparison," *International Journal of Agricultural and Biological Engineering*, vol. 11, no. 4, pp. 15–31, 2018.
[14] R. R. Shamshiri, I. A. Hameed, M. Karkee, and C. Weltzien, "Robotic harvesting of fruiting vegetables: A simulation approach in v-rep, ros and matlab," *Proceedings in automation in agriculture-securing food supplies for future generations*, vol. 126, pp. 81–105, 2018.
[15] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 1321–1326.
[16] N. Habibie, A. M. Nugraha, A. Z. Anshori, M. A. Ma'sum, and W. Jatmiko, "Fruit mapping mobile robot on simulated agricultural area in gazebo simulator using simultaneous localization and mapping (SLAM)," in *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*. IEEE, 2017, pp. 1–7.
[17] L. Emmi, L. Paredes-Madrid, A. Ribeiro, G. Pajares, and P. Gonzalez-de Santos, "Fleets of robots for precision agriculture: a simulation environment," *Industrial Robot*, vol. 40, no. 1, pp. 41–58, 2013.
[18] J. Lee, X. Wu, S. J. Lee, and M. W. Mueller, "Autonomous flight through cluttered outdoor environments using a memoryless planner," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 1131–1138.
[19] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision," in *Conference on Robot Learning*. PMLR, 2023, pp. 403–415.
[20] N. Bucki, J. Lee, and M. W. Mueller, "Rectangular pyramid partitioning using integrated depth sensors (rappids): A fast planner for multicopter navigation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4626–4633, 2020.
[21] J. Weber and J. Penn, "Creation and rendering of realistic trees," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, pp. 119–128.
[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "ROS: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.
[23] Cropbox, *Strawberry Cropbox*, 2023. [Online]. Available: https://cropbox.co/
[24] Unity Technologies, *Universal Render Pipeline Documentation*, 2023. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@17.0/manual/index.html
[25] Unity3D, *Unity ML-ImageSynthesis*, 2017. [Online]. Available: https://bitbucket.org/Unity-Technologies/ml-imagesynthesis/
[26] Intel, *Intel RealSense Depth Camera D455*, 2024. [Online]. Available: https://www.intelrealsense.com/depth-camera-d455/



JIAMING ZHA received a B.S. degree from Rice University in 2018, a M.S. degree from UC Berkeley in 2020 and a Ph.D. degree from UC Berkeley in 2023, all in Mechanical Engineering. His research interests are novel mechanical design, control, and path planning algorithms for aerial vehicles.



TEAYA YANG received her Bachelor of Science degree from Cornell University in 2022. She is currently a Ph.D. student at the High Performance Robotics Lab at UC Berkeley. Her current research interests include the control and path planning of aerial vehicles.

MARK W. MUELLER is an associate professor of Mechanical Engineering at UC Berkeley, and runs the High Performance Robotics Laboratory (HiPeRLab). He received a Dr.Sc. and M.Sc. from the ETH Zurich in 2015 and 2011, respectively, and a B.Sc. from the University of Pretoria in 2008. His research interests include aerial robotics, their design and control, and especially the interactions between physical design and algorithms.