

PAPER

Low-Cost Car-Like Autonomous Mobile Robot Under-Canopy Row Following, and Detection of Patterns Associated with Diseases in Plantain Crops

Danna C. Montenegro¹,
 Iván D. Duarte²,
 Andrés L. Jutinico¹,
 Henry B. Guerrero²  

¹Universidad Distrital
 Francisco José de Caldas,
 Bogotá, Colombia

²BITPOINTER S.A.S.,
 Villavicencio, Colombia

hbguerreiro@ieee.org

ABSTRACT

Labor constraints and the need for frequent visual inspection of lower plantain (*Musa paradisiaca*) leaves in Colombia's Orinoquia region motivate the development of low-cost robotic tools capable of autonomous under-canopy navigation and systematic image capture. This study reports the current stage of an ongoing research effort on a low-cost, small-scale, car-like agricultural robot for autonomous under-canopy row-following and systematic image capture in plantain fields. The platform uses a modular ROS 2 workflow, a low-cost 2D LiDAR, an inertial measurement unit, and an ESP32-based actuation layer. LiDAR data were processed through an adaptive measurement cone guided by the vehicle heading error to estimate the lateral distance to the crop row during forward row-following motion, and a fuzzy controller generated steering references. Initial field tests across five trials provided preliminary evidence of consistent under-canopy navigation under the tested conditions. In the first row-following segment, the median MAE and RMSE of the lateral error were 0.0258 m and 0.0342 m, respectively; after the row-to-row maneuver, they were 0.0356 m and 0.0602 m, respectively. Additionally, an upward-facing RGB camera captured under-canopy images that were processed offline to create probability-based visual summaries of leaf-condition patterns. The results show that low-cost robotic navigation can help structure under-canopy data collection for scalable crop health monitoring in plantain plantations.

KEYWORDS

autonomous mobile robot, fuzzy control, leaf images, offline analysis, pattern recognition, plantain crops

1 INTRODUCTION

Agricultural production faces structural challenges, including labor and skill shortages that affect regular fieldwork [1–3]. Automation, robotics, low-cost

Montenegro, D. C., Duarte, I. D., Jutinico, A. L., Guerrero, H. B. (2026). Low-Cost Car-Like Autonomous Mobile Robot Under-Canopy Row Following, and Detection of Patterns Associated with Diseases in Plantain Crops. *International Journal of Online and Biomedical Engineering (iJOE)*, 22(10), pp. 61–89. <https://doi.org/10.3991/ijoe.v22i10.61223>

Article submitted 2026-02-23. Revision uploaded 2026-07-06. Final acceptance 2026-07-06.

© 2026 by the authors of this article. Published under CC-BY.

digital tools, IoT crop-monitoring systems, and innovation strategies have been explored to reduce labor constraints, improve task consistency, and support decision-making [2, 3, 5–7]. However, adoption in small and medium-sized farms remains limited by cost, maintenance, and the lack of adaptable field workflows [4], especially in plantain (*Musa paradisiaca*) production, where operations are labor-intensive and technology use is constrained by economic and infrastructure issues [8, 9]. Regular inspection of plantain leaf conditions is essential because diseases in the Sigatoka disease complex can reduce yield and profitability [10–15]. Yet manual monitoring is time-consuming and skill-dependent, motivating scalable and cost-effective field methods. Mobile robots can navigate crop rows while collecting sensory and visual data [16–19], but under-canopy operation remains challenging owing to unreliable GNSS, variable lighting, obstructive foliage, and uneven ground. Thus, robust and practical perception systems are required: 2D LiDARs support short-range geometric perception and wall-following behaviors [20–22], whereas low-cost IMUs help maintain orientation during turns and disturbances [23, 24].

Driven by these needs, this study presents a low-cost, small-scale, car-like agricultural robot that combines autonomous under-canopy row following with structured RGB image capture in plantain fields. The platform integrates ROS 2, a 2D LiDAR, a low-cost IMU, ESP32-based actuation, and fuzzy steering control. LiDAR estimates the crop-wall distance within a heading-guided adaptive angular sector, while RGB images are processed offline with BananaSqueezeNet to summarize leaf-condition probabilities [25]. Since hyperspectral imaging and computationally demanding deep-learning pipelines can be costly and difficult to deploy in small-farm contexts [26, 27], the proposed workflow emphasizes practical RGB-based screening.

This work extends our group's previous research on H_∞ LiDAR-based row following, ROS 1 LiDAR–IMU fusion with LQR/LQG control, kinematic control, and ROS 2 fuzzy control for plantain row following [28–31], transferring this line of work to a car-like platform that combines row-following navigation, systematic image acquisition, and offline leaf-condition pattern summarization.

The main contributions of this study are: (i) a low-cost car-like robotic platform with a ROS 2-based workflow for row following and under-canopy image capture in plantain crops; (ii) a LiDAR–IMU method that adjusts the measurement cone using heading error to estimate lateral distance; (iii) a maneuver-aware field protocol with separate tracking metrics before and after row-to-row transitions; and (iv) an offline RGB image-processing workflow that generates probability-based summaries of leaf conditions along the robot path.

The paper is organized as follows: Section 2 describes the robot platform, operating environment, ROS 2–ESP32 architecture, LiDAR–IMU processing, fuzzy control, image capture, and embedded firmware. Section 3 presents the field results, including maneuver-aware tracking metrics and probability maps from the collected images. Section 4 concludes the paper and outlines future work.

2 GENERAL ASPECTS OF THE ROBOT AND ITS OPERATING ENVIRONMENT

A low-cost, car-like autonomous mobile robot was developed and field-tested in plantain plantations to navigate between crop rows and collect under-canopy

sensory and visual data. As shown in Figure 1, the platform was conceived as a compact and robust field testbed for tropical crop conditions, including uneven ground, high humidity, and intense or highly variable illumination, which may degrade low-cost LiDAR sensing. The collected data are intended to support future research on early disease detection in plantain crops, including Black Sigatoka.

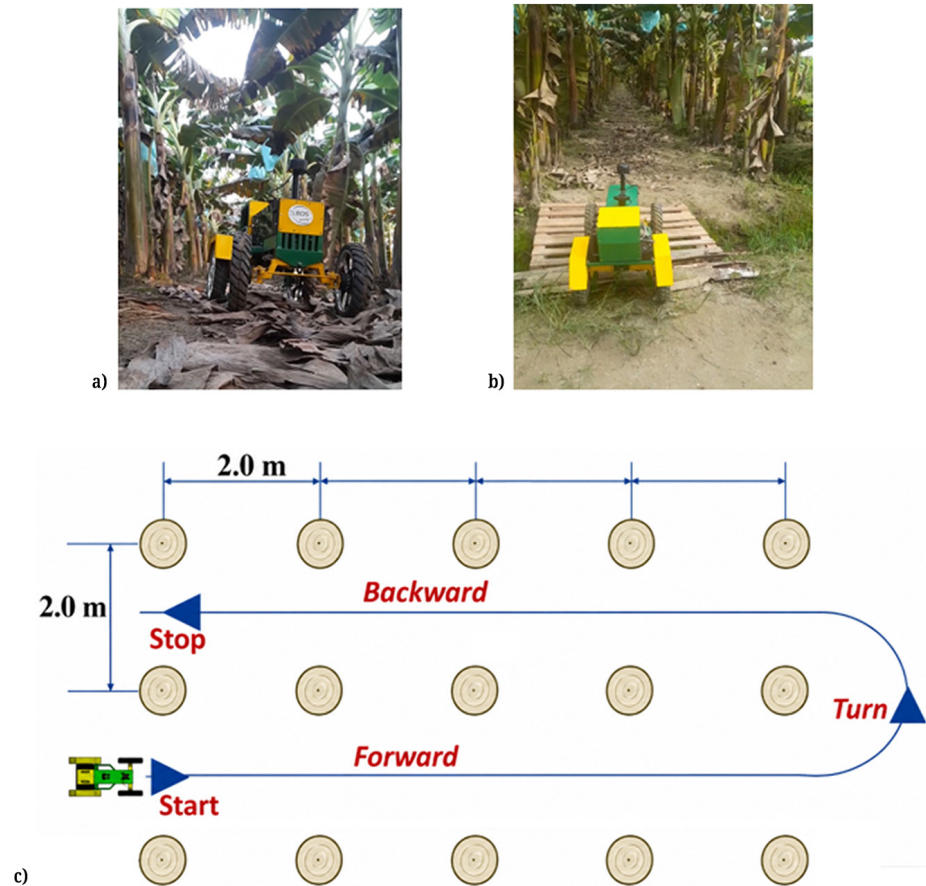


Fig. 1. Experimental context: a) and b) Robot navigating under the canopy; c) planting layout (2 m × 2 m) and mission path

The test plantation was located near Villavicencio, Meta, Colombia, as shown in Figure 2, and followed a regular 2 m × 2 m planting layout. The operating area consisted of narrow inter-row paths with soft soil, ground biomass, dry leaves, exposed roots, stems, and dense foliage, which together contributed to uneven terrain and variable traction conditions. Under-canopy illumination also varied considerably, ranging from bright sunlit patches to deep shadows, further challenging LiDAR-based perception and image acquisition under real field conditions.

During each run, the robot followed a crop row using onboard sensing and reactive control and then executed a row-end turn to align with the next inter-row path. These field trials evaluated the main navigation functions of the platform, including Ackermann steering, PWM-based propulsion, and LiDAR-IMU-based distance estimation. An upward-facing camera mounted near the center of the robot captured images of lower leaf surfaces for subsequent offline leaf-condition analysis.

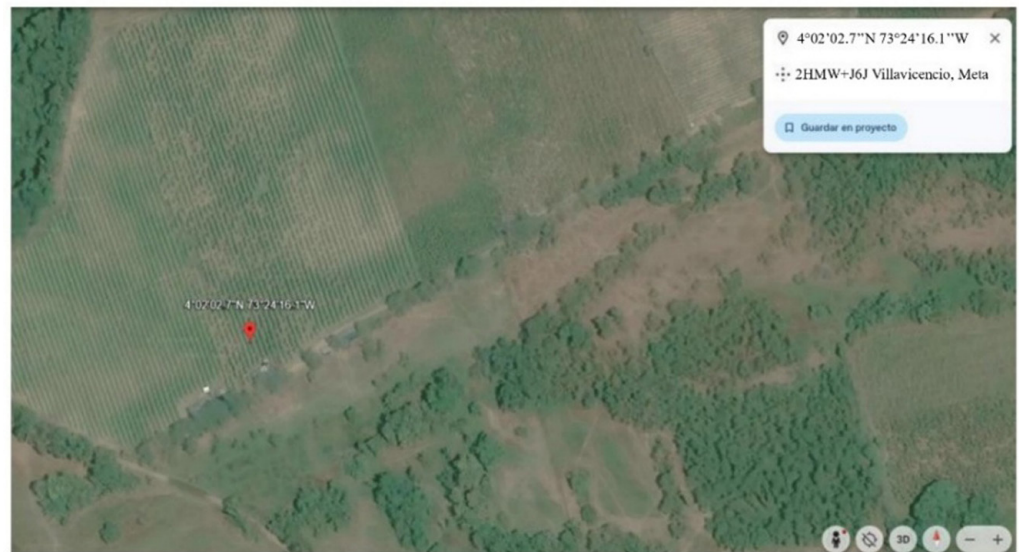


Fig. 2. Test area near Villavicencio, Meta, Colombia (Google Earth, © 2026)

2.1 Mechanical structure and onboard devices

The robot has a modular car-like frame with four-wheel Ackermann steering, providing maneuverability, stability, ground clearance, and ease of maintenance on uneven terrain [30, 32–34]. As shown in Figure 3, the frame was built from welded steel profiles and covered with black acrylic panels to protect the electronics and power sections while allowing inspection and future upgrades [35].

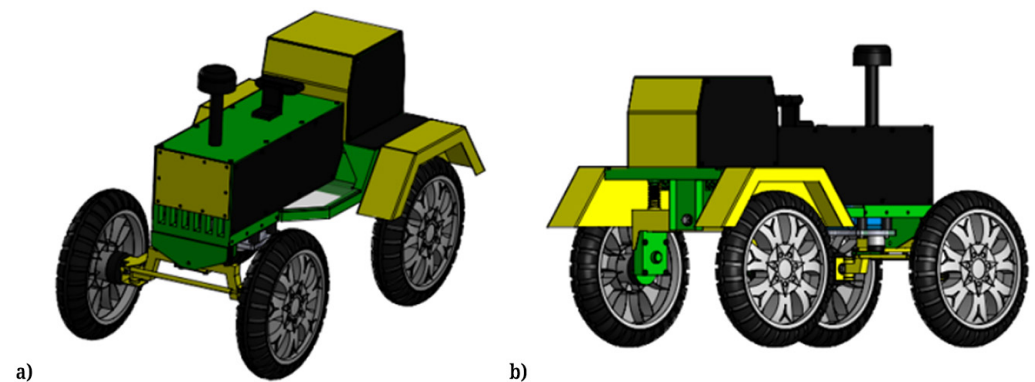


Fig. 3. CAD model of the robot: a) front isometric view and b) side-oriented view

Figure 4 shows the front and rear views of the car-like robot, highlighting its main sensors and drivetrain components. In the front view, the numbered callouts identify the top-mounted RPLIDAR A2 for environmental scanning (1) [20, 21], the upward-facing USB camera for under-canopy image capture (2), the Ackermann linkage connected to the front wheels (3), and the front wheels (4, 5). In the rear view, they identify the steering gear motor with a Hall-effect encoder (6), the basic spring suspension elements (7, 8), the two 24 V DC planetary drive motors (9, 10), and the rear wheels (11, 12) [35].

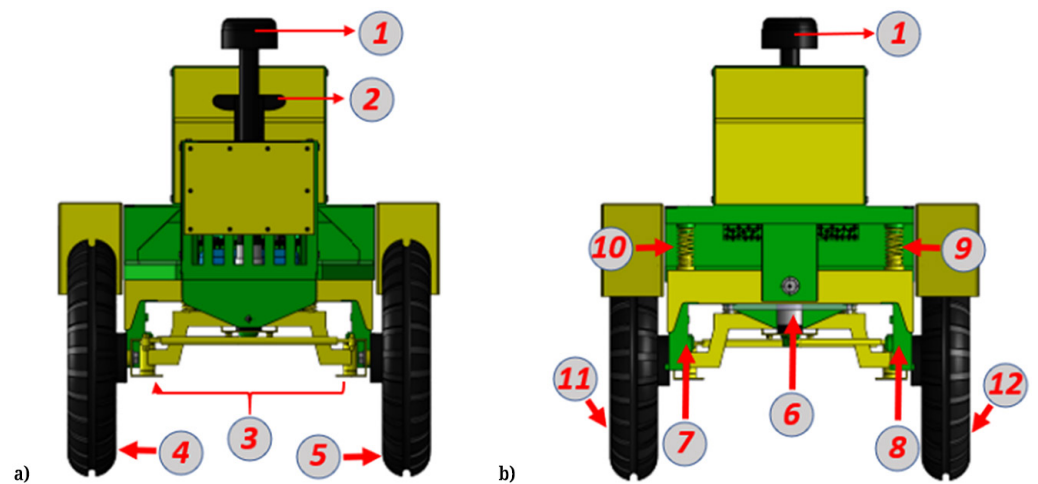


Fig. 4. Front a) and rear b) views of the car-like robot

Figure 5 shows the bottom and top views of the robot, highlighting the steering, power, control, and computing layout. In the bottom view, the numbered callouts identify the front Ackermann mechanism (1), centrally located 24 V battery (2), H-bridge drivers (3), ESP32–BNO055 electronics bay (4), and symmetrically mounted rear DC gear motors (5, 6). In the top view, they identify the steering motor (7) and the Intel® mini-PC running Ubuntu 24.04 (8), which manages ROS 2 communication, LiDAR processing, and image capture. This compact layout reduces cable length and simplifies maintenance.

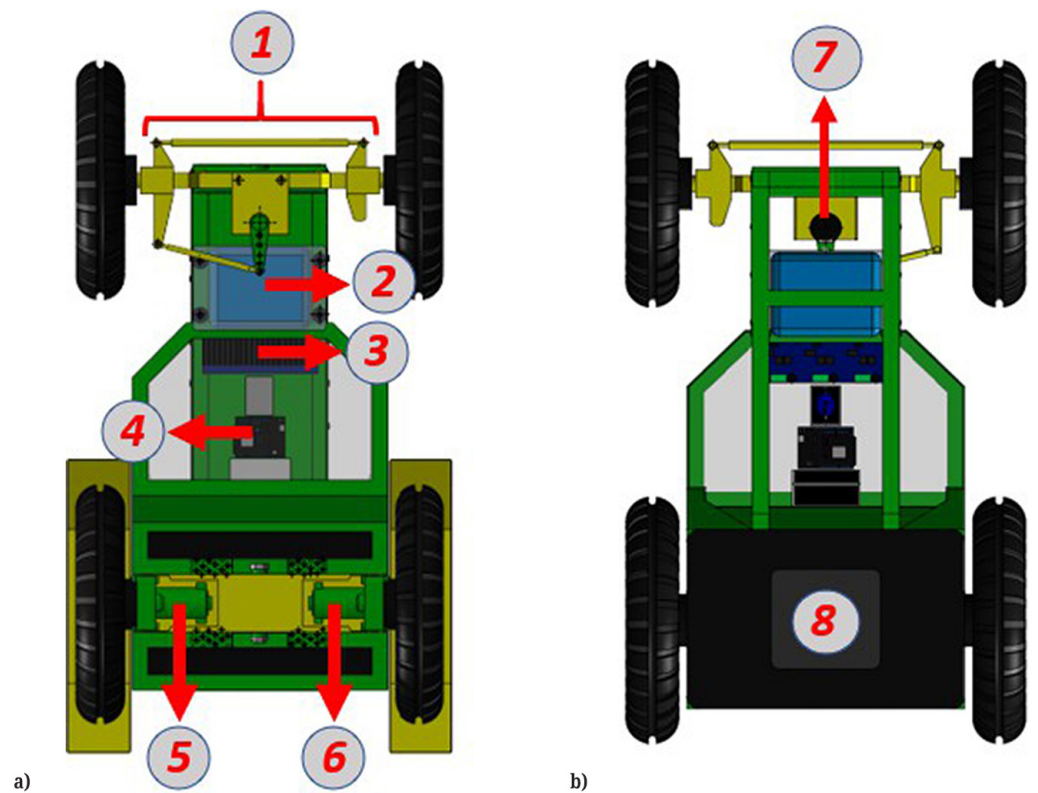


Fig. 5. Internal layout of the car-like robot: a) Bottom view b) Top view

Figure 6 shows the orthogonal views and main dimensions of the car-like robot. The platform was 89.5 cm long, with a 67.7 cm wheelbase and front/rear track

widths of 47.5 cm and 49.8 cm, respectively. These dimensions, together with a 23–25 cm ground clearance, allowed the robot to move between plantain rows and pass over plant debris, roots, and uneven soil. The centrally located 24 V battery, 31 cm-diameter rubber wheels, and modular frame supported balance, traction, shock absorption, and ease of maintenance under canopy field conditions.

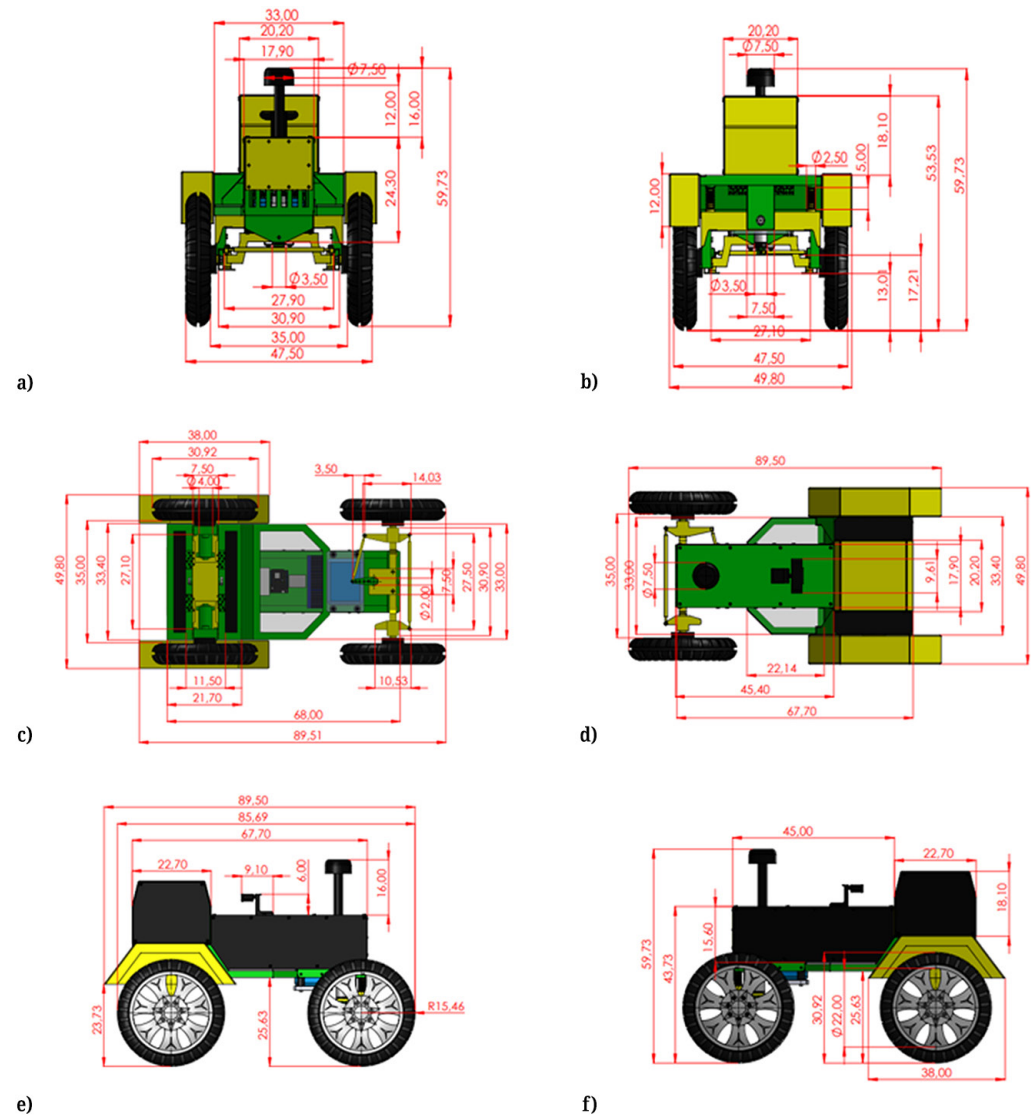


Fig. 6. Orthogonal projections of the car-like robot with principal dimensions (cm): a) front, b) rear, c) bottom, d) top, e) left, and f) right

Movement relies on rear-wheel traction and front-wheel Ackermann steering. Two 24 V DC planetary gear motors mounted on the rear axle provided propulsion, delivering approximately 45 rpm and 2 N·m. Their full-metal gearboxes and 8 mm D-shaped shafts were connected to the wheels through rigid hubs to reduce backlash and maintain traction on soft soil. Traction currently operates in open loop, but the chassis allows future integration of wheel encoders for closed-loop speed control. The steering system is driven by a Chihai Motor CHP-36GP-555-ABHLL 24 V DC gear motor with a dual-channel Hall-effect incremental encoder. A rigid arm attached to the 8 mm D-shaped shaft actuates the linkage and aligns the front-wheel

angles according to the Ackermann geometry. Further mechanical details are provided in [35].

2.2 Sensing–computation–actuation information flow

Figure 7 summarizes the sensing–computation–actuation flow. A mini-PC executes the high-level ROS 2 tasks, whereas an ESP32 manages low-level control. Perception relies on a webcam, LiDAR, and IMU: the LiDAR sends polar scans (l, α) for crop-row distance estimation, the IMU provides orientation data (ψ, θ, ϕ), with yaw (ψ) used as a geometric reference, and the webcam captures RGB images of the lower canopy for offline leaf analysis.

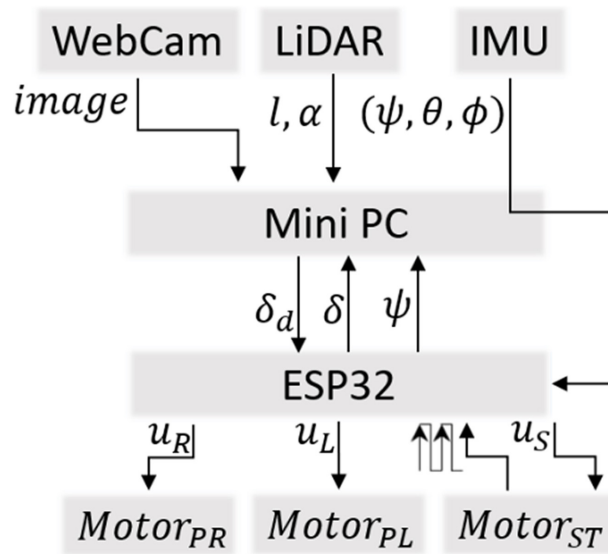


Fig. 7. Sensing–computation–actuation information flow

On the mini-PC, the ROS 2 nodes process LiDAR and IMU data to generate the steering reference (δ_d), which is sent to the ESP32 through serial communication together with supervisory commands. The ESP32 returns the measured steering angle (δ) and yaw angle (ψ), drives the rear traction motors ($Motor_{PR}$, $Motor_{PL}$) using open-loop PWM, and controls the steering motor ($Motor_{ST}$) in closed loop using steering encoder feedback. The commands u_R , u_L , and u_S represent the duty-cycle/voltage inputs for the right, left, and steering motors, respectively, while the encoder pulse train in Figure 7 indicates the feedback path used to measure δ . This architecture assigns perception and decision-making to ROS 2 while preserving the timing required for low-level control on the microcontroller.

2.3 ROS 2 communication architecture and ESP32 interface

Robot Operating System 2 (ROS 2) was used as middleware to connect sensing, decision-making, and control in the proposed platform. ROS 2 follows a publish/subscribe model in which separate processes, called nodes, exchange typed messages through named channels, called topics. This architecture supports modular perception-control pipelines, distributed execution on different hardware,

and reproducible experiments through standardized logging and replay tools, such as bag files [36, 37]. In this system, each node performs a specific function, including LiDAR preprocessing, wall-following control, or serial communication, and publishes or subscribes to topics carrying data such as laser scans, images, and scalar signals. Asynchronous communication separates producers from consumers, allowing logging, visualization, and experimental nodes to be added without modifying the primary control stack.

Figure 8 illustrates the ROS 2–ESP32 interface. The nodes inside the dashed area run on the mini-PC, whereas the ESP32 handles low-level loops and interfaces with the sensors and actuators. The diagram summarizes the exchange of LiDAR-derived data, heading/yaw signals, steering references, and stem indices used for image annotation.

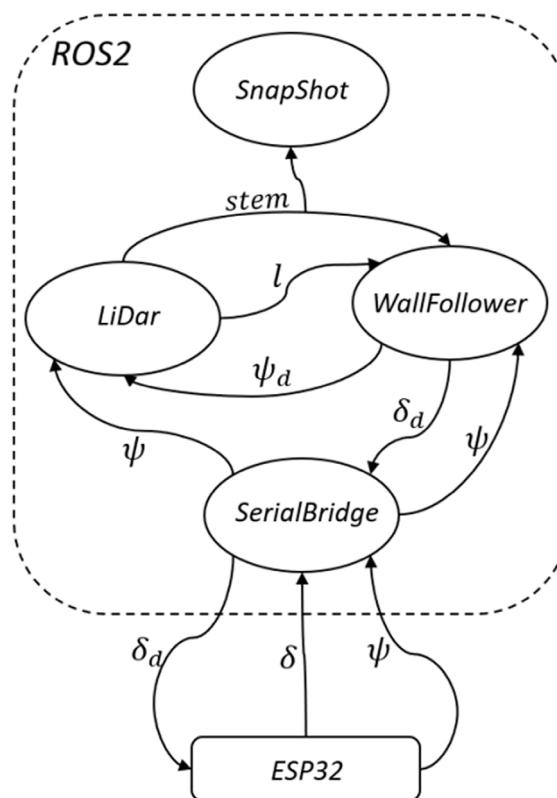


Fig. 8. ROS 2 node graph and ROS 2-ESP32 interface

The SerialBridge node (see Figure 8) provides the two-way ROS 2–ESP32 interface between the mini-PC and the ESP32 firmware. It manages the serial protocol and exposes the exchanged data as scalar ROS 2 topics, including the steering reference δ_d and feedback signals δ and ψ .

On the ROS 2 side, SerialBridge subscribes to the steering reference δ_d generated by the WallFollower node. Each value is formatted as a line-oriented ASCII message with fixed decimals and a newline character, then sent to the ESP32 to simplify parsing and allow verification with standard serial tools. The serial port and baud rate were configured through ROS 2 parameters. After opening the port, the node discards residual boot messages, keeps write operations separated, attempts reconnection after errors or disconnections, and reports events through ROS 2 logs. In the opposite direction, the ESP32 sends tagged telemetry lines with the measured steering angle δ , obtained from the steering encoder, and yaw angle ψ , obtained

from the IMU. SerialBridge reconstructs complete lines, parses the tagged fields, and publishes δ and ψ as ROS 2 topics for the other nodes, as shown in Figure 8. For traceability, it can also log δ_d , δ , and ψ to CSV files with ROS timestamps, keeping framing, error handling, and file management within the same node.

The current implementation was validated as a low-speed ROS 2–ESP32 workflow. However, detailed temporal profiling of the ROS 2 graph, including node execution frequencies, CPU/memory load, serial communication latency, and end-to-end jitter, was not performed in this study. Therefore, these timing metrics are not reported as performance results; instead, they are identified as future tasks for reproducibility and real-time characterization in high-speed navigation and onboard inference experiments.

Regarding the ROS 2 nodes shown in Figure 8, the LiDAR node estimates the lateral crop-row distance, denoted as l_y , following the approach in [28–31], and also computes the stem index. This node subscribes to the yaw angle ψ and desired heading ψ_d . Because the 2D LiDAR provides range and bearing measurements around the robot [20, 29, 31, 35], crop-row detection was restricted to the robot's left side, as shown in Figure 9. In the adopted LiDAR polar frame, $\alpha = 0$ points forward and angles increase clockwise; therefore, the left crop row is located around $\alpha = -\pi/2$. The angular limits α_{low} and α_{high} define the measurement window, and only beams whose wrapped bearing lies inside this window are used to estimate l_y , as expressed in Equation (1).

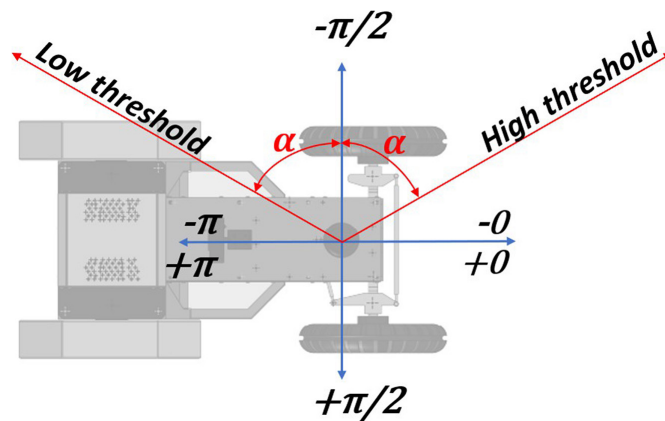


Fig. 9. LiDAR angle convention and left side measurement cone

$$\alpha_{low} \leq \tilde{\alpha} \leq \alpha_{high} \quad (1)$$

Figure 10 illustrates the LiDAR measurements used to estimate l_y , defined as the geometric lateral distance from the LiDAR reference point on the robot to the detected crop row, measured along the direction normal to the crop-row wall. Only beams within the measurement cone that reflected from plantain stems were considered. The cone center α_c represents the row-normal direction, and each valid beam contributes its lateral projection $l_i \cos \alpha_i$. The LiDAR processing stage estimates l_y as the average of the valid beam projections, as expressed in Equation (2), where N is the number of valid beams. Based on the plantation geometry, projections were accepted only when $0 < l_i \cos \alpha_i \leq l_{max}$, with $l_{max} = 2.0$ m corresponding to the row spacing.

$$l_y = \frac{1}{N} \sum_{i=1}^N l_i \cos \alpha_i, \quad (2)$$

The measurement strategy illustrated in Figure 10 is accurate when the vehicle is aligned with the crop row. However, if the vehicle orientation changes during

row following, a LiDAR measurement cone fixed to the robot body may produce biased lateral distance estimates. Figure 11 illustrates this limitation. In Figure 11a, the vehicle is aligned with the crop row, and the fixed-cone estimate l_f coincides with the actual lateral offset l_y . In contrast, Figures 11b and 11c show heading deviations in opposite directions; in both cases, the body-fixed cone is no longer centered on the row-normal direction, so l_f is computed from beams that do not represent the perpendicular distance to the crop row and therefore does not coincide with l_y .

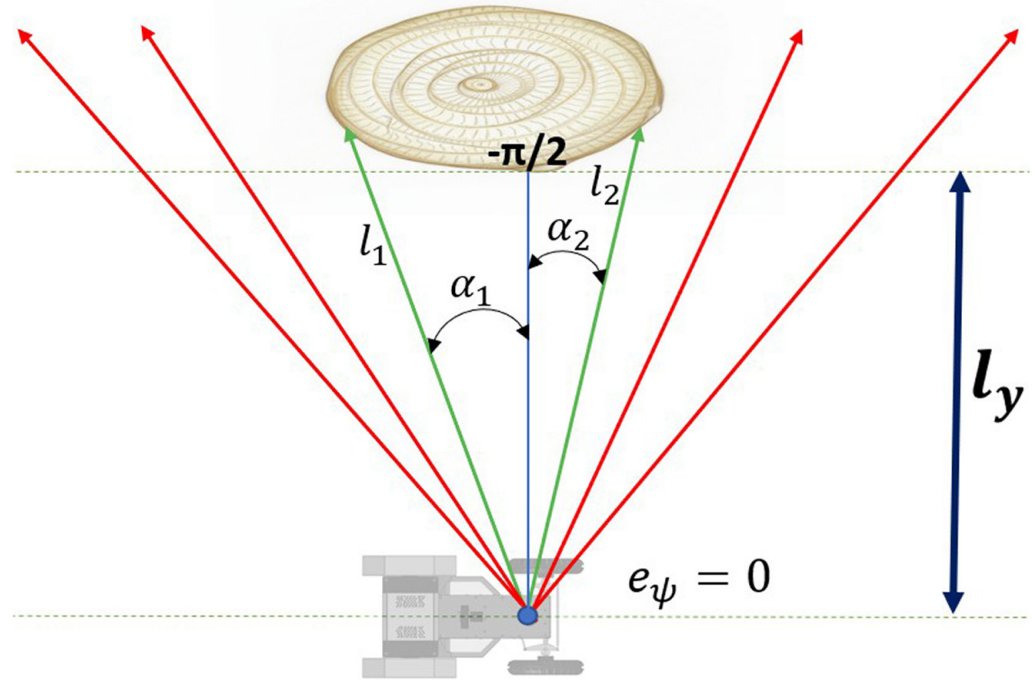


Fig. 10. A valid beam projection used to estimate the lateral distance l_y

Situations such as those shown in Figures 11b and 11c are common during row following; therefore, the heading error was included in the LiDAR measurement strategy. The desired heading ψ_d was set at the beginning of each run, when the robot was manually positioned and aligned with the crop row according to the initial experimental configuration shown in Figure 1c. After each row-to-row maneuver, ψ_d was updated to represent the next row segment, computed as the initial row heading plus 180°. The heading error was calculated as $e_\psi = \psi_d - \psi$, where ψ is the measured yaw angle, and normalized to the range $(-\pi, \pi]$ to avoid discontinuities when crossing the $\pm\pi$ boundary. This error was not used directly as a control action but as a geometric correction to align the LiDAR measurement cone with the crop-row normal direction, following the LiDAR-based row-following principle in [28–31]. For each LiDAR-processing sample k , the adaptive cone center is defined by Equation (3), where $-\pi/2$ is the nominal left-side row-normal direction when the robot is aligned with the crop row, and $e_\psi(k)$ is the heading error at that sample. Thus, the cone center shifts with the heading error so that the selected LiDAR beams remain oriented toward the crop-row normal instead of being fixed to the robot body. A LiDAR beam (i), with bearing α_i , is selected for lateral distance estimation when its angular distance from the adaptive cone center is within the cone half-aperture, as defined in Equation (4).

$$\alpha_c(k) = -\frac{\pi}{2} + e_\psi(k), \quad (3)$$

$$-\alpha_\omega \leq \text{wrap}_\pi(\alpha_i - \alpha_c(k)) \leq \alpha_\omega \quad (4)$$

In Equation (4), α_ω is the half-aperture of the measurement cone, and $\text{wrap}_\pi(\cdot)$ maps angular differences to the range $(-\pi, \pi]$. Therefore, only beams within an angular window centered at $\alpha_c(k)$ are selected, and the lateral offset l_y is computed from their valid projections using Equation (2). Figures 12 and 13 illustrate the effect of this angular correction for positive and negative heading errors, respectively. In each case, subfigure (a) shows the body-fixed cone, which remains attached to the robot frame and may become misaligned with the crop row, whereas subfigure (b) shows the adaptive cone, whose center is shifted according to e_ψ to remain aligned with the row-normal direction. This correction preserves the geometric meaning of l_y during moderate heading deviations and row reacquisition after maneuvers.

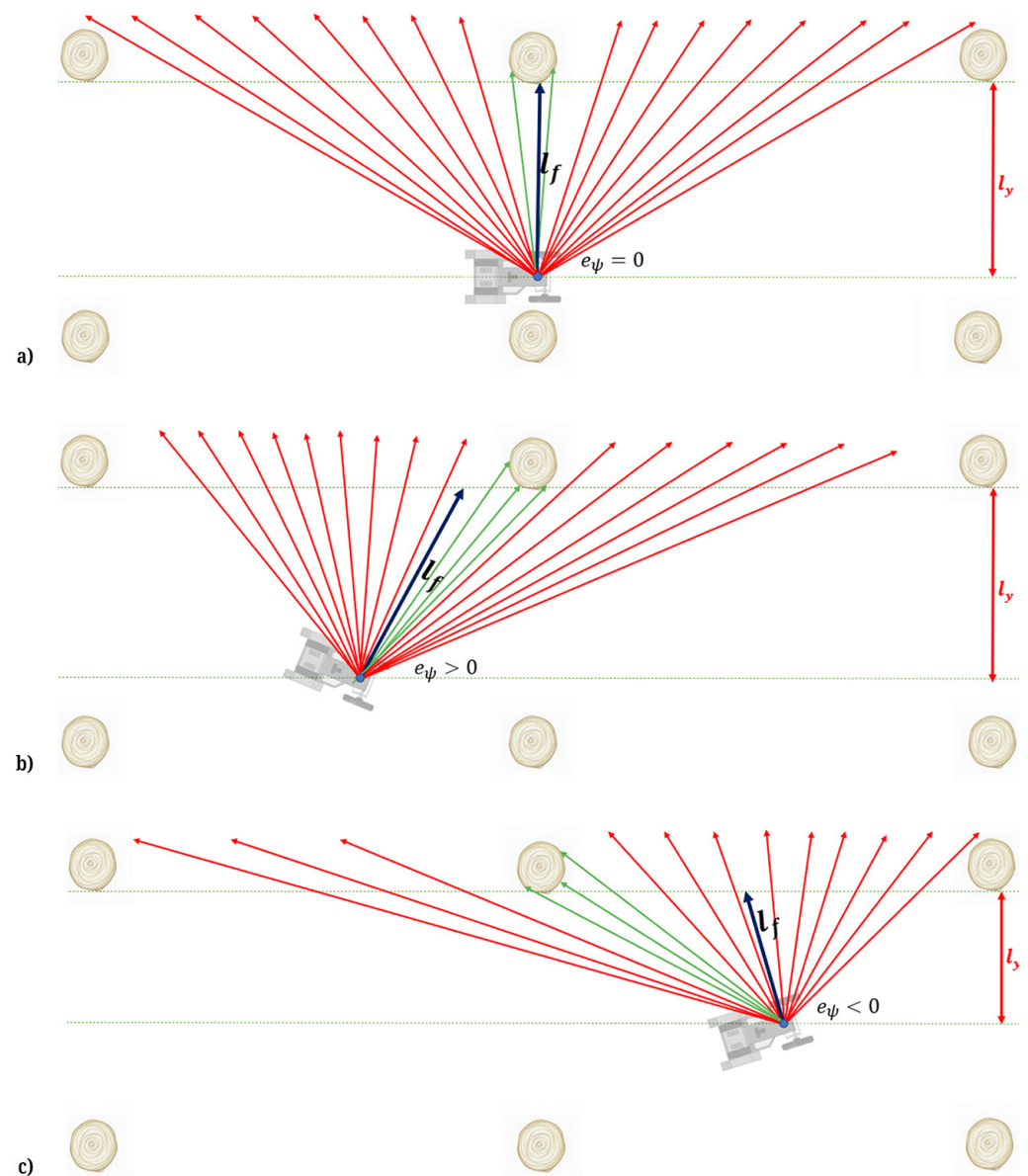


Fig. 11. Effect of heading deviations on body-fixed LiDAR cone

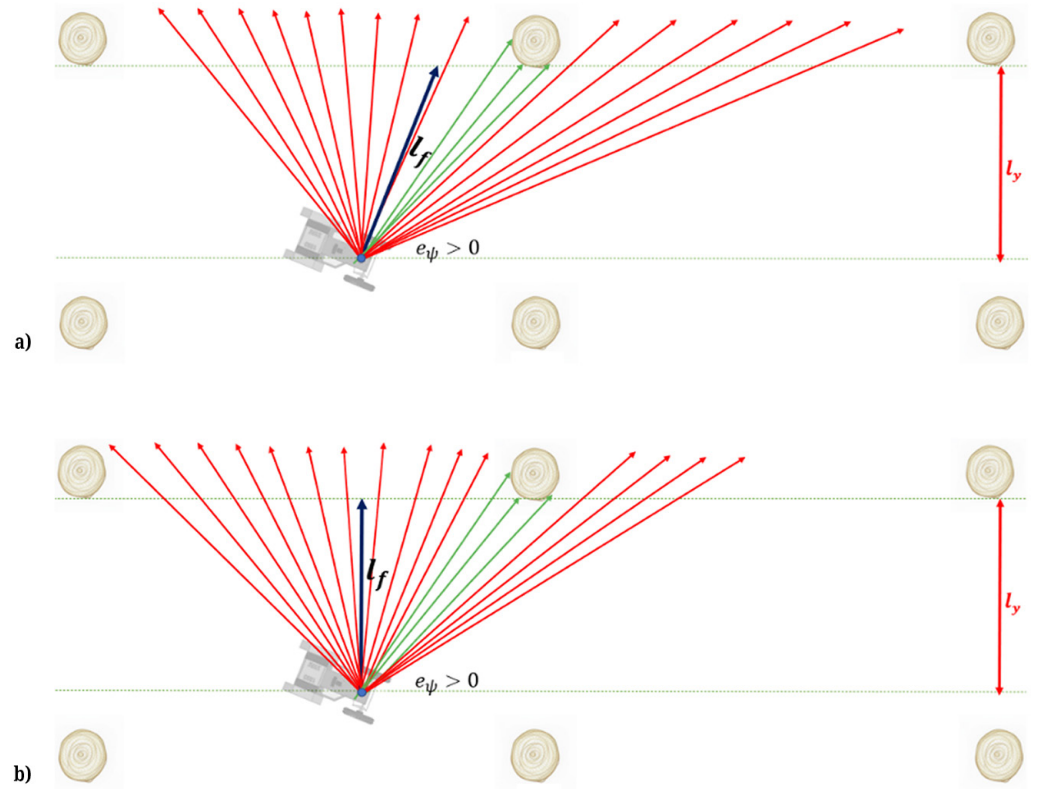


Fig. 12. Adaptive LiDAR cone correction for positive heading error ($e_\psi > 0$)

For l_y estimation, the wide measurement cone was selected according to the 2.0 m row spacing of the plantain plantation. A full aperture of 160° , corresponding to a half-aperture of 80° , provided a balance between collecting enough valid beams from the target crop row and rejecting measurements from nearby rows. In addition, the LiDAR node counts stems using a narrower cone centered at the same adaptive cone center, as shown in Figure 14. This cone uses a half-aperture of $\alpha_n = 10^\circ$ and a maximum projected distance of $l_{n,max} = 1.5$ m. Let $\tilde{\alpha}_i(k) = \text{wrap}_\pi(\alpha_i - \alpha_c(k))$ be the wrapped local angle of the i -th LiDAR beam relative to the adaptive cone center. A LiDAR beam is considered a stem candidate when it satisfies Equation (5).

$$-\alpha_n \leq \tilde{\alpha}_i(k) \leq \alpha_n \text{ and } 0 < l_i(k) \cos(\tilde{\alpha}_i(k)) \leq l_{n,max} \quad (5)$$

Let $N_n(k)$ be the number of stem candidates in the k -th scan. A stem event is declared when $N_n(k) \geq N_{min} = 3$ and $t_k - t_{last} > T_{ref}$ with $T_{ref} = 24$ s, to avoid counting the same stem multiple times while the robot remains nearby. When a valid event is detected, the stem index is incremented and published through the stem topic, as shown in Figure 8. This index provides a rough longitudinal reference for the mission sequence, links camera snapshots to approximate positions along the crop rows, and is reset to zero during turns to restart the count in the next row-following segment.

The WallFollower node connects perception and actuation, as shown in Figure 8. It subscribes to the lateral-distance topic carrying l_y , the stem-index topic (*stem*), and the yaw (ψ) topic, and publishes δ_d and ψ_d . The LiDAR node also subscribes to ψ and ψ_d , using ψ_d to direct the adaptive measurement cone toward the target row-normal direction. At start-up, the robot was manually aligned with the crop row, and the initial heading ψ_0 was used as the desired heading for the first row-following segment. The outer loop controls the lateral offset with respect to the crop row by computing the distance error as defined in Equation (6).

$$e_l(k) = l_d - l_y(k) \quad (6)$$

where $l_d = 1.0 \text{ m}$ is the desired lateral distance, $l_y(k)$ is the LiDAR-based lateral distance estimate at sample k , and $e_l(k)$ is the input of the fuzzy controller. With this sign convention, $e_l(k) < 0$ means that the robot is farther than desired from the followed crop row, whereas $e_l(k) > 0$ means that it is closer than desired. The fuzzy controller converts $e_l(k)$ into the steering reference δ_d , which is limited to $\pm 45^\circ$ and smoothed to prevent sudden steering changes.

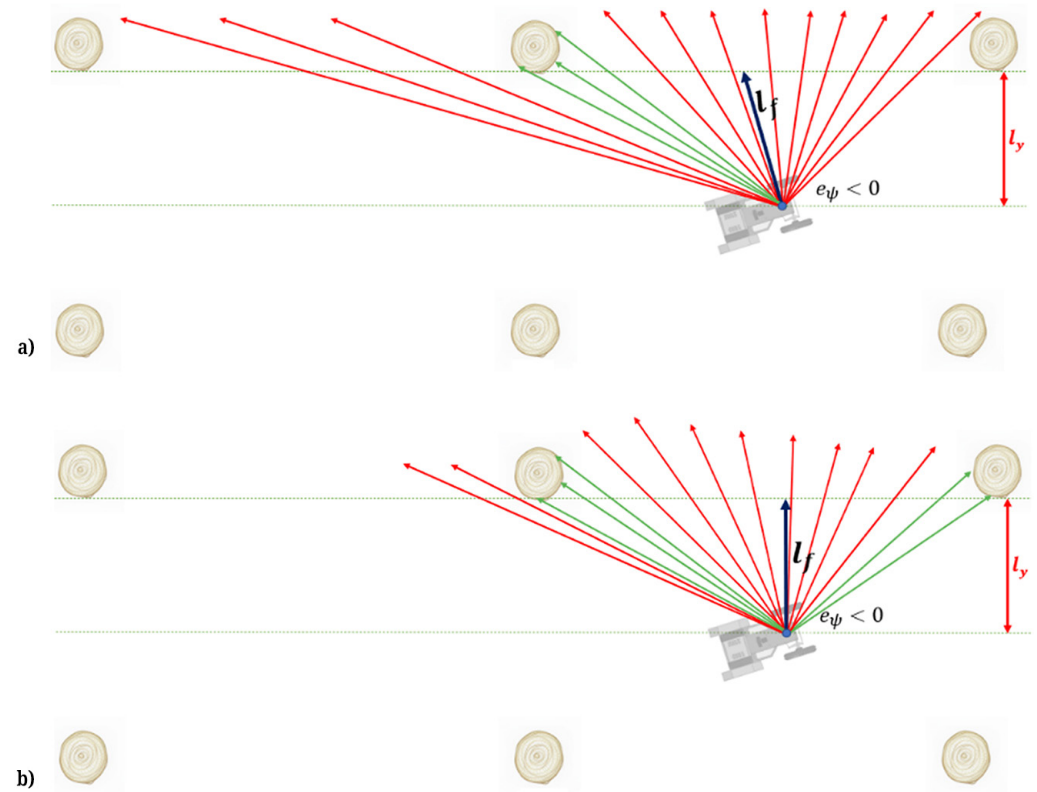


Fig. 13. Adaptive LiDAR cone correction for negative heading error ($e_\psi < 0$)

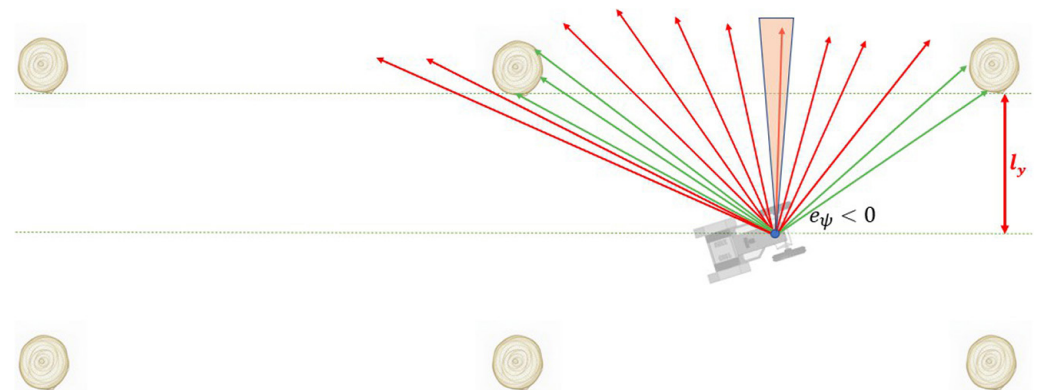


Fig. 14. Narrow adaptive LiDAR cone for stem counting

Figure 15 shows the cascade control structure used to connect the navigation-level lateral-distance control with the embedded steering-angle control. The outer loop runs in ROS 2 and uses the LiDAR-based lateral distance l_y , that is, the estimated lateral offset with respect to the crop row, as feedback. This value is compared with

the desired distance l_d to compute the lateral distance error defined in Equation (6), which the fuzzy controller converts into the desired steering angle δ_d .

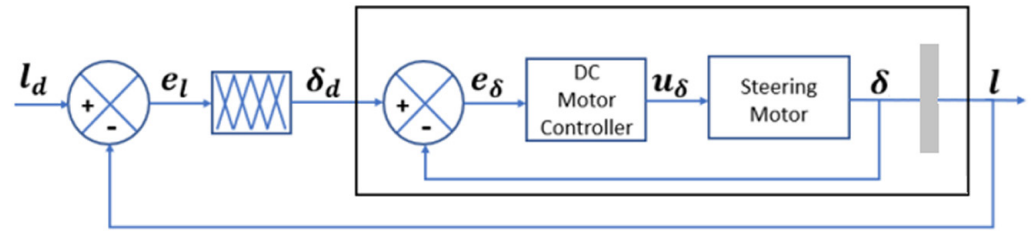


Fig. 15. Cascade control architecture: outer fuzzy lateral-distance loop in ROS 2 and inner steering angle loop in ESP32

The gray block in Figure 15 represents the functional effect of the steering angle on the vehicle path and, consequently, on the LiDAR-based lateral distance estimate, rather than a detailed mathematical vehicle model. The inner steering loop runs on the ESP32, where δ_d is compared with the measured steering angle δ from the encoder to compute the steering position error defined in Equation (7). The embedded controller uses this error to generate the steering control action applied to the steering motor H-bridge. For diagnostics, the ESP32 sends the desired and measured steering angles as tagged ASCII lines, which SerialBridge publishes as ROS 2 topics; additionally, the WallFollower node records the lateral distance error in a CSV file for subsequent performance analysis.

$$e_\delta(k) = \delta_d(k) - \delta(k) \quad (7)$$

In this study, the outer fuzzy controller was designed around the desired lateral distance. Its input, the lateral distance error e_l , was defined over the range $[-1, 1]$ m using five triangular membership functions: negative large NL, negative small NS, zero Z, positive small PS, and positive large PL, as shown in Figure 16. This range covers the expected lateral deviations during low-speed row following while making the controller less sensitive to small measurement variations near the target distance. The output, the desired steering angle δ_d , was limited to $[-45^\circ, 45^\circ]$, matching the Ackermann steering limits, and was described by five labels: left large LL, left small LS, zero Z, right small RS, and right large RL, as shown in Figure 17.

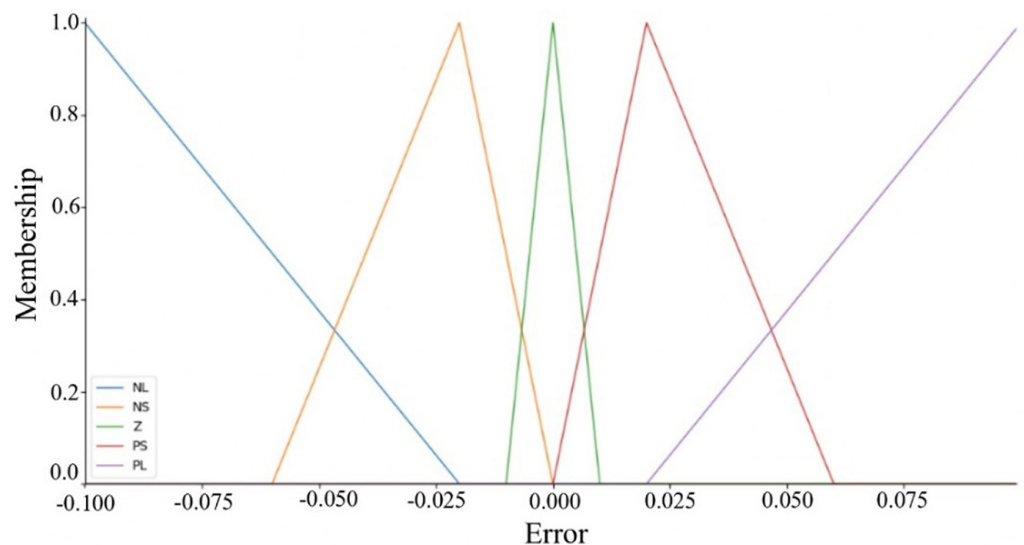


Fig. 16. Fuzzy sets for e_l

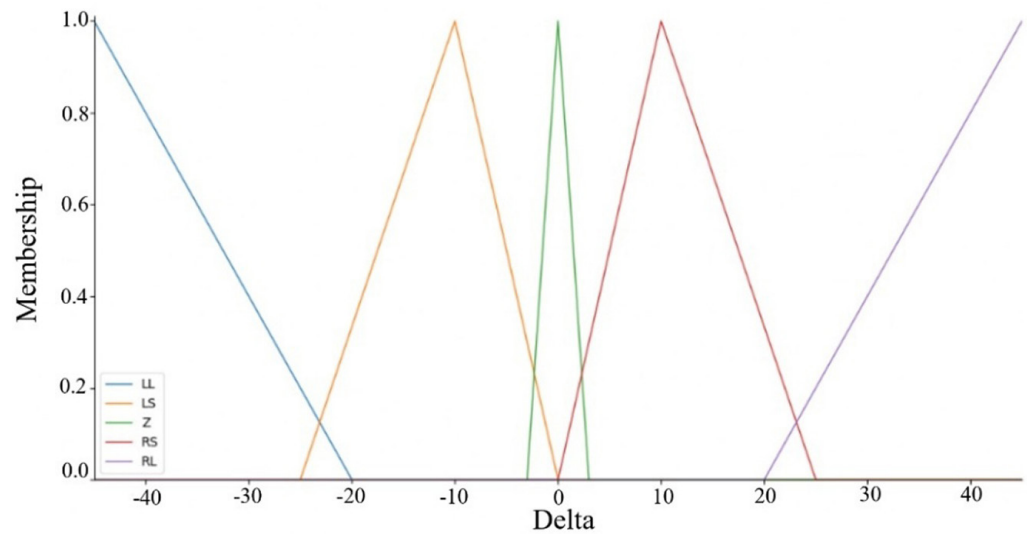


Fig. 17. Fuzzy sets for δ_d

The rule base in Table 1 defines a direct mapping between the lateral error and the steering correction. When the robot is farther than desired from the crop row, the controller steers toward the row; when it is too close, the controller steers away from it. Small errors produce minor or zero steering corrections, whereas large errors generate stronger actions within the mechanical steering limits. The membership functions and rules were selected from practical experience with low-cost agricultural robots and constrained by the platform speed, row spacing, and steering range. Future work will include sensitivity analysis and model-based controller design.

Table 1. Fuzzy rule base ($e_l \rightarrow \delta_d$)

Error Label for e_l	Steering Label for δ_d
NL	LL
NS	LS
Z	Z
PS	RS
PL	RL

In addition to continuous control, the WallFollower node implements the four-state mission sequence shown in Figure 18: FOLLOW, PREPARE, TURNING, and STOPPED. In FOLLOW, the fuzzy controller regulates the lateral distance to the crop row; in PREPARE, the robot moves straight for a short distance after detecting the last stem; in TURNING, it performs an open-loop row-to-row maneuver; and in STOPPED, it pauses briefly before starting the next row-following segment. Row-end detection relies on the stem index and the reliability of the LiDAR-based lateral-distance estimate l_y , whereas turn completion is identified from the yaw angle ψ when the chassis rotation is approximately 180° . After each row-to-row maneuver, the desired heading is updated so that the LiDAR node can keep the adaptive measurement cone aligned with the row-normal direction during the return segment. The updated heading is computed using Equation (8), where ψ_d^{fwd} and ψ_d^{bwd} are the desired headings for the forward and return row-following segments, respectively, and $wrap_\pi(\cdot)$ maps the resulting angle to $(-\pi, \pi]$.

$$\psi_d^{bwd} = wrap_\pi(\psi_d^{fwd} + \pi) \quad (8)$$

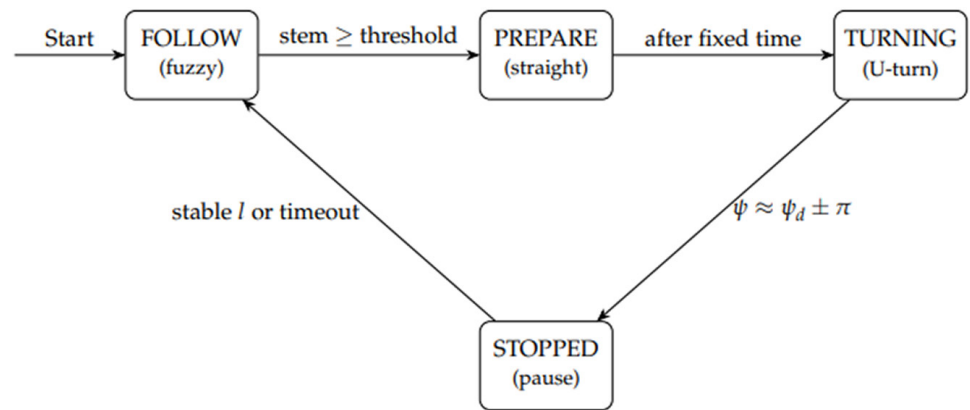


Fig. 18. WallFollower mission state machine

In the ROS 2 graph, the snapshot module (see Figure 8) listens to the (stem) topic to organize camera frames for later analysis. During field runs, the camera stream was sampled at regular intervals, selected frames were saved as JPEG files, and a CSV log recorded the filename and time metadata, including ROS time and image-header timestamp. The stem index was also included in the filename using the pattern (img_YYYYMMDD_HHMMSS_mmm_stem.ext), providing a basic alignment between images and navigation logs.

To reduce under-canopy variations caused by exposure, shadows, and color changes, the snapshots were enhanced offline using reference images. In the HSV color space, adaptive gamma correction was applied to the (V) channel to approach the reference brightness, with $\epsilon = 10^{-6}$ and γ limited to [0.75, 0.95]. Saturation was increased only within the banana-leaf green hue range $H \in [17, 43]$ in OpenCV units, using a factor limited to [1.0, 1.6]. Finally, local contrast was slightly enhanced by applying CLAHE to the CIELAB L^* channel, with a clip limit of 1.5 and an 8×8 tile grid. This procedure improved image consistency before offline analysis.

The enhanced images were processed offline using the BananaSqueezeNet classifier [25]. This lightweight model was originally developed for the Banana Leaf Spot Disease (BananaLSD) dataset, which includes four classes: healthy, Pestalotiopsis, Sigatoka, and Cordana. In [25], the dataset contained 937 images, with 777 for training, 80 for validation, and 80 for testing; the test set was balanced with 20 images per class. The training set was expanded to reduce class imbalance and increase diversity, whereas the validation and test sets remained unchanged. The performance and complexity metrics reported for BananaSqueezeNet on the original BananaLSD test set are summarized in Table 2, supporting its use as a lightweight offline inference tool in the proposed robotic workflow.

Table 2. BananaSqueezeNet reference metrics on the original BananaLSD test set

Metric	Value Reported in [25]
Accuracy	96.25%
Precision	96.53%
Recall	96.25%
Specificity	98.75%
F1-score	96.17%
Matthews correlation coefficient (MCC)	95.13%
Model size	4.78 MB
Number of parameters	737.48 k

In this study, the classifier was not retrained. Instead, each enhanced robot snapshot was resized to 224×224 pixels, converted to RGB, normalized to [0, 1], and processed by the pretrained model. For each image, the network produced a softmax probability vector $p = [p_1, p_2, p_3, p_4]$, where each component represents the probability assigned to one of the four classes and $p_1 + p_2 + p_3 + p_4 = 1$. These probabilities were saved in a CSV file together with the time tag and stem index. A row identifier was then generated from the stem-index reset between rows, producing a compact probability log for plotting and visual inspection.

Because the images in this study were captured in a different under-canopy plantain field environment and did not have expert annotations, the performance metrics reported in [25] were used only as reference values for the classifier on its original dataset. Therefore, the image analysis presented here should be interpreted as an offline probability-based screening process, not as an independent diagnostic validation of disease classification on robot-acquired field images.

To preserve the sequence within each row despite uneven image sampling, snapshots were first sorted by acquisition time and then grouped by row identifier and stem index. For each group associated with a stem and containing N_s snapshots, a continuous traversal coordinate was assigned using Equation (9), where s is the integer stem index obtained from the LiDAR-based counting routine, N_s is the number of snapshots associated with that stem, and k is the snapshot order within the group. This coordinate preserves the discrete stem index while distributing multiple snapshots from the same stem along the horizontal axis of the probability maps.

$$stem_{norm}(k) = s + \frac{k}{N_s}, \quad k = 0, 1, \dots, N_s - 1 \quad (9)$$

Probability maps were generated using $stem_{norm}$, the continuous traversal coordinate defined in Equation (9) as the horizontal axis and the row identifier as the vertical axis. A small vertical jitter of ± 0.03 was added to reduce point overlap. Two map types were produced: (i) a dominant-class map, obtained from $\arg \max_j p_j$, and (ii) class-specific probability maps using the intervals [0, 0.25), [0.25, 0.50), [0.50, 0.75), and [0.75, 1.0].

The ESP32 firmware implements the embedded steering and propulsion logic corresponding to the inner loop in Figure 15. It performs three main tasks: acquiring the steering angle δ from the quadrature encoder and the yaw angle ψ from the BNO055 IMU, receiving and validating the steering reference δ_d sent by ROS 2, and driving the steering and rear propulsion motors. The steering encoder was read every $T_s = 10$ ms, and the estimated steering angle was reported back to the ROS 2 layer as a scalar value. The reference δ_d was received through the serial port as an ASCII floating-point value in degrees and checked against predefined safety limits to protect the Ackermann steering mechanism. If δ_d exceeded the allowed range, all PWM outputs for the steering and propulsion motors were set to zero, and the control logic was bypassed for that cycle. Otherwise, the firmware computed the steering position error from the difference between δ_d and the measured angle δ , as defined in Equation (7), and used a rule-based discrete-time position-tracking logic to command the steering motor H-bridge until the measured angle was sufficiently close to the reference.

This embedded strategy was selected because the robot operated at low speed, and preliminary tests showed that rule-based steering was sufficient to track the paths generated by the ROS 2 fuzzy controller. During turns, the firmware also slightly adjusted the PWM commands of the rear motors to improve maneuverability without requiring additional sensors or computational resources. Therefore, this implementation should be interpreted as a low-cost practical solution for validating

the navigation concept under real field conditions, rather than as an optimal or model-based steering controller. Future work will replace this rule-based logic with a discrete-time feedback controller and include a detailed evaluation of steering dynamics, sampling time, saturation effects, and drivetrain differences.

3 RESULTS

This section presents two sets of results: (i) crop-row-following performance using the LiDAR-IMU navigation strategy, including maneuver-aware segmentation and tracking error statistics; and (ii) offline mapping of onboard RGB snapshots, which provides spatial summaries of leaf-condition class probabilities along the traveled paths.

3.1 Maneuver-aware tracking performance

A representative multi-signal time-series plot was included to illustrate the typical behavior of the main recorded variables and the maneuver-aware segmentation used in the experiments. Similar plots were generated for all trials and used to compute the quantitative metrics reported below. Each trial followed the same protocol: the robot was manually placed at the start, approximately 1 m from the left crop row, as shown in Figure 1c, and then followed the first row while tracking the left crop wall and counting five stems. After detecting the fifth stem, the robot performed an open-loop row-to-row maneuver and continued tracking the second row along the return path. To evaluate tracking performance, the two row-following segments were analyzed separately, excluding the maneuver interval.

Figure 19 shows the maneuver-aware segmentation for a representative trial. From top to bottom, the subplots show: (i) the lateral distance error e_l with respect to the left crop wall; (ii) the yaw angle ψ and the row-wise heading reference ψ_{ref} shown as a dashed line; (iii) the orientation error e_ψ used to align the LiDAR cone with the target row-normal direction; and (iv) the valid LiDAR readings indicator (VLR) used by the stem-counting routine, with markers 1–5. The vertical dashed lines delimit the open-loop maneuver window.

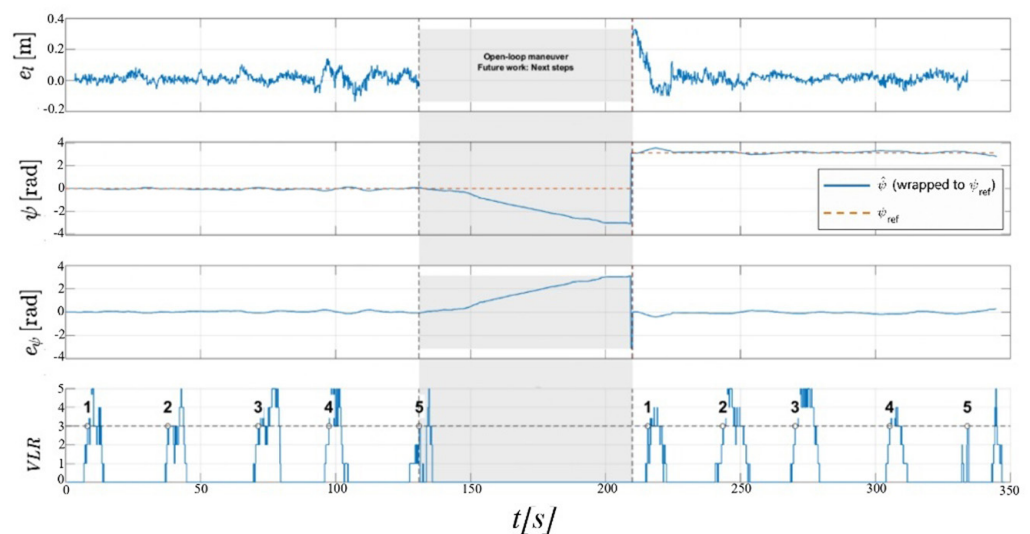


Fig. 19. Representative trial tracking signals with stem markers and maneuver windows

During row following, both e_l and e_ψ remained close to zero, indicating stable lateral control and consistent cone alignment despite canopy occlusions and uneven terrain. During the maneuver, ψ changed toward the next row direction, approximately a π rad rotation, producing a temporary increase in e_ψ and a reduction in VLR. Once closed-loop tracking resumed, the robot briefly adjusted to the new left crop wall, after which both errors returned to their normal operating ranges.

Figure 20 shows Trials P1–P5 for the main tracking variables e_l and e_ψ , with the maneuver limits marked in each trial. For consistent comparison, quantitative performance metrics were computed separately for the pre-maneuver row-following segment (row-following 1) and the post-maneuver segment (row-following 2), excluding the open-loop maneuver period. These metrics were calculated for both the lateral distance error e_l and the orientation error e_ψ .

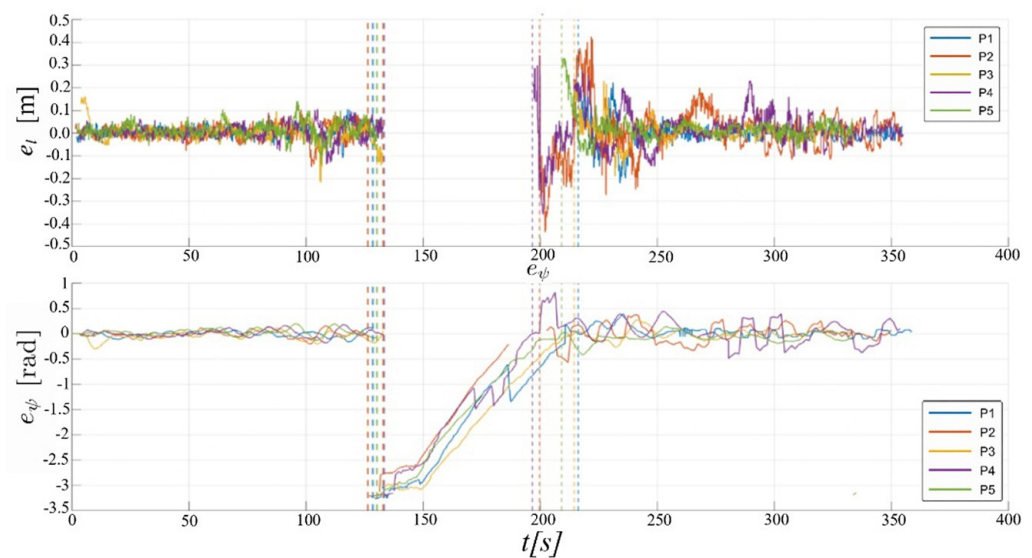


Fig. 20. Overlay of lateral and heading errors across the trials

Tracking performance was evaluated using the mean absolute error (MAE) and root mean square error (RMSE), together with robust indicators: the 95th percentile of the absolute error (P95) and the percentage of samples within acceptable limits $|e_l| \leq 0.1$ m and $|e_\psi| \leq 0.2$ rad. These additional indicators support the interpretation of field performance by reducing the influence of occasional outliers. Table 3 reports the MAE and RMSE values for each trial and row-following segment, computed only from tracking samples and excluding the open-loop maneuver intervals. Row-following 1 corresponds to the first tracking segment before the row-to-row maneuver, whereas row-following 2 corresponds to the segment after the maneuver, when the new left crop wall is reacquired. Here, e_l and e_ψ denote the lateral distance error and heading error, respectively.

To complement the MAE/RMSE results and better capture occasional error spikes, Table 4 reports P95 and the percentage of samples within the practical tolerance bands. Row-following 1 remained consistently accurate across trials, whereas row-following 2 showed greater variability, mainly associated with reacquiring the crop row after the maneuver.

Figure 21 presents boxplots of the tracking errors across trials, excluding the open-loop maneuver intervals. For e_p , the medians remained close to zero and the

interquartile ranges were small, indicating that the controller generally maintained the desired left-wall distance. The e_ψ distributions showed greater variability, reflecting sensitivity to canopy gaps and row re-acquisition after maneuvers. The observed outliers were associated with temporary perception issues and transient realignment events, supporting the use of robust statistics such as those reported in Table 4.

Table 3. MAE and RMSE tracking metrics for each trial and row-following segment

Trial	Segment	MAE (e_l) [m]	RMSE (e_l) [m]	MAE (e_ψ) [rad]	RMSE (e_ψ) [rad]
P1	Row-following 1	0.0193	0.0255	0.0430	0.0546
P1	Row-following 2	0.0331	0.0561	0.0987	0.1322
P2	Row-following 1	0.0258	0.0339	0.0528	0.0677
P2	Row-following 2	0.0785	0.1097	0.1471	0.1894
P3	Row-following 1	0.0278	0.0403	0.0701	0.0928
P3	Row-following 2	0.0320	0.0478	0.0747	0.0990
P4	Row-following 1	0.0256	0.0342	0.0556	0.0717
P4	Row-following 2	0.0605	0.0866	0.1953	0.2663
P5	Row-following 1	0.0275	0.0365	0.0579	0.0741
P5	Row-following 2	0.0356	0.0602	0.1322	0.3633

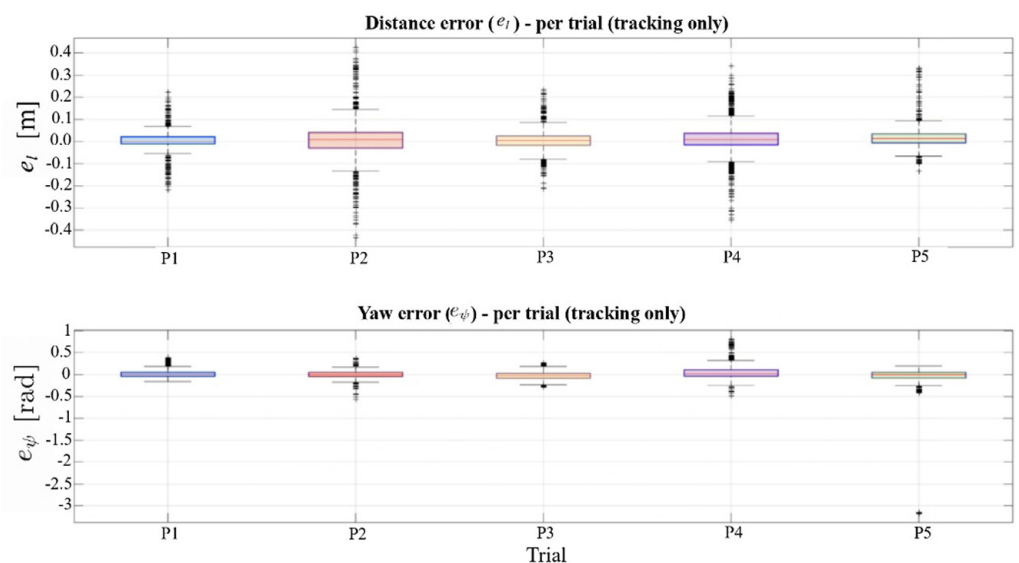


Fig. 21. Error distribution across the trials. Boxplots of e_l (top) and e_ψ (bottom)

Across trials, Row-following 1 showed consistently low errors, with median [IQR] values of $MAE(e_l) = 0.0258$ [0.0019] m, $RMSE(e_l) = 0.0342$ [0.0026] m, $MAE(e_\psi) = 0.0556$ [0.0051] rad, and $RMSE(e_\psi) = 0.0717$ [0.0064] rad. Row-following 2 showed greater variability, with $MAE(e_l) = 0.0356$ [0.0274] m, $RMSE(e_l) = 0.0602$ [0.0305] m, $MAE(e_\psi) = 0.1322$ [0.0484] rad, and $RMSE(e_\psi) = 0.1894$ [0.1341] rad. The robust statistics showed a similar trend: median P95(e_l) and (P95 e_ψ) increased from 0.0683 [0.0110] m and 0.1540 [0.0233] rad in Row-following 1 to 0.1452 [0.0951] m and 0.3207 [0.0785] rad in Row-following 2, respectively, reflecting the challenge of crop-row re-acquisition after the maneuver.

Table 4. Robust tracking metrics for each trial and row-following segments

Trial	Segment	P95($ e_l $) [m]	Samples Within $ e_l \leq 0.1$ m [%]	P95($ e_v $) [rad]	Samples Within $ e_v \leq 0.2$ rad [%]
P1	Row-following 1	0.0494	99.64	0.1086	100.00
P1	Row-following 2	0.1452	89.17	0.2937	85.06
P2	Row-following 1	0.0646	98.44	0.1353	99.87
P2	Row-following 2	0.2598	77.36	0.3722	73.74
P3	Row-following 1	0.0812	95.93	0.1955	95.49
P3	Row-following 2	0.1041	94.28	0.2334	92.32
P4	Row-following 1	0.0683	98.69	0.1540	100.00
P4	Row-following 2	0.1992	79.94	0.6466	61.19
P5	Row-following 1	0.0756	99.18	0.1586	100.00
P5	Row-following 2	0.0979	95.13	0.3207	92.07

In addition to navigation evaluation, the platform was assessed for structured under-canopy image capture and offline probability mapping along crop rows. Snapshots were saved for each trial (P1–P5), together with a CSV log containing timestamps and filenames to preserve acquisition order. Because the robot-acquired images did not have expert image-level annotations, no new accuracy, precision, recall, F1-score, or confusion matrix was computed for this field dataset. Instead, the classifier metrics reported in [25] were used only as reference values for BananaSqueezeNet on the original BananaLSD dataset.

Before inference, reference-based photometric normalization was applied per trial to reduce under-canopy variability caused by shadows, specularities, and foliage motion. A well-exposed snapshot was selected as the reference for each trial folder, and the remaining images were enhanced using bounded adaptive gamma correction on the HSV (V) channel, selective saturation adjustment on foliage-like pixels through a green-hue mask, and CLAHE on the CIELAB luminance channel. Each enhanced snapshot was then processed offline with BananaSqueezeNet, producing a four-class probability vector and a dominant class label. The predictions were exported to per-trial CSV files and consolidated into a map-ready format by adding traversal indices, including the corridor identifier, stem index, and continuous traversal coordinate.

Figure 22 shows the crop layout and corridor traversed during each test for the offline image analysis. The robot followed the left-side crop wall with a nominal lateral offset of approximately 1.0 m from the plants; therefore, the upward-facing camera captured foliage from both the left and right rows while moving along the corridor.

Figure 23 illustrates the 2D indexing scheme used to assign each snapshot to a point according to the corridor index and normalized traversal coordinate, enabling direct visualization of the predicted conditions along the robot path. Based on this indexing, Figure 24 presents the class-specific probability maps generated by the offline classifier. Each subfigure corresponds to one class—healthy, Sigatoka, Pestalotiopsis, or Cordana—and each point represents a robot-acquired snapshot. The horizontal axis shows the normalized traversal coordinate, whereas the vertical axis separates the corridor positions. Point intensity indicates the predicted probability of the class shown in that subfigure, grouped into four ranges: 0–25%,

25–50%, 50–75%, and 75–100%. Thus, darker points indicate higher class probability, allowing regions with consistently stronger class responses to be identified along the trajectory.

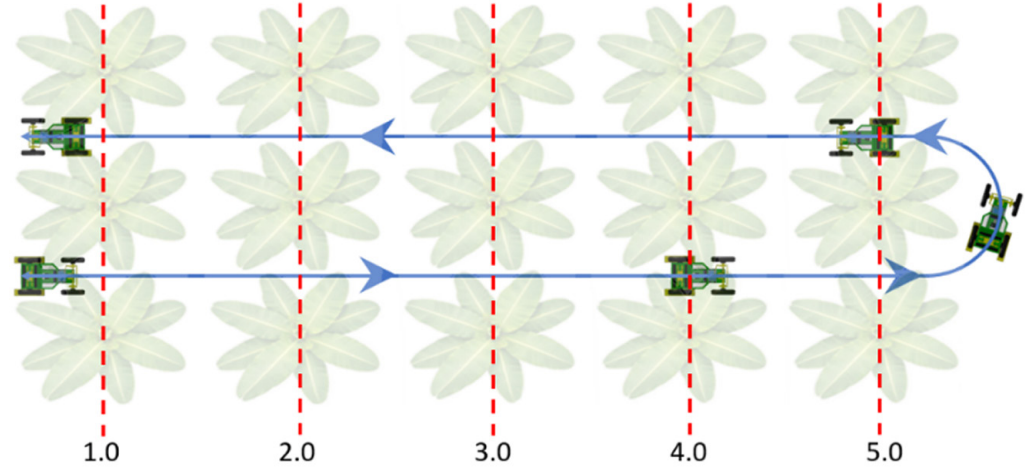


Fig. 22. Field layout and test corridor for offline image analysis

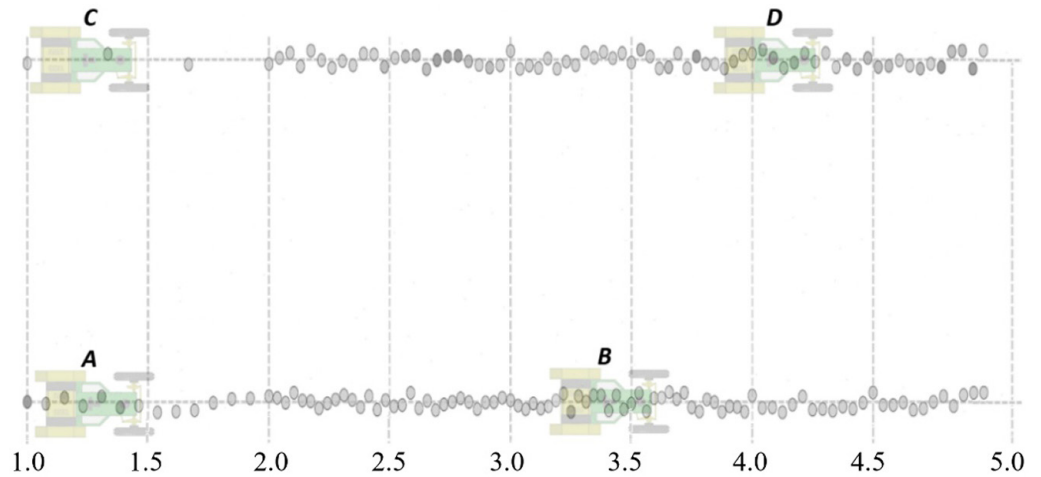


Fig. 23. Snapshot indexing for 2D probability mapping

For the per-trial summary statistics, let $\mathcal{T}$ denote the set of snapshots in a given trial, and let $\mathcal{C} = H, S, P, C$ denote the set of considered classes, corresponding to healthy, Sigatoka, Pestalotiopsis, and Cordana, respectively. For each snapshot (i), the classifier output is represented by the softmax probability vector:

$$p_i = [p_i^{(H)}, p_i^{(S)}, p_i^{(P)}, p_i^{(C)}], \sum_{c \in \mathcal{C}} p_i^{(c)} = 1, \hat{c}_i = \arg \max_{c \in \mathcal{C}} p_i^{(c)} \quad (10)$$

where $p_i^{(c)}$ is the predicted probability of class (c) for snapshot (i), and $\hat{c}_i$ is the dominant class. The mean predicted probability and the dominant-class frequency for each class were computed as described in Equations (11) and (12), where $1(\cdot)$ is the indicator function. Thus, $\bar{p}(c)$ reports the average probability assigned to class (c), whereas $f_{\text{dom}}^{(c)}$ reports the percentage of snapshots for which class (c) was the dominant prediction. To summarize prediction certainty, the highest softmax probability was also computed for each snapshot as described in Equation (13).

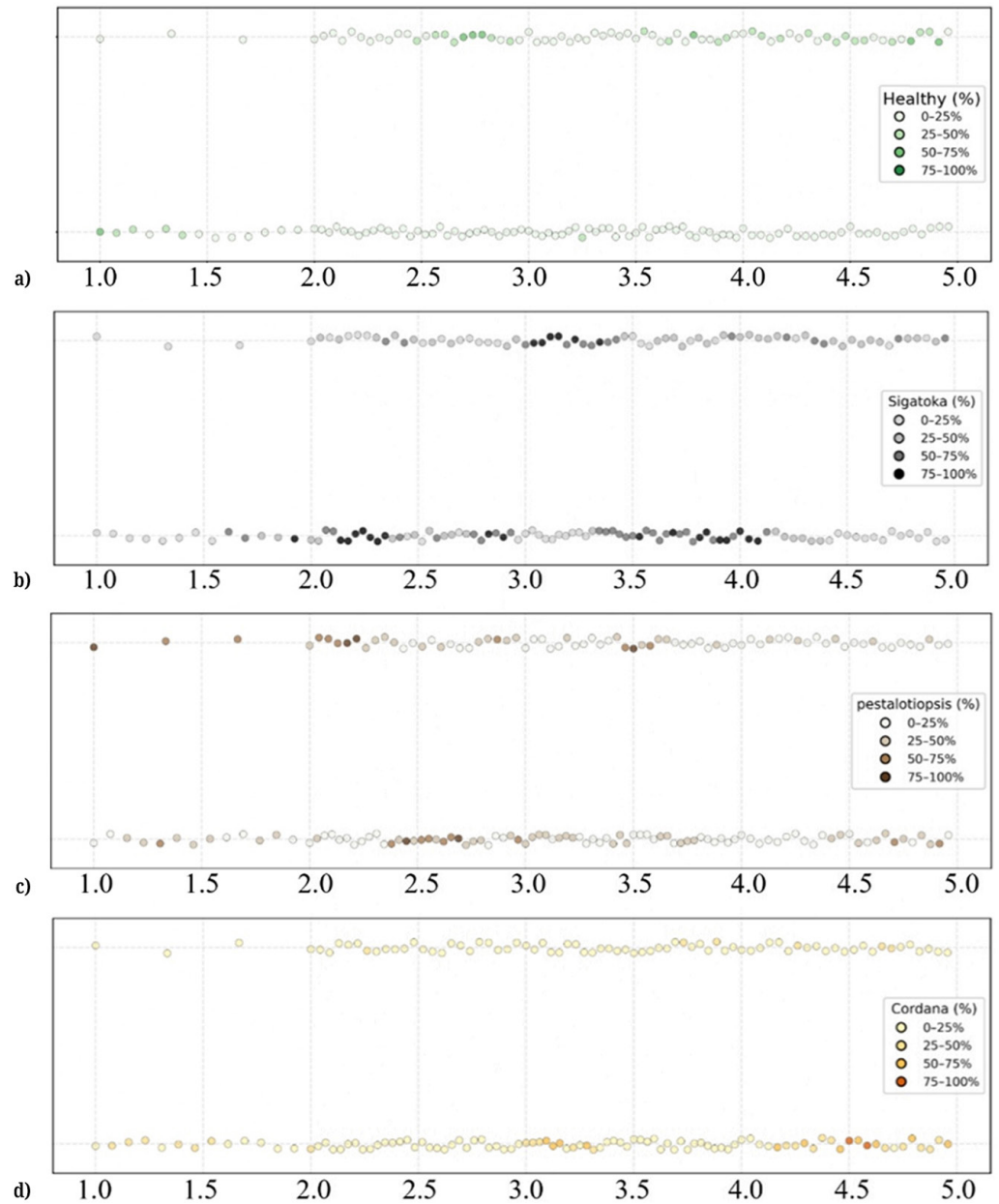


Fig. 24. Class-specific probability maps from the offline classifier

$$\bar{p}^{(c)} = \frac{1}{|T|} \sum_{i \in T} p_i^{(c)} \quad (11)$$

$$f_{\text{dom}}^{(c)} = \frac{100}{|T|} \sum_{i \in T} 1(\hat{c}_i = c) \quad (12)$$

$$p_{\max, i} = \max_{c \in \mathcal{C}} p_i^{(c)} \quad (13)$$

Figure 25 shows a simplified dominant-class map derived from the softmax probability outputs. Each snapshot is labeled only with the class having the highest predicted probability, rather than displaying the four class probabilities separately. This visualization complements Figure 24 by highlighting the dominant predicted leaf-condition label at each sampled location along the mapped path.

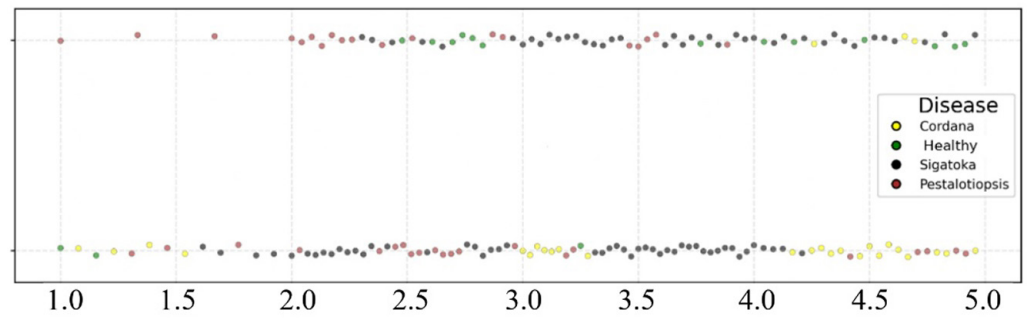


Fig. 25. Dominant class map from the offline classifier

Table 5 summarizes the offline inference results obtained from the combined CSV files of the five field trials, including the number of processed snapshots, average predicted probability per class, dominant-class frequency, and average maximum softmax probability as a model-confidence indicator. Sigatoka was the most frequent dominant label across trials, followed consistently by Pestalotiopsis. These results should be interpreted as probability-based screening summaries produced by a lightweight classifier previously evaluated on its original dataset, not as final disease diagnoses for robot-acquired field images. Together, the spatial maps and per-trial summaries show that repeatable low-cost navigation can support structured under-canopy image logging and map-like outputs for guiding future localized inspections by trained personnel.

Table 5. Per-trial probability and dominant-class summaries from offline inference

Trial	N	$\bar{p}^{(H)}$	$\bar{p}^{(S)}$	$\bar{p}^{(P)}$	$\bar{p}^{(C)}$	Dom (H) %	Dom (S) %	Dom (P) %	Dom (C) %	$\bar{p}_{max}$
P1	173	0.140	0.416	0.278	0.166	9.2	51.4	23.1	16.2	0.579
P2	244	0.184	0.342	0.297	0.176	18.9	38.5	25.8	16.8	0.583
P3	217	0.169	0.406	0.261	0.165	12.0	48.4	24.4	15.2	0.563
P4	241	0.212	0.379	0.263	0.147	23.7	41.1	23.2	12.0	0.596
P5	189	0.178	0.381	0.306	0.135	13.8	42.9	32.8	10.6	0.584
All	1064	0.179	0.382	0.280	0.158	16.1	44.0	25.8	14.2	0.581

4 CONCLUSIONS AND FUTURE WORKS

This study presented a low-cost, small-scale, car-like agricultural robot for under-canopy row following and structured image capture in plantain (*Musa paradisiaca*) plantations. The platform integrates off-the-shelf sensing components, including a 2D LiDAR, low-cost IMU, RGB camera, ESP32-based actuation layer, and a modular ROS 2 workflow. The proposed architecture was designed to support reproducible field experiments and to provide an affordable robotic testbed for crop-row navigation and under-canopy data collection in resource-limited agricultural contexts.

The navigation strategy combined LiDAR-based lateral distance estimation with an adaptive measurement cone guided by the heading error e_ψ . In this approach, the IMU yaw angle was not used as a direct stabilizing control action but as a geometric

correction to keep the LiDAR measurement sector aligned with the crop-row normal direction when the robot was moderately misaligned with the row. This helped preserve the geometric interpretation of the lateral distance estimate l_y during row following and after row-to-row maneuvers.

Field tests across five trials provided preliminary evidence that the platform can perform under-canopy row following in real plantain plantations. The first row-following segment showed consistently low tracking errors, with median MAE and RMSE of the lateral error equal to 0.0258 m and 0.0342 m, respectively. After the row-to-row maneuver, the median MAE and RMSE increased to 0.0356 m and 0.0602 m, respectively, reflecting the additional difficulty of reacquiring the new left crop wall. These results indicate that the proposed low-cost LiDAR-IMU workflow can support stable navigation under the tested conditions, although broader validation is still required across different crop layouts, terrain conditions, canopy structures, weather conditions, and operating speeds.

In addition to navigation, the robot demonstrated a complete workflow for under-canopy image acquisition and offline probability-based analysis. RGB snapshots were indexed using LiDAR-based stem counting and organized along the traveled rows without GNSS. After photometric normalization, the images were processed offline using BananaSqueezeNet to generate class-probability vectors and map-like visual summaries of leaf-condition patterns. Because the robot-acquired images did not have expert image-level annotations, these results should not be interpreted as an independent diagnostic validation. Instead, they show that repeatable low-cost navigation can structure under-canopy image logging and produce probability-based summaries that may guide future localized inspections by trained personnel.

Future work will focus first on improving the embedded control layer. The current rule-based steering logic on the ESP32 will be replaced with a discrete-time feedback controller that explicitly considers sampling time, saturation, anti-windup, and steering-rate limits. The rear traction motors should also be upgraded from open-loop PWM operation to closed-loop velocity control using wheel encoders, enabling more consistent speed regulation on mud, slopes, leaf litter, and uneven soil. These improvements will support a more systematic analysis of tracking performance as a function of speed, terrain, drivetrain asymmetry, and steering dynamics.

Further work is also required to strengthen the LiDAR perception pipeline under canopy variability. Although the adaptive cone improves geometric consistency, occlusions, leaf motion, row gaps, sunlight, and intermittent returns can still affect distance estimation and stem counting. Future versions should evaluate robust filtering of projected LiDAR beams, validity masks, simple clustering rules, adaptive cone aperture, sensor shielding, and more robust LiDAR units. A controlled ablation study comparing a body-fixed cone and the proposed heading-adaptive cone under the same field paths is also necessary to quantify the isolated effect of the cone correction. In addition, the valid LiDAR readings indicator can be used as an internal perception-quality signal to trigger slowing down, stopping, realignment, or temporary filtering changes when sensing conditions degrade.

The ROS 2-ESP32 workflow should also be characterized in greater temporal detail. Future experiments should report node execution frequencies, CPU and memory load, serial communication latency, and end-to-end jitter. This characterization is important for reproducibility and for future high-speed navigation or onboard inference, where timing constraints will become more critical.

For plant-health analysis, future work will move from offline probability-based screening toward onboard inference using limited computing resources. This will require robot-acquired datasets with expert annotations, evaluation across different

farms, seasons, varieties, lighting conditions, and disease stages, and class-wise performance reporting under real field conditions. Additional low-cost sensing strategies, such as controlled illumination or affordable multispectral cues, may also improve robustness against shadows, reflections, and canopy variability. The final goal is to evolve from qualitative probability maps toward actionable under-canopy health summaries that support targeted manual inspection and localized crop-management decisions.

Finally, broader experimental validation and improved reproducibility will be necessary before practical deployment. Future studies should include more field trials, longer missions, different planting layouts, crop densities, row spacings, corridor widths, soil moisture levels, terrain conditions, and lighting regimes. Practical baselines, such as GNSS-assisted guidance, vision-based corridor detection, and alternative LiDAR-processing methods, should also be compared. To facilitate adoption by other research groups, future releases should include detailed calibration procedures, bill of materials, CAD files, firmware, ROS 2 launch files, datasets, and processing scripts.

5 ACKNOWLEDGMENT

The authors gratefully acknowledge the support provided by the Ministry of Science, Technology, and Innovation of Colombia (MINCIENCIAS) and the Francisco José de Caldas Fund through the allocation of resources under Call 934 of 2023, contract 332-2023, project 99209. The authors also thank the LASER research group at Universidad Distrital Francisco José de Caldas for their guidance and advice during the development of this study project. Special thanks are extended to BITPOINTER S.A.S., the beneficiary of project 99209, particularly to its manager, Eng. Juan Fajardo Barrero, and its administrative director, lawyer Alexandra Reyes Agudelo, for their valuable contributions and commitment to this initiative. The authors also gratefully acknowledge CODALTEC – Corporación de Alta Tecnología para la Defensa, particularly Yolanda Parra, Director of Research, Development, and Innovation (R&D+i), for her support and contribution to the development of this project.

6 DATA AVAILABILITY

All experimental data, control code, and ROS 2 nodes that underpin the findings of this study are publicly accessible in the GitHub repository <https://github.com/Celenaxz/Robot-el-chino-de-los-mandados>. This repository contains the ESP32 firmware, fuzzy control scripts, field datasets in CSV format, and documentation necessary to replicate the results presented in this study, as well as a detailed list of hardware components and materials used to build the robot prototype.

7 REFERENCES

- [1] Organization for Economic Cooperation Development, *Development Co-Operation Report 2023: Debating the Aid System*. Paris, France: OECD Publishing, 2023. <https://doi.org/10.1787/f6edc3c2-en>
- [2] S. Das, "Transforming agriculture: Harnessing robotics and drones for sustainable farming solution," *J. Exp. Agric. Int.*, vol. 46, no. 7, pp. 219–231, 2024. <https://doi.org/10.9734/jeai/2024/v46i72577>

- [3] Food Agriculture Organization, *The State of Food and Agriculture 2022: Leveraging Automation in Agriculture for Transforming Agrifood Systems*. Rome, Italy: FAO, 2022. <https://doi.org/10.4060/cb9479en>
- [4] M. McCampbell, "Agricultural digitalization and automation in low- and middle-income countries: Evidence from ten case studies," Food and Agriculture Organization of the United Nations, Rome, Italy, FAO Agricultural Development Economics Technical Study 25, 2022. <https://doi.org/10.4060/cc2914en>
- [5] W. Boonneua, S. Chai-Arayalert, and N. Boonnam, "Automated hydroponics notification system using IoT," *International Journal of Interactive Mobile Technologies (ijIM)*, vol. 16, no. 6, pp. 206–220, 2022. <https://doi.org/10.3991/ijim.v16i06.27959>
- [6] E. L. Silva, C. M. Oliveira, A. B. Mendes, and H. M. Guerra, "Technology and innovation in agriculture: The Azores case study," *International Journal of Interactive Mobile Technologies (ijIM)*, vol. 11, no. 5, pp. 56–66, 2017. <https://doi.org/10.3991/ijim.v11i5.7070>
- [7] A. Mehta, A. Barfield, and M. Thomson, "The learning hub – empowering remote rural communities across India through blended learning," *International Journal of Advanced Corporate Learning (ijAC)*, vol. 17, no. 4, pp. 129–132, 2024. <https://doi.org/10.3991/ijac.v17i4.51251>
- [8] C. E. Ospina-Parra, J. A. Valencia-Montoya, J. E. Cardona-Cardona, M. Duque-Rios, F. D. J. Usuga, and C. A. Alvarez Delgado, "Plantain socio-productive system (Musa AAB): An initiative to revalue agriculture in western Antioquia," *Revista de Ciencias Agrícolas*, vol. 42, no. 1, Art. no. e1256, 2025. <https://doi.org/10.22267/rcia.20254201.256>
- [9] E. Raji, T. Ijomah, and O. Eyieyien, "Improving agricultural practices and productivity through extension services and innovative training programs," *Int. J. Appl. Res. Soc. Sci.*, vol. 6, no. 7, pp. 1297–1309, 2024. <https://doi.org/10.51594/ijarss.v6i7.1267>
- [10] J. G. Esguera, M. A. Balendres, and D. P. Paguntalan, "Overview of the Sigatoka leaf spot complex in banana and its current management," *Tropical Plants*, vol. 3, no. 1, Art. no. e002, 2024. <https://doi.org/10.48130/tp-0024-0001>
- [11] N. Jiménez, S. Orellana, B. Mazon-Olivo, W. Rivas-Asanza, and I. Ramírez-Morales, "Detection of leaf diseases in banana crops using deep learning techniques," *AI*, vol. 6, no. 3, Art. no. 61, 2025. <https://doi.org/10.3390/ai6030061>
- [12] D. Luna-Moreno *et al.*, "Early detection of the fungal banana black Sigatoka pathogen *Pseudocercospora fijiensis* by an SPR immunosensor method," *Sensors*, vol. 19, no. 3, Art. no. 465, 2019. <https://doi.org/10.3390/s19030465>
- [13] J. Ugarte Fajardo *et al.*, "Early detection of black Sigatoka in banana leaves using hyper-spectral images," *Appl. Plant Sci.*, vol. 8, no. 8, Art. no. e11383, 2020. <https://doi.org/10.1002/aps3.11383>
- [14] J. Henderson *et al.*, "Black Sigatoka disease: New technologies to strengthen eradication strategies in Australia," *Austral. Plant Pathol.*, vol. 35, no. 2, pp. 181–193, 2006. <https://doi.org/10.1071/AP06017>
- [15] L. H. Jacome and W. Schuh, "Effects of leaf wetness duration and temperature on development of black Sigatoka disease on banana infected by *Mycosphaerella fijiensis* var. *difformis*," *Phytopathology*, vol. 82, no. 5, pp. 515–520, 1992. <https://doi.org/10.1094/Phyto-82-515>
- [16] D. F. Yépez-Ponce, J. V. Salcedo, P. D. Rosero-Montalvo, and J. Sanchis, "Mobile robotics in smart farming: Current trends and applications," *Front. Artif. Intell.*, vol. 6, Art. no. 1213330, 2023. <https://doi.org/10.3389/frai.2023.1213330>
- [17] A. Ahmadi, L. Nardi, N. Chebrolu, and C. Stachniss, "Visual servoing-based navigation for monitoring row-crop fields," Institute of Electrical Electronics Engineers, 2020. <https://doi.org/10.1109/ICRA40945.2020.9197114>

- [18] M. Łągiewska and E. Panek-Chwastyk, "Integrating remote sensing and autonomous robotics in precision agriculture: Current applications and workflow challenges," *Agronomy*, vol. 15, no. 10, Art. no. 2314, 2025. <https://doi.org/10.3390/agronomy15102314>
- [19] A. Abanay, L. Masmoudi, M. El Ansari, J. Gonzalez-Jimenez, and F.-A. Moreno, "LiDAR-based autonomous navigation method for an agricultural mobile robot in strawberry greenhouse: AgriEco Robot," *AIMS Electron. Electr. Eng.*, vol. 6, no. 3, pp. 317–328, 2022. <https://doi.org/10.3934/electreng.2022019>
- [20] P. Correa, A. Barranco, F. Pérez, P. Bautista, and H. Rostro, "Commissioning of the RPLIDAR A2 M8 Sensor using Linux and Python," *Pistas Educativas*, vol. 40, no. 130, pp. 1538–1546, 2018. [Online]. Available: <https://pistaseducativas.celaya.tecnm.mx/index.php/pistas/article/view/1638>
- [21] P. Biernacki and A. Ziebinski, "The method of reflection-based marker detection and identification to ensure accurate AGV docking," *Scientific Reports*, vol. 15, no. 1, Art. no. 41343, 2025. <https://doi.org/10.1038/s41598-025-25357-x>
- [22] W. Wang *et al.*, "Integrated navigation method for Orchard-Dosing robot based on LiDAR/IMU/GNSS," *Agronomy*, vol. 14, no. 11, Art. no. 2541, 2024. <https://doi.org/10.3390/agronomy14112541>
- [23] Bosch Sensortec GmbH, *BNO055 Intelligent 9-axis Absolute Orientation Sensor: Data Sheet*, document no. BST-BNO055-DS000-18, Rev. 1.8, October 2021. Available: <https://www.bosch-sensortec.com/media/boschsensortec/downloads/datasheets/bst-bno055-ds000.pdf>
- [24] P. Thavitchasri, D. Maneetham, and P. N. Crisnapati, "Intelligent surface recognition for autonomous tractors using ensemble learning with BNO055 IMU sensor data," *Agriculture*, vol. 14, no. 9, Art. no. 1557, 2024. <https://doi.org/10.3390/agriculture14091557>
- [25] M. A. B. Bhuiyan, H. M. Abdullah, S. E. Arman, S. Saminur Rahman, and K. Al Mahmud, "BananaSqueezeNet: A very fast, lightweight convolutional neural network for the diagnosis of three prominent banana leaf diseases," *Smart Agricultural Technology*, vol. 4, Art. no. 100214, 2023. <https://doi.org/10.1016/j.atech.2023.100214>
- [26] J. Ugarte Fajardo, M. Maridueña-Zavala, J. M. Cevallos-Cevallos, and D. Ochoa Donoso, "Effective methods based on distinct learning principles for the analysis of hyperspectral images to detect black Sigatoka disease," *Plants*, vol. 11, no. 19, Art. no. 2581, 2022. <https://doi.org/10.3390/plants11192581>
- [27] H.-T. Thai, K.-H. Le, and N. L.-T. Nguyen, "EF-CenterNet: An efficient anchor-free model for UAV-based banana leaf disease detection," *Computers and Electronics in Agriculture*, vol. 231, Art. no. 109927, 2025. <https://doi.org/10.1016/j.compag.2025.109927>
- [28] A. E. B. Velasquez *et al.*, "Reactive navigation system based on H_∞ control system and LiDAR readings on corn crops," *Precision Agric.*, vol. 21, no. 2, pp. 349–368, 2019. <https://doi.org/10.1007/s11119-019-09672-8>
- [29] H. B. Guerrero, J. J. López A., and A. L. Jutinico, "Autonomous mobile robot prototype for vegetative furrows following based on optimal control over ROS," *Int. J. Onl. Eng.*, vol. 22, no. 2, pp. 4–20, 2026. <https://doi.org/10.3991/ijoe.v22i02.58423>
- [30] H. B. Guerrero, "Development of a control system for a small-scale agricultural mobile robot for inter-row navigation," Ph.D. dissertation, School of Engineering of São Carlos, University of São Paulo, São Carlos, Brazil, 2016. <https://doi.org/10.11606/T.18.2017.tde-24122016-154205>
- [31] H. B. Guerrero, D. C. Montenegro, and A. L. Jutinico, "Reactive fuzzy control and field validation of a low-cost autonomous tracked robot for plantain crop row following," *Eng. Res. Express*, vol. 8, Art. no. 045227, 2026. <https://doi.org/10.1088/2631-8695/ae458e>
- [32] Q. Qiu *et al.*, "Extended Ackerman steering principle for the coordinated movement control of a four-wheel drive agricultural mobile robot," *Computers and Electronics in Agriculture*, vol. 152, pp. 40–50, 2018. <https://doi.org/10.1016/j.compag.2018.06.036>

- [33] T. D. Gillespie, *Fundamentals of Vehicle Dynamics*. Warrendale, PA, USA: Society of Automotive Engineers, 1992. <https://doi.org/10.4271/R-114>
- [34] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer U, 2012. <https://doi.org/10.1007/978-1-4614-1433-9>
- [35] I. D. Duarte Brito, M. A. Roldan Benavides, L. F. Lindarte Muñoz, A. L. Jutinico Alarcón, and H. B. Guerrero, “Small-scale car-like agricultural mobile robot for autonomous navigation in crop rows: A first ROS2 approach,” in *Proc. 19th Int. Congr. Electron., Control Telecommun. (CIECT XIX)*, Bogotá D.C., Colombia, Nov. 21–22, 2024, vol. 15, pp. 745–769.
- [36] S. Macenski, A. Soragna, M. Carroll, and Z. Ge, “Impact of ROS 2 node composition in robotic systems,” *IEEE Robot. Autom. Lett.*, vol. 8, no. 7, pp. 3996–4003, 2023. <https://doi.org/10.1109/LRA.2023.3279614>
- [37] A. Bonci, F. Gaudeni, M. C. Giannini, and S. Longhi, “Robot operating system 2 (ROS 2)-based frameworks for increasing robot autonomy: A survey,” *Applied Sciences*, vol. 13, no. 23, Art. no. 12796, 2023. <https://doi.org/10.3390/app132312796>

8 AUTHORS

Danna C. Montenegro is an Electronic Engineering student at Universidad Distrital Francisco José de Caldas, Bogotá, Colombia. She holds a Technologist degree in Mechatronics from the Servicio Nacional de Aprendizaje (SENA) and leads the SEROB Student Research Group, affiliated with the LASER Research Group at Universidad Distrital Francisco José de Caldas (E-mail: dcmontenegroo@udistrital.edu.co).

Iván D. Duarte received his degree in Electronic Engineering from Universidad de los Llanos, Colombia, and his master's degree in Radar Technologies, Equipment, and Systems Development from Universidad Politécnica de Madrid, Spain. He has over 10 years of experience in radar systems, hardware design, firmware development, embedded software, software architecture, and mission-critical system integration. His work includes electronic design, C/C++ embedded development, communications, Linux server administration, backend development with Java and Python, and projects related to radar, defense systems, UAV technologies, digital identity, biometric verification, and specialized technology platforms (E-mail: ivan.duarte.brito@unillanos.edu.co).

Andrés L. Jutinico received his degree in electronic engineering from Universidad Distrital Francisco José de Caldas, Bogotá, Colombia, in 2005; an M.Sc. degree in industrial automation from Universidad Nacional de Colombia in 2012; and a Ph.D. in mechanical engineering from the University of São Paulo, Brazil, in 2019. He is Director of the LASER Research Group at Universidad Distrital Francisco José de Caldas (E-mail: aljutinicoa@udistrital.edu.co).

Henry B. Guerrero received his degree in electronic engineering from Universidad de los Llanos, Villavicencio, Colombia, in 2004; a specialist degree in automatic and industrial informatics from Universidad Autónoma de Colombia in 2007; and a Ph.D. in mechanical engineering from the University of São Paulo, Brazil, in 2016. He is currently a lecturer at Universidad Distrital Francisco José de Caldas and a postdoctoral researcher at BITPOINTER S.A.S., within a project supported by the Ministry of Science, Technology, and Innovation of Colombia (MINCIENCIAS) and the Francisco José de Caldas Fund, under Call 934 of 2023, contract 332-2023, project 99209 (E-mail: hbguerreiro@ieee.org).