Towards Safe and Efficient Through-the-Canopy Autonomous Fruit Counting with UAVs

Teaya Yang[1], Roman Ibrahimov[1] and Mark W. Mueller[1]

***Abstract*— We present an autonomous aerial system for safe and efficient through-the-canopy fruit counting. Aerial robot applications in large-scale orchards face significant challenges due to the complexity of fine-tuning flight paths based on orchard layouts, canopy density, and plant variability. Through-the-canopy navigation is crucial for minimizing occlusion by leaves and branches but is more challenging due to the complex and dense environment compared to traditional over-the-canopy flights. Our system addresses these challenges by integrating: i) a high-fidelity simulation framework for optimizing flight trajectories, ii) a low-cost autonomy stack for canopy-level navigation and data collection, and iii) a robust workflow for fruit detection and counting using RGB images. We validate our approach through fruit counting with canopy-level aerial images and by demonstrating the autonomous navigation capabilities of our experimental vehicle.**

Fig. 1. Experimental vehicle flying autonomously at canopy level while collecting visual data using a RGB camera. This figure is a compound image showing multiple states of the vehicle during flight.

I. Introduction

Fruit counting in orchards helps farmers make management decisions, such as harvest scheduling, labor allocation, and storage strategies [1]. With recent advances in precision agriculture and remote sensing, robotic systems have demonstrated great success in automating the counting process, significantly reducing labor cost and enabling frequent crop inspections.

In particular, fruit counting using sequences of RGB images collected by autonomous systems through computer vision algorithms [2] has enabled robots equipped with low-cost sensors to perform yield estimation and prediction tasks. For instance, [3] presents a method to reliably detect green fruits using RGB images, in contrast to previous techniques that relied on expensive multispectral sensors [4]. Robust fruit detection of holly fruits, which are small and occur in clusters, is demonstrated in [5], further highlighting the capability of modern algorithms to extract valuable information using only colored images. The You Only Look Once (YOLO) algorithm [6] has become a prominent method in fruit detection tasks [5]-[7], and its successive versions continue to lead advancements in this field. Building upon these successes, we incorporate YOLO as the primary detection tool for our proposed autonomous counting method.

Detection alone, however, is not sufficient for providing reliable fruit counting results using visual data. The major challenge in fruit counting with image sequence is the risk of double counting, especially when the same fruit appears multiple times across different frames. In [8], a robust fruit counting framework that integrates detection, tracking, and structure from motion is outlined. Detected fruits are used as features and tracked using the Hungarian Algorithm [9], with the results serving as raw inputs for the structure-from-motion step using COLMAP [10], [11]. We adopt this method in our work, tailoring it to data collected from canopy-level flight, and incorporate up-to-date tracking techniques. Specifically, we use ByteTrack [12] in place of the Hungarian Algorithm to better maintain fruit tracking, particularly in cases of low-confidence detections. This approach is crucial when image quality is reduced due to factors such as strong sunlight or motion blur, conditions that are common in outdoor environments.

Despite the successes of computer vision techniques in image processing, reliable data collection using autonomous robotic systems remains a challenge. Previous efforts to ensure the safety of robots include using UAVs to fly above tree canopies [13], [14], enabling rapid data collection thanks to their agility and maneuverability. An alternative solution involves operating ground vehicles [15] between tree rows, which has shown promise in counting large fruits in orchards with wall-like canopies, where ground-level data is sufficient to capture most fruits. Nevertheless, both methods miss critical vantage points that are essential for detecting fruits in trees with dense foliage. Flying at canopy level offers a better solution, as illustrated in Figure 4, which compares simulated fruit counting results in a walnut orchard. While some works focusing on image processing collected data using manually controlled canopy-level flights [16], [17] addressed this challenge by using stereo images for obstacle avoidance through onboard local planning. Our approach offers an alternative solution, relying on fewer onboard sensors by

[1]The authors are with the High Performance Robotics Lab, Department of Mechanical Engineering, University of California, Berkeley, CA 94709, USA.

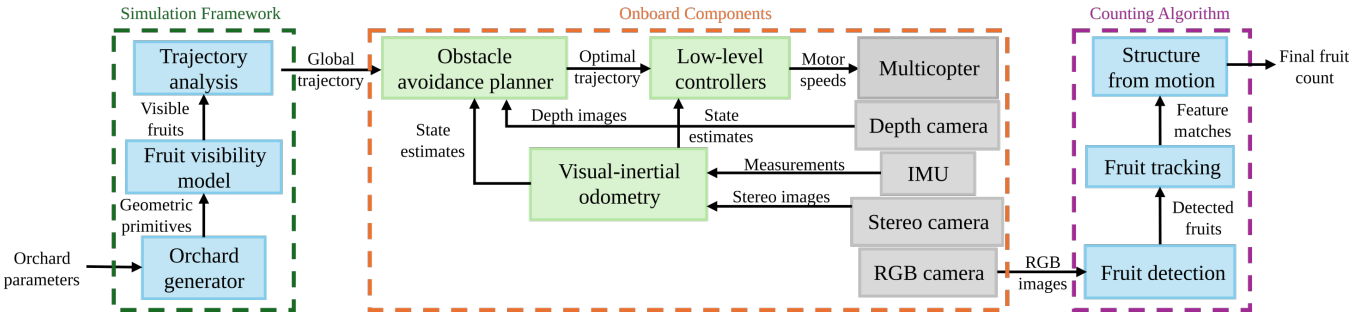

Fig. 2. Block diagram of the proposed autonomous system, consisting of three main components: a simulation framework that leverages specified orchard parameters (such as average tree height, tree spacing, and plant type) for global trajectory planning; onboard autonomy components that ensure safe and efficient data collection during flight; and a post-flight counting algorithm that processes the collected RGB images, producing the final fruit count.

performing state estimation and obstacle detection using the same camera unit, thus ensuring a low-cost, power-efficient data collection vehicle.

In the context of through-the-canopy flight, global trajectory planning methods are extremely limited. While [17] addresses the challenges of safety and navigation, their approach to global trajectory planning assumes fruit trees have wall-like properties and applies coverage-path planning techniques commonly used in structural inspection [18]. These methods are inadequate for orchards with dense foliage, where fruits within the camera's field of view may be easily occluded by leaves and branches. Effective path planning must account for these occlusions to optimize fruit visibility. Previous efforts that consider these occlusion relationships, such as [19] and [20], focus on inspecting individual fruit clusters from a close distance, making these methods unsuitable for orchard-scale planning, where hundreds of fruits may appear in a single image. Moreover, plant science research like [21] has shown that fruit distribution varies across different levels of the tree canopy, making it essential to account for both local and global fruit distributions when planning paths at the orchard level. Our proposed method provides a more comprehensive solution to this path planning problem by incorporating high-fidelity fruit distribution and occlusion models, thereby optimizing flight trajectories for efficient and large-scale data collection.

Our proposed system achieves autonomous fruit counting in orchard environments through three main steps: global trajectory planning based on high-fidelity modeling, data collection using an autonomous vehicle, and fruit detection and counting from onboard RGB images. As illustrated in Figure 2, the system begins with a simulation framework, described in Section II, which uses high-fidelity tree models with realistic architectures to compute visible fruit counts from predefined trajectories. The next step, outlined in Section III, involves data collection with a low-cost experimental vehicle equipped with minimal sensors. This vehicle ensures safe, autonomous operation through vision-based state estimation and depth-image-based obstacle avoidance. Finally, Section IV presents the fruit detection and counting workflow incorporating state-of-the-art tools for detection, tracking, and structure from motion to process collected RGB images.

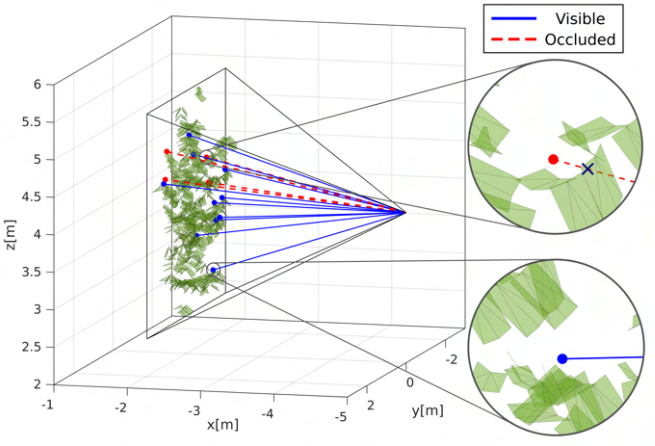

Fig. 3. Illustration of the occlusion checking process. Frustum culling is first performed, where only the geometric primitives contained in the camera's field of view are considered. A ray is cast from the camera to each contained fruit's center, and a fruit is marked as visible only if the ray does not intersect any surrounding mesh triangles.

II. Orchard Simulation for Trajectory Generation

In this section, we introduce the proposed simulation framework, which provides insights into optimal trajectory parameters for data collection. A key element of this framework is the realistic modeling of fruit trees and their architectural structures, as accurate representation of the spatial relationships between fruits and tree components is essential for determining fruit visibility. To achieve this, we use the Helios 3D plant modeling framework [23] and its Weber-Penn [24] tree generation plug-in, which ensures realistic and adaptable tree architecture generation.

Helios includes models for commonly seen fruit trees, such as orange, walnut, almond, and apple, with built-in parameters such as recursive branching levels, leaf angles, and fruit occurrence that mimic real-world characteristics. Additionally, the canopy generator plug-in allows users to input custom parameters such as tree spacing, trunk height, and fruit radius, which can be adjusted to match the conditions of specific orchards. Since real orchards exhibit varying tree types and spacing, these customizations are crucial to creating a simulation environment tailored to specific

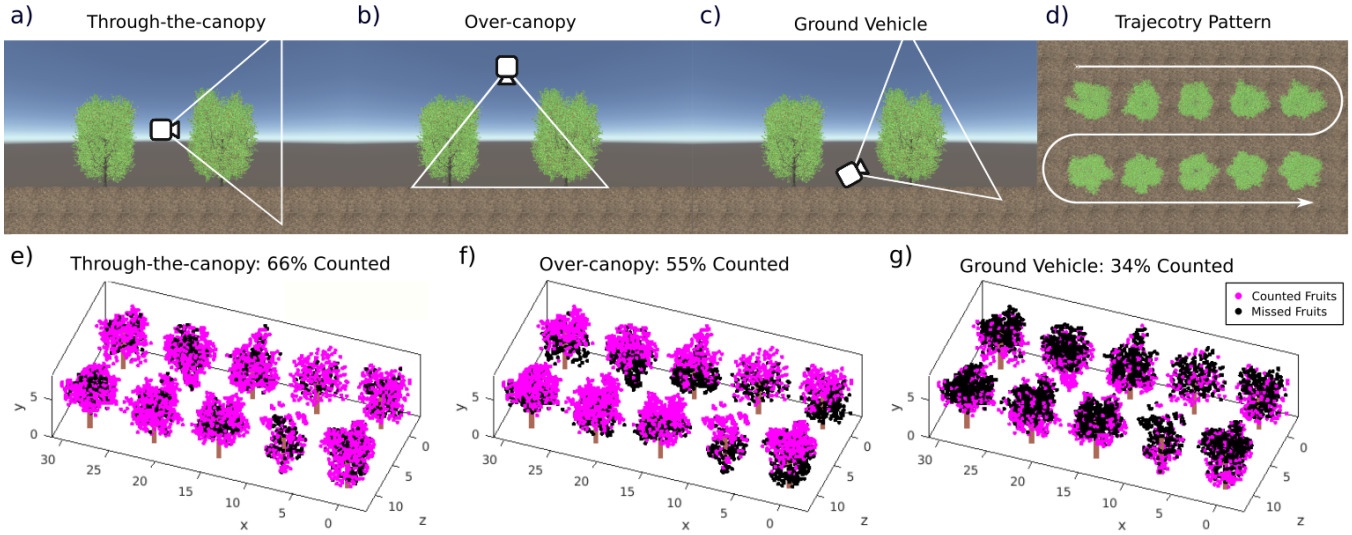

Fig. 4. Simulation results comparing fruit counts using three distinct data collection strategies. (a)-(c) depict camera configurations for through-the-canopy flight, over-canopy flight, and ground vehicle collection, respectively. (d) shows the lawn-mower pattern applied uniformly across all methods for comparison. (e)-(g) present fruit counting results and the corresponding fruit distribution patterns. Data collection using the lawn-mower pattern in the simulation captured 66%, 55%, and 34% of the total fruits for each method, respectively, highlighting the advantage of through-the-canopy UAV data collection.

Algorithm 1 Visible Fruit Counting Algorithm

Require: C: Set of camera configurations
Require: T: Set of trees
Require: F: Set of fruits on tree T_j
Ensure: Detected fruits for each camera configuration C_i

for $C_i \in C$ **do**
  Compute camera frustum $\mathcal{F}(C_i)$
  for $T_j \in T$ and $F_k \in F$ **do**
    if F_k is inside $\mathcal{F}(C_i)$ **then**
      Cast ray $\mathbf{r}_{C_i \to F_k}$
      for each triangle $\Delta \in T_j$ (leaves, branches, trunks) **do**
        Check for ray-triangle intersection [22]
        if no intersection **then**
        $V \leftarrow \{T_j, F_k\}$
      end for
    end if
  end for
end for
$n_{\text{visible}} \leftarrow |\text{set}(V)|$

properties of the target orchard, where autonomous fruit counting will be performed.

In this simulation environment, the smallest subdivisions of all generated trunks, branches, leaves, and fruits are represented as triangles. Figure 3 illustrates selected details focusing on the leaves, while Figure 5a provides examples of full tree renderings. During this generation process, the vertices of these shapes are stored, with each tree assigned a unique ID and each fruit assigned a corresponding fruit ID. With this information about the simulated orchard environment, we can perform fruit visibility analysis for a given camera orientation. We assume that branches, leaves, and trunks may obscure the fruits. Given a path with discrete camera orientations and parameters such as field of view and depth of view, the goal is to determine the total number of visible fruits along the path. This process is presented in Algorithm 1 and a visualization of the occlusion checking process is shown in Figure 3.

The flight trajectories are defined by a set of camera configurations, C. For each camera configuration, C_i, we can construct a frustum $\mathcal{F}(C_i)$ with an associated field of view and depth that simulates the maximum distance at which a fruit may still be captured by the camera's resolution. For each tree T_j from the set of generated trees T, there is an associated set of fruits denoted as F. Frustum culling is first performed, ensuring that only fruits within the camera's view are checked for occlusion. For each F_k that is contained, a ray $\mathbf{r}_{C_i \to F_k}$ is cast from the camera location toward the fruit's center. We then iterate through the mesh triangles of the surrounding geometric primitives, using the Möller–Trumbore algorithm [22] to check for intersections. If the ray intersects any triangle between the fruit and the camera, the fruit is considered occluded. Figure 3 illustrates this process with examples showing how occlusions are determined by intersections with surrounding geometry. A fruit that passes this occlusion check is marked as visible, and its corresponding fruit ID, along with the associated tree ID, is stored in a multiset V. While storing the fruit and tree IDs is not strictly required for counting, doing so provides additional insights into the spatial distribution of visible fruits and their associations with specific trees. This enriched data makes our method valuable not only for fruit counting but also for other applications that benefit from through-the-canopy data collection, such as identifying low-

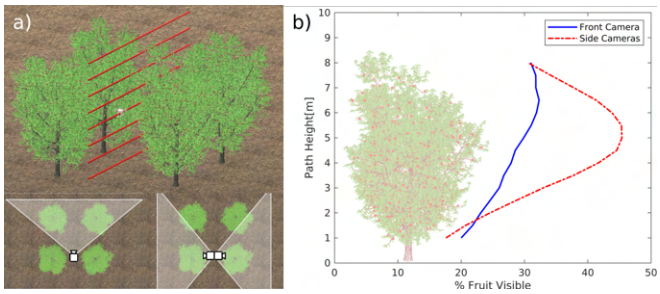

Fig. 5. Simulation results comparing fruit counting performance at different through-the-canopy flight heights and camera configurations. (a) Simulated trajectories tested at heights ranging from 1 m to 8 m, using two configurations: a single front-facing camera and two side-facing cameras. (b) Results showing fruit visibility at different flight heights, with optimal heights of 6.5 m for the front-facing camera and 5 m for the side-facing cameras. The two side-facing cameras achieve a maximum visible fruit coverage of 45.4%, compared to 32.3% for the front-facing camera.

productivity zones or detecting diseases in trees [1].

The proposed simulation framework provides insights into the effectiveness of different methods for collecting fruit counting data. Figure 4 illustrates a simulated orchard with 10 walnut trees, which is used to compare three distinct data collection methods, all following a lawn-mower pattern. The simulation includes three camera configurations: a side-mounted camera on a vehicle flying through the canopy, a downward-facing camera on a UAV flying above the canopy, and an upward-facing camera angled at 30 degrees from a 1-meter-tall ground vehicle. The results suggest that the through-the-canopy flight achieves the highest fruit visibility, capturing 66% of the total fruits visible, compared to 55% for the over-canopy flight and 36% for the ground vehicle, emphasizing the need for canopy-level autonomous data collection for this application. By recovering the positions of the fruits through their recorded fruit IDs, this analysis also reveals the distribution of visible fruits for each method, as shown in purple in Figure 4. In practice, such insights can help users optimize flight paths and data collection strategies before deploying vehicles, reducing experimental costs and the need for multiple iterations.

Building on the through-the-canopy flight strategy, we apply the same occlusion model to optimize trajectory parameters and camera orientations. In Figure 5, we present an example where a 7.6 m by 7.3 m orchard block is repeatedly generated. In each simulation, the vehicle follows a straight-line trajectory down the center of the 7.6 m spacing, and a range of flight heights from 1 m to 8 m is tested. Two different camera configurations are also compared: one front-facing camera and two side-facing cameras. The fruit counting results for each trajectory and configuration are computed and averaged across the 10 generated orchard blocks. The results indicate that the optimal trajectory height for the front-facing camera is 6.5 m, achieving a visual coverage of 32.3% of all the fruits in the orchard block. In contrast, carrying two side-facing cameras provides significantly greater coverage, reaching 45.4% at the optimal height of 5 m. It should be noted that the chosen straight-line trajectories inevitably miss fruits on the opposite sides of the trees, which are not covered by the UAV's flight path, resulting in lower overall percentage coverage compared to the previous example. Additionally, the two side-facing cameras consistently result in higher fruit visibility across nearly all through-the-canopy flight heights, making this configuration a more effective solution for maximizing coverage. Our proposed framework enables users to fine-tune flight paths and camera configurations in simulation, reducing the cost of experimental fine-tuning.

Additionally, the tools introduced in this section can be applied to generate ground truth annotations for synthetic visual data, facilitating the development and testing of vision algorithms, such as those outlined in Section IV. By leveraging the known fruit IDs, the 3D positions of the fruits can be accurately projected into the image frame, enabling the creation of annotated datasets. When paired with synthetic images generated by UAV simulators with integrated autonomy stacks, this approach supports comprehensive evaluation of the entire autonomous system, from trajectory planning to data processing. For instance, these capabilities are supported by our prior work on simulating autonomous flight systems in agricultural environments [25].

III. Drone Autonomy for Data Collection

In this section, we introduce the proposed autonomy pipeline and the vehicle developed for through-the-canopy data collection. The vehicle's intelligence consists of three key components: a low-level controller that converts thrust and angular velocity commands into motor speeds; the RAPPIDS planner [26], which uses depth images for obstacle avoidance during flight; and a visual-inertial-odometry (VIO)-based estimator using OpenVINS [27], which integrates IMU and stereo camera inputs for accurate state estimation. The system interfaces through the Robot Operating System (ROS), and a block diagram illustrating these communications is shown in Figure 2.

The experimental vehicle comprises of the following components: a RealSense D455 camera for depth and stereo image collection, a Pixracer R15 flight controller, and a Qualcomm RB5 computing board. Additionally, an IDS camera for collecting RGB images is mounted on the side of the vehicle. These hardware components, along with the vehicle design, are depicted in Figure 6. Since both state estimation and local planning rely solely on the outputs from the depth camera unit, this design eliminates the need for additional onboard sensors, ensuring cost and power efficiency, and making the vehicle applicable to large-scale monitoring missions. Furthermore, the reliance on vision-based state estimation allows the vehicle to operate in diverse environments, including indoor plant monitoring missions where GPS signals are unavailable [28].

Visual-inertial state estimation is performed using the OpenVINS framework, which we employ for its accuracy and compatibility, enabling seamless integration with our hardware components. It is based on the Multi-State Constraint Kalman Filter (MSCKF) [29], which efficiently fuses

visual and inertial data to estimate the vehicle's state. The MSCKF method takes inputs from IMU measurements and visual features extracted from stereo images, with the IMU providing measurements at 400 Hz and the stereo images processed at 15 Hz. An example of the left stereo camera image with feature tracking is shown in Figure 7a. As a result, the vehicle's 6-degrees-of-freedom pose, along with its velocities and angular velocities, is estimated in real time.

In addition to accurate state estimation, obstacle avoidance is crucial for ensuring the safety of the robot during through-the-canopy flight. To achieve this, we implement the RAPPIDS planner [26], which uses depth images collected at 15 Hz from the RealSense camera to plan collision-free trajecotries. This feature allows the vehicle to avoid unexpected obstacles, such as tree branches, while in flight. The RAPPIDS planner receives a goal point and conducts local trajectory planning by collision-checking samples of feasible trajectories. Using the simulation framework outlined in Section II, the planned global trajectory is first provided to the planner, and the local planner continuously modifies this path based on depth data during flight. The planner samples minimum jerk trajectories, $s(t)$, with duration T using the method outlined in [30], where each trajectory is represented by a fifth-order polynomial:

$$s(t) = \alpha\frac{t^5}{120} + \beta\frac{t^4}{24} + \gamma\frac{t^3}{6} + \ddot{s}(0)\frac{t^2}{2} + \dot{s}(0)t + s(0)$$

where $s(0)$, $\dot{s}(0)$, and $\ddot{s}(0)$ are the initial position, velocity, and acceleration of the vehicle. The coefficients α , β, $\gamma \in \mathbb{R}^3$ depend on $s(T)$, $\dot{s}(T)$, $\ddot{s}(T)$, and T, ensuring smooth motion. The planner then checks each sampled trajectory for input feasibility, speed limits, and collisions. The detailed process for feasibility and velocity checking is discussed in [30]. For collision checking, the planner partitions the free space into rectangular pyramids based on the depth image, and a trajectory remains collision-free if it stays within the union of these pyramids. This method efficiently reduces the computational cost while maintaining reliable obstacle avoidance. More details about this pyramid generation and collision checking process can be found in [26]. The planning process is repeated with each new depth image received to account for the latest obstacle information, and the vehicle follows the previously planned trajectory if no new feasible, collision-free path is found.

Finally, we demonstrate the feasibility of our proposed autonomy stack and vehicle design through an experimental flight in an orchard. The flight is shown in the supplementary video, and Figure 1 displays an action sequence with selected states of the vehicle. In addition, Figure 7 includes samples of the onboard images and the visualization of local planning results. Specifically, Figure 7a shows a sample of the onboard depth image, a monochrome image from the stereo camera for visual-inertial odometry, and a collected RGB image with fruits detected in post processing. Figure 7b presents the reference trajectory in green, the planned trajectory at selected time steps in blue, and the resulting flight path, based on the position output from the estimator, in black.

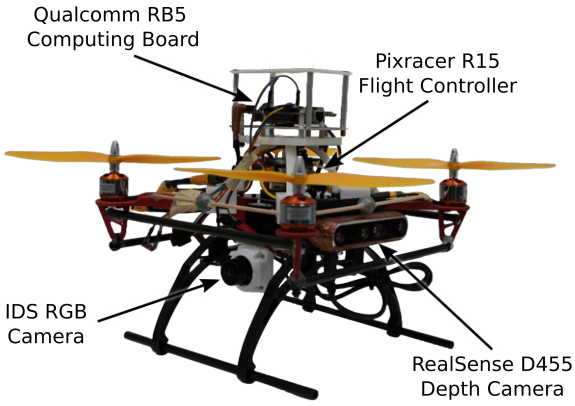

Fig. 6. Experimental vehicle with the proposed autonomy stack. A RealSense D455 depth camera captures both depth and stereo images, while an IDS camera is responsible for RGB image collection for fruit counting. The autonomy algorithms are executed on a Qualcomm RB5 board, which sends desired thrust and angular velocity commands to a Pixracer R15 flight controller.

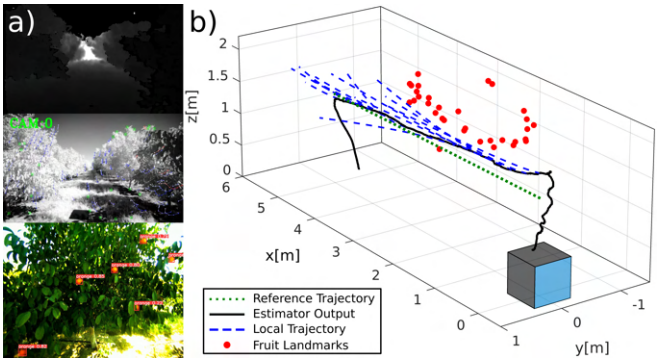

Fig. 7. Example data collection flight results. (a) Samples of the onboard depth image, left stereo image, and RGB image. (b) Visualization of the experimental flight. The supplementary video showcases the same experiment, where the vehicle follows the global reference trajectory (green), and locally planned trajectories (blue) are shown at selected time steps. The visual-inertial-based estimation output is displayed in black, with reconstructed fruit landmarks highlighted in red.

Additionally, using the collected onboard data, we can reconstruct the locations of detected fruits, as outlined in Section IV. This example highlights the potential for our proposed data collection method for fruit counting missions.

IV. Autonomous Fruit Counting

While our proposed methods outlined in previous sections ensure effective data collection flights, accurate fruit identification and counting from the onboard images remain key to completing the autonomous fruit counting mission. To this end, we adopt the approach proposed in [8], which effectively mitigates double counting, where the same fruit may appear in multiple frames. This method uses fruits as features to reconstruct 3D landmarks, providing an accurate fruit count while maintaining computational efficiency in the reconstruction process. We further enhance this approach by integrating the latest computer vision advancements for detection and tracking. Specifically, we use YOLOv8 [6] for fruit detection and ByteTrack [12] for tracking across image

sequences. The proposed workflow involves: i) detecting fruit in each frame using fine-tuned YOLOv8, ii) tracking detected fruits across the image sequence with ByteTrack, iii) recovering 3D fruit locations via COLMAP [10], [11], and iv) clustering the reconstructed fruit landmarks to obtain the final count.

For detection, we first fine-tune the YOLOv8 model using a dataset of 100 images of the target fruit type, collected under good lighting condition and manually labeled with bounding boxes. This process ensures that the model adapts to the unique visual characteristics of the fruit, providing robust performance even in suboptimal conditions, such as varying lighting and partial occlusion from leaves or branches. YOLOv8 is well-suited for orchard fruit detection due to its ability to efficiently detect multiple overlapping fruits in cluttered, natural environments while maintaining high accuracy even for smaller, partially obscured objects.

While [8] uses the Hungarian Algorithm for fruit tracking, we adopt ByteTrack for its superior handling of low-confidence detections, which are common under poor imaging conditions. ByteTrack effectively maintains feature correspondences between frames, providing robust raw inputs for the structure-from-motion step. Figure 8a shows a sample of the tracking output, with each tracked fruit assigned a unique ID to distinguish it through the sequence.

COLMAP remains the state-of-the-art structure-from-motion toolbox, and we use it to reconstruct the 3D landmarks for the fruits. Using the feature correspondences from the tracking step, COLMAP recovers camera intrinsics, extrinsics, and 3D landmarks, with fruits serving as features. Since the RGB camera is integrated into the autonomy stack, the reconstruction process may be initialized using the camera extrinsics, improving the reliability and accuracy of fruit landmark positioning. Finally, we apply density-based spatial clustering [31] to group the reconstructed landmarks by individual fruits, yielding the final fruit count.

Figure 8 illustrates an example of autonomous fruit counting using the proposed workflow. A video captured during a low-altitude flight near a small orange tree with 12 fruits is processed, and the method accurately recovers all 12 fruits along with their estimated positions. The fruits are first detected and tracked in the image frames, as shown in Figure 8a. During the structure from motion step, the camera extrinsics and 33 landmarks are reconstructed, as depicted in Figure 8b. Finally, after applying the clustering algorithm, the landmarks are grouped into 12 distinct clusters, as shown in different colors, revealing the total fruit count. This example with orange counting demonstrates the effectiveness of the proposed approach. However, in larger orchards with smaller fruits, increased occlusions may impact detection accuracy, like discussed in Section II. With further refinement, the pipeline is expected to handle these more challenging scenarios effectively.

V. Conclusion and Future Work

In this work, we have developed an autonomous aerial system designed for efficient and safe through-the-canopy

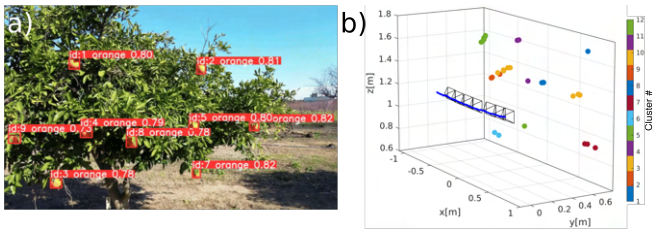

Fig. 8. Example results for the proposed autonomous counting workflow. (a) Tracking output where an ID is assigned to each detected and tracked fruit. (b) Counting results after the structure-from-motion step for a tree with 12 fruits. The 33 reconstructed 3D landmarks are grouped into 12 distinct clusters, denoted by different colors. The reconstructed camera extrinsics are marked in black at select time steps.

fruit counting in large-scale orchards. Our approach leverages a high-fidelity simulation framework to optimize flight trajectories, a cost-effective autonomy stack for canopy-level navigation and data collection, and state-of-the-art computer vision tools for fruit counting. The system's performance was validated through successful fruit counting using aerial images captured at canopy level, along with demonstrations of the autonomous navigation capabilities of the experimental vehicle. These results demonstrate the system's potential to enhance data collection efficiency and fruit yield estimation in complex orchard environments, offering a promising solution for large-scale agricultural applications.

For future work, several improvements can be made to further enhance the system's capabilities. First, global planning could be improved by considering the flight speed of the vehicle, as it significantly impacts both power consumption and orchard coverage. Second, incorporating real-time feedback on the quality of the collected images could further ensure the reliability of data collection. While the integration of the RGB camera into the autonomy stack provides the potential for this enhancement, it has not yet been implemented in the current system. Finally, closer integration of the autonomy stack and fruit counting pipeline would enable real-time, onboard fruit counting, which is achievable given the computational efficiency of our method. Addressing these factors would help further optimize the system for large-scale, continuous agricultural monitoring.

Acknowledgment

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 Graduate Student Summer Fellowship, and the William C. Webster Graduate Fellowship from the Department of Mechanical Engineering at UC Berkeley. The authors would like to thank Dylan Lee and Ting-Hao Wang for their contributions to the development of the autonomous vehicle, and Brian Bailey along with the members of the UC Davis Plant Simulation Laboratory for their valuable advice on plant modeling and fruit detection.

REFERENCES

[1] C. Zhang, J. Valente, L. Kooistra, L. Guo, and W. Wang, "Orchard management with small unmanned aerial vehicles: A survey of sensing and analysis approaches," *Precision Agriculture*, vol. 22, no. 6, pp. 2007–2052, 2021.

[2] G. Farjon, L. Huijun, and Y. Edan, "Deep-learning-based counting methods, datasets, and applications in agriculture: a review," *Precision Agriculture*, pp. 1–29, 2023.

[3] W. Maldonado Jr and J. C. Barbosa, "Automatic green fruit counting in orange trees using digital images," *Computers and Electronics in Agriculture*, vol. 127, pp. 572–581, 2016.

[4] N. Aleixos, J. Blasco, F. Navarron, and E. Moltó, "Multispectral inspection of citrus in real-time using machine vision and digital signal processors," *Computers and electronics in agriculture*, vol. 33, no. 2, pp. 121–137, 2002.

[5] Y. Zhang, W. Zhang, J. Yu, L. He, J. Chen, and Y. He, "Complete and accurate holly fruits counting using yolox object detection," *Computers and Electronics in Agriculture*, vol. 198, p. 107062, 2022.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[7] A. I. B. Parico and T. Ahamed, "Real time pear fruit detection and counting using yolov4 models and deep sort," *Sensors*, vol. 21, no. 14, p. 4803, 2021.

[8] X. Liu, S. W. Chen, C. Liu, S. S. Shivakumar, J. Das, C. J. Taylor, J. Underwood, and V. Kumar, "Monocular camera based fruit counting and mapping with semantic data association," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2296–2303, 2019.

[9] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[10] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[11] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[12] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang, "Bytetrack: Multi-object tracking by associating every detection box," in *European conference on computer vision*. Springer, 2022, pp. 1–21.

[13] A. Mokrane, A. C. Braham, and B. Cherki, "Uav coverage path planning for supporting autonomous fruit counting systems," in *2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID)*, vol. 1. IEEE, 2019, pp. 1–5.

[14] O. E. Apolo-Apolo, J. Martínez-Guanter, G. Egea, P. Raja, and M. Pérez-Ruiz, "Deep learning techniques for estimation of the yield and size of citrus fruits using a uav," *European Journal of Agronomy*, vol. 115, p. 126030, 2020.

[15] A. Mokrane, A. Choukchou-Braham, and B. Cherki, "Coverage path planning of autonomous marsupial systems for supporting fruit counting process," in *2020 International Conference on Electrical Engineering (ICEE)*. IEEE, 2020, pp. 1–6.

[16] Z. Zheng, J. Xiong, X. Wang, Z. Li, Q. Huang, H. Chen, and Y. Han, "An efficient online citrus counting system for large-scale unstructured orchards based on the unmanned aerial vehicle," *Journal of Field Robotics*, vol. 40, no. 3, pp. 552–573, 2023.

[17] N. Stefas, H. Bayram, and V. Isler, "Vision-based monitoring of orchards with uavs," *Computers and Electronics in Agriculture*, vol. 163, p. 104814, 2019.

[18] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[19] P. Roy and V. Isler, "Active view planning for counting apples in orchards," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6027–6032.

[20] S. Marangoz, T. Zaenker, R. Menon, and M. Bennewitz, "Fruit mapping with shape completion for autonomous crop monitoring," in *2022 IEEE 18th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 471–476.

[21] Y.-T. Wang, B. N. Bailey, K. Fu, and K. Shackel, "Topological and spatial analysis of within-tree fruiting characteristics for walnut trees," *Scientia Horticulturae*, vol. 318, p. 112127, 2023.

[22] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection," in *ACM SIGGRAPH 2005 Courses*, 2005, pp. 7–es.

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

[24] 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.

[25] J. Zha, T. Yang, and M. W. Mueller, "Agri-fly: Simulator for uncrewed aerial vehicle flight in agricultural environments," 2024, submitted for publication.

[26] 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.

[27] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4666–4672.

[28] M. F. Aslan, A. Durdu, K. Sabanci, E. Ropelewska, and S. S. Gültekin, "A comprehensive survey of the recent studies with uav for precision agriculture in open fields and greenhouses," *Applied Sciences*, vol. 12, no. 3, p. 1047, 2022.

[29] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE international conference on robotics and automation*. IEEE, 2007, pp. 3565–3572.

[30] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE transactions on robotics*, vol. 31, no. 6, pp. 1294–1310, 2015.

[31] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.