Preprints of the 23rd IFAC World Congress
Busan, Republic of Korea, August 23-28, 2026

Crop-Row Following in Grass Fields using Deep Learning Detection, Sliding Mode Control and Fuzzy Velocity Modulation ⋆

Igor Ferreira da Costa * **Erik Lykke Trier** *
Antonio Candea Leite *

* *Norwegian University of Life Sciences*
Faculty of Science and Technology, Droebakveien 31, 1430 Aas, Norway
(E-mail: igfe@nmbu.no, antonile@nmbu.no)

Abstract: Improving the nitrogen use efficiency (NUE) of perennial ryegrass is an important sustainability goal for livestock feed production. While autonomous ground robots can facilitate phenotyping in large field trials, navigation in unstructured grass fields remains a challenge for standard vision-based systems. This work presents an integrated navigation pipeline featuring an improved deep-learning line detection model and a robust image-based visual servoing (rIBVS). A new line modeling approach and a fuzzy-based velocity modulation module that adjusts forward speed to prevent line loss are proposed. The system was evaluated using a custom simulator and a Gazebo 3D environment built from real-world UAV field scans. The results demonstrate that fuzzy modulation reduces overshoot and prevents navigation failure during in-row transitions. The velocity modulation enabled a 40% increase in maximum allowable operational speed while maintaining real-time CPU performance (10 Hz) on a low-power CPU.

Keywords: Agricultural robotics; Autonomous vehicles; Fuzzy systems; Robust control; Robot perception and sensing.

1. INTRODUCTION

In temperate climate agricultural systems, perennial ryegrass (*Lolium perenne*) is one of the most important forage grasses due to its high-quality dry matter for livestock production. However, high levels of nitrogen fertilizer are recommended to maintain high productivity, which can lead to increased costs and greater environmental losses. Therefore, improving nitrogen use efficiency (NUE) is essential. Traditional plant phenotyping is a repetitive and laborious task in which similar measurements must be conducted on several plant targets throughout the experiment. In such scenarios, where consistency is primary, robotic automated image-based phenotyping is a promising alternative. By automating the process in a controlled environment, rich datasets can be built in large trials while monitoring phenotypic traits non-destructively.

To deploy autonomous robotic systems, robust navigation must overcome the unstructured nature of outdoor field conditions, including changing canopy geometry, uneven ground, and varying luminosity levels. Real-Time Kinematic Global Navigation Satellite Systems (RTK-GNSS) are often the preferred choice for high-precision global localization. However, they represent a high-cost solution dependent on external communication links, where outages and dropouts can occur. Consequently, the use of on-board cameras for navigation is one of the available alternatives. By exploiting either the visible path between crop-rows or the crop-row itself, as performed by Nkwocha and Wang (2025); Ponnambalam et al. (2020); Ahmadi et al. (2020), a robot can track and follow the crop-row reference while relying on internal sensing and processing.

Several approaches have been studied for the visual guidance of mobile robots in crop fields and similar unstructured environments. Cherubini et al. (2008) introduced an image-based visual servoing (IBVS) framework relying on two primitives (row and column) selected depending on the current feature position in the image. A proportional controller was used, requiring a low constant forward velocity for stability. Originally tested with an urban electric vehicle, this method was adapted for agricultural crop-row following. Barbosa et al. (2021) extended the IBVS formulation with a sliding-mode controller, increasing robustness to model uncertainties and uneven terrain. Despite the improved control layer, the underlying feature extraction pipeline remained limited.

Ahmadi et al. (2020) improved the detection with color-based segmentation for mask creation, fitting a line to the mask and applying the IBVS row primitive for crop-row following. Meanwhile, da Costa et al. (2025) proposed a Deep-Learning end-to-end line detection system, enabling the robot to infer the line to be followed directly from the crop-row image. This work also relied only on the IBVS row primitive, assuming low driving speeds and being limited to in-row scenarios. Xaud et al. (2025) leveraged thermal imaging to detect canopy temperature patterns from which navigable paths could be extracted, coupling pose-based visual servoing and sliding-mode control to operate in visually degraded settings.

⋆ This work was supported by the Research Council of Norway (RCN), through the DLT-Farming project, grant number 344288.

©2026 the authors. Accepted by IFAC for publication under a Creative Commons Licence CC-BY-NC-ND

In the work of de Silva et al. (2024), a U-Net deep-learning architecture was leveraged to produce masks, from which row lines were extracted using a triangular scanning algorithm. A simpler proportional controller was applied under the assumption of a direct correspondence between image features and pose errors. The IBVS approach was also explored by de Silva et al. (2023), who noted that more sophisticated control strategies were required for deployment in challenging field conditions.

Alternative sensing and control paradigms have also been explored. Affonso et al. (2025) used deep learning to interpret LiDAR scans and infer crop-row structure, combined with Model Predictive Control and a synthetic dataset generation technique. Other authors have focused on different control approaches, such as Bach and Yi (2022), who proposed a Fuzzy-PID controller capable of adapting robot speed during curved paths. In the work of Oliveira et al. (2019), the authors explored a fuzzy controller for navigation with positive results.

The aforementioned studies highlight classical IBVS approaches, along with their challenges and limitations. Gaps remain regarding robustness under complex field conditions, initial line reaching, performance, and combined testing. Prior works that utilized advanced control techniques often relied on simple or limited feature extraction, while advanced perception methods often lack advanced controllers, limiting performance in challenging field conditions. In the present work, we deploy an improved line detection mechanism based on the work of da Costa et al. (2025), the robust controller approach proposed by Barbosa et al. (2021), with a fuzzy-based velocity modulation module to dynamically adjust forward speed.

2. METHODOLOGY

A mobile robot line following system can be split into two main components: the line detection module and the controller module. Such systems extract trajectories from observable environmental features as color, texture, or geometric structure to guide the robot on the path defined by the detected line. Agricultural fields are often organized into parallel crop-rows separated by navigable inter-row spaces. Crop-row following takes advantage of this structure by detecting the crop-rows or the inter-row spaces, thereby keeping the robot within a desired operational corridor.

In order to follow the row, the detection module must estimate the crop-row line $\mathcal{L}(P, \Theta)$ in the image, retrieving both its orientation $\Theta \in (-\pi, \pi)$ and position $P(X, Y)$. Here Θ denotes the angle of the detected line relative to the negative y_v-axis, and $X, Y \in [-1, 1]$ represent the normalized image coordinates of its position. This parameterization is illustrated in Fig. 1a. Afterwards, the controller module is designed to compute the necessary angular velocity given the detected line $\mathcal{L}$, the camera tilt offset $\varphi \in \left(0, \frac{\pi}{2}\right]$ relative to the horizontal plane, and the camera coordinates (t_x, t_y, t_z) with respect to the robot frame, as described in Fig. 1b.

The control module was originally designed considering: the camera is coarsely extrinsically calibrated within the system; the mobile robot maintains a constant, positive desired velocity; and the crop-row line remains within the

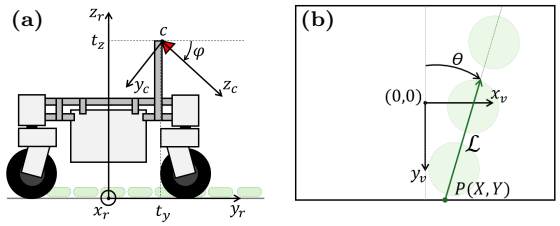


Fig. 1. Robot system overview for the row following task. (a) Camera placement on the robot; (b) Robot camera view, displaying the desired image features.

camera's field of view throughout the task. By introducing the fuzzy linear-velocity controller, we aim to allow the robot to modulate its linear speed within $v \in (0, v_{\max}]$. The fuzzy controller will reduce the robot linear speed when the error increases, achieving a practical maximum safe velocity. In addition, the line-detection module was improved to provide lines more compatible with the column primitive.

2.1 Detection Module

The detection module was originally developed around a DeepLabV3+ deep learning segmentation model. Its encoder-decoder structure allowed the use of segmentation masks to train the encoder while also training prediction heads that inferred the start and finish endpoints of a line in the image. Despite the auxiliary task approach lending the best results, the direct training approach also shown adequate results (da Costa et al., 2025). By removing the auxiliary task and not using all layers of the ResNet50 backbone, the model becomes lightweight, allowing faster training process and CPU real time inference. The simplified model diagram for line detection is shown in Fig. 2.

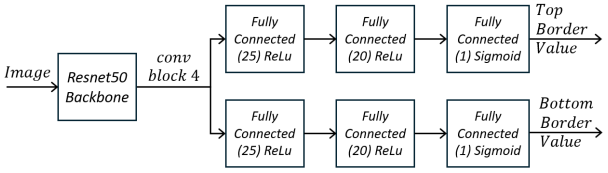


Fig. 2. Line detection model block diagram.

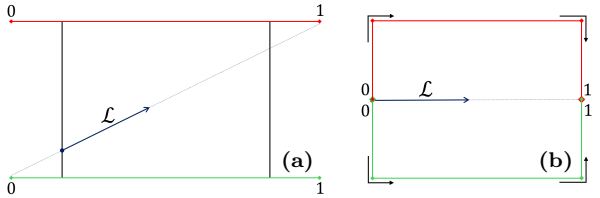


Fig. 3. Border value parametrization: (a) Old parametrization; (b) New parametrization. Note that the new parametrization allows for a broader range of lines.

The original line prediction module assumed that the line would start at the top row and end at the bottom row of the image, defining a straight segment that connects the points $A = (X_0, 0)$ and $B = (X_1, H)$, as seen in Fig. 3a. In the new border value parametrization, given H and W as the image height and width, each border value starts at the left border middle point $(0, \frac{H}{2})$ and finishes at the right border middle point $(W, \frac{H}{2})$. The top border value can be split into three sections: origin to top-left image corner $(0, 0)$, top-left to top-right image corner $(W, 0)$,

and top-right image corner to the end point. The bottom border value is described similarly, passing through the bottom image border via points $(0, \frac{H}{2})$, $(0, H)$, (W, H), and $(W, \frac{H}{2})$. In a 256×256 image, the border value results in a range of $[0, 512]$, decomposed into $128 + 256 + 128$ and normalized into the $[0, 1]$ interval.

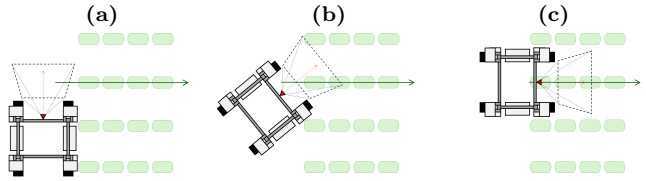


Fig. 4. Worst case scenario for crop-row approach, step by step: (a) Perpendicular, outside of the row; (b) Line reaching maneuver; (c) Alignment complete.

The dataset used in this work consists of images captured both in the real field and in a Gazebo-simulated environment, comprising a total of 645 from the real field and 1332 from simulation. All images were manually annotated with the expected line to be followed by the robot using a custom labeling tool. A representative sample of the annotated dataset can be viewed in Fig. 5.

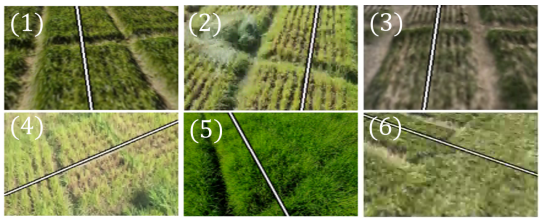


Fig. 5. Dataset sample with annotated reference lines.

The dataset is naturally imbalanced, as instances of the column primitive (i.e., row entrances) are less frequent. This imbalance is visible in Fig. 6, which illustrates the distribution of border values: the first and last two histogram bins contain significantly fewer samples, as they correspond to the column-primitive cases.

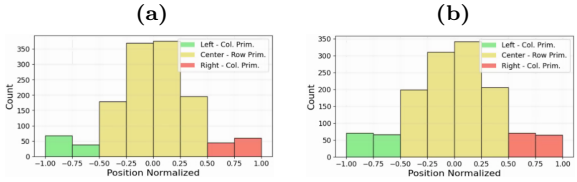


Fig. 6. Border value dataset distribution: (a) Upper half occurrences; (b) Bottom half occurrences.

Images were resized to 256×256, providing an effective trade-off between performance and computational speed (da Costa et al., 2025), and normalized to the $[-1, 1]$ range. A 70/20/10 split was used for training, validation, and testing. Training data were augmented with additive noise and random brightness or contrast shifts of up to 20%. The model, implemented in Keras/TensorFlow 2.12 with pre-trained ResNet50 weights, was trained on an NVIDIA RTX 3050 Laptop GPU. Optimization used Mean Squared Error and the Adam optimizer with a 10^{-5} learning rate. Using a batch size of 12, training was tested from 40 to 100 epochs in increments of 10; 50 epochs was selected as the optimal point based on validation loss and test performance.

2.2 Robust IBVS (rIBVS) Controller

The unicycle robot is a common abstraction used to model many mobile robotic systems. Its configuration is represented as $q = [x,\, y,\, \theta]^T \in \mathbb{R}^3$, where $x, y \in \mathbb{R}$ denote the Cartesian coordinates of the robot frame, and $\theta \in \mathbb{R}$ describes the robot's orientation with respect to the world frame. The nonholonomic constraint arises because the wheel cannot generate lateral motion, which enforces that the velocity component orthogonal to the wheel plane must be zero. This restriction leads to the pure rolling constraint for the wheel given by $\dot{x} \sin\theta - \dot{y} \cos\theta = 0$ which defines the set of admissible motions (Siciliano et al., 2009). The kinematic model of the unicycle robot can be written as

$$\dot{q} = G(q)\, u \quad \Rightarrow \quad \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \qquad (1)$$

where $G(q) \in \mathbb{R}^{3\times 2}$ is the so-called input mapping matrix, $u = [v,\, w]^T \in \mathbb{R}^2$ is the control input, and $v \in \mathbb{R}$ and $\omega \in \mathbb{R}$ denote the driving and steering velocities, respectively. The same structure describes a differential drive mobile robot provided that v and ω are expressed as functions of the angular speeds of the left and right wheels, given respectively by $\dot{\phi}_L \in \mathbb{R}$ and $\dot{\phi}_R \in \mathbb{R}$, which represent the actual control inputs. The robot velocity $V_r = [v_r,\, \omega_r]^T \in \mathbb{R}^6$ is related to the control input $u = [v,\, \omega]^T \in \mathbb{R}^2$ through the axis-selection matrix $S \in \mathbb{R}^{6\times 2}$ as $V_r = S\, u$ such that $V_r = [v, 0, 0, 0, 0, \omega]^T$.

Consider a pin-hole camera that observes a line $\mathcal{L}$ with a set of image features $s = [X,\, Y,\, \Theta]^T \in \mathbb{R}^3$, as described in Section 2. The camera velocity V_c is related to the robot velocity V_r by using the adjoint matrix $A_{cr}(t_{cr}, R_{cr}) \in \mathbb{R}^{6\times 6}$ as $V_c = A_{cr}\, V_r$, where $t_{cr} \in \mathbb{R}^3$ and $R_{cr} \in \mathbb{SO}(3)$ describe the robot frame's translation and orientation with respect to the camera frame respectively. Here, without loss of generality, we assume that $t_{cr} = [0, t_y, t_z]^T$ and $R_{cr} = R_x(\pi/2 + \varphi)$. Now, we can express the relationship between the camera velocity V_c and the control input u as follows: $V_c = A_{cr} S u = T_{cr} u$ where $T_{cr} = [\, T_v \;\; T_w \,] \in \mathbb{R}^{6\times 2}$ is a block matrix, and $T_v \in \mathbb{R}^6$ and $T_\omega \in \mathbb{R}^6$ are the columns of T_{cr} associated with v and ω, respectively.

The velocity of the image features $\dot{s} = [\dot{X},\, \dot{Y},\, \dot{\Theta}]^T \in \mathbb{R}^3$ is related to the camera velocity $V_c = [v_c,\, \omega_c]^T \in \mathbb{R}^6$ by

$$\dot{s} = L_s\, V_c + \xi_s = L_s\, T_{cr}\, u + \xi_s\,, \qquad (2)$$

where $L_s = [L_x \;\; L_y \;\; L_\theta]^T \in \mathbb{R}^{3\times 6}$ is the interaction matrix and $\xi_s = [\xi_x,\, \xi_y,\, \xi_\theta]^T$ is a bounded disturbance term that denotes the uncertain motion of the image feature. For more details about the elements of the L_s and T_{cr} matrices, please refer to Cherubini et al. (2008); Barbosa et al. (2021). The rIBVS controller is specifically designed to ensure robustness in the presence of uncertainties within these matrices, as well as disturbances in the motion of the image feature.

In the IBVS control design, two control primitives (row and column) are defined by selecting appropriate rows and columns of the interaction matrix L_s. For the row primitive, it is assumed that $Y = Y^*$ remains constant, whereas for the column primitive, $X = X^*$ is taken to be constant:

$$\begin{cases} \dot{Y}=0 \;\Rightarrow \dot{s}_r = \begin{bmatrix} \dot{X} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} L_x \\ L_\theta \end{bmatrix} \begin{bmatrix} T_v & T_\omega \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} + \xi_{s_r}\,, \\ \dot{X}=0 \;\Rightarrow \dot{s}_c = \begin{bmatrix} \dot{Y} \\ \dot{\Theta} \end{bmatrix} = \begin{bmatrix} L_y \\ L_\theta \end{bmatrix} \begin{bmatrix} T_v & T_\omega \end{bmatrix} \begin{bmatrix} v_i \\ \omega_i \end{bmatrix} + \xi_{s_c}\,. \end{cases} \tag{3}$$

Both control primitives can be written in a compact form as $\dot{s}_i = A_i\, v_i + B_i\, \omega_i + \xi_{s_i}$ with $i \in \{r, c\}$. Considering the regulation errors as

$$e_r = \begin{bmatrix} e_x \\ e_\theta \end{bmatrix} = \begin{bmatrix} X - X^* \\ \Theta - \Theta^* \end{bmatrix}\,, \quad e_c = \begin{bmatrix} e_y \\ e_\theta \end{bmatrix} = \begin{bmatrix} Y - Y^* \\ \Theta - \Theta^* \end{bmatrix}\,, \tag{4}$$

and defining

$$A_{r,c} = \begin{bmatrix} L_{x,y} \\ L_\theta \end{bmatrix} T_v\,, \quad B_{r,c} = \begin{bmatrix} L_{x,y} \\ L_\theta \end{bmatrix} T_\omega\,, \quad \xi_{s_{r,c}} = \begin{bmatrix} \xi_{x,y} \\ \xi_\theta \end{bmatrix}\,,$$

the IBVS system for the crop-row following problem can be controlled according to:

$$\begin{bmatrix} v_i \\ \omega_i \end{bmatrix} = \begin{bmatrix} v_d \\ -B_i^\dagger (\Lambda_i\, e_i + A_i\, v_d + \xi_{s_i}) \end{bmatrix}\,, \tag{5}$$

where $\Lambda_i \in \mathbb{R}^{2\times 2}$ is a positive definite gain matrix. The control design in Eq. (5) assumes perfect IBVS system calibration and full knowledge of the disturbance term ξ_{s_i}. It also considers the driving velocity v constant, whereas only the steering velocity ω_i is actively controlled.

Accounting for model uncertainties and bounded disturbances, the robust formulation can be written by expressing the angular velocity ω_i as:

$$\omega_i = -\hat{B}_i^\ddagger\, (\nu_i + \hat{A}_i\, v_d)\,, \tag{6}$$

with $\nu_i \in \mathbb{R}^2$ being the robust control action to be designed. Then the following perturbed IBVS system can be written as $\dot{s} = \nu_i + \eta_i$ where the disturbance term η_i is given by

$$\eta_i = \Delta A_i v_d + \Delta B_i \omega_i + \xi_{s_i}\,, \tag{7}$$

where $\Delta A_i = (A_i - \hat{A}_i)$ and $\Delta B_i = (B_i - \hat{B}_i)$ are the camera-robot calibration errors.

The rIBVS approach can be implemented as follows:

$$\nu_i = \Lambda_i\, e_i + \tau_i\,, \tag{8}$$

where $\tau_i \in \mathbb{R}^2$ is the robustness term given by the super-twisting algorithm:

$$\tau_i = K_i \lceil \sigma_i \rfloor^\gamma + \zeta_i\,, \quad \dot{\zeta}_i = -\mu_i\, \mathrm{sgn}(\sigma_i)\,, \tag{9}$$

where $\sigma_i = e_i + \int_0^t e_i(t')dt'$ is the sliding surface, $K_i \in \mathbb{R}^{2\times 2}$ and $\mu_i \in \mathbb{R}^{2\times 2}$ are positive definite gain matrices, and $\lceil \sigma_i \rfloor^\gamma = |\cdot|^\gamma \mathrm{sgn}(\cdot)$ for $0 < \gamma \leq 1$, where $\mathrm{sgn}(\cdot)$ is the sign function. The stability analysis for the robust IBVS system can be found in Barbosa et al. (2021), whereas Cherubini et al. (2008) presents the stability analysis for the original IBVS controller.

2.3 Fuzzy Velocity Modulation Controller

The motion of a mobile robot is naturally limited by motor capabilities or traction. Human operators naturally compensate for these constraints by reducing speed when tighter turns are required. For replicating this approach, each primitive receives its own set of fuzzy inputs, outputs, and linguistic rules. A Mamdani inference system was selected for its interpretability, with Centroid defuzzification to ensure smooth transitions in the commanded speed. The proposed fuzzy module attenuates the desired linear velocity before the computation of the desired angular velocity, as illustrated in Fig. 7.

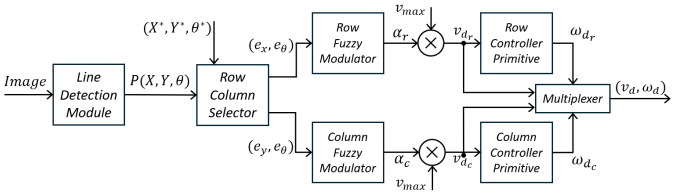


Fig. 7. System block diagram comprising inputs, detection, fuzzy modulation, and control primitives.

The fuzzy inputs are the same normalized error variables used in the IBVS controller, as defined in Eq. (4). The linguistic variables are defined as follows: e_θ uses the labels Negative (N), Zero (Z), and Positive (P); e_y uses Small (S), Medium (M), and Large (L); and e_x uses Negative Big (NB), Negative Small (NS), Zero (Z), Positive Small (PS), and Positive Big (PB). The output linear-velocity modulation variable is described by three fuzzy sets: Low (L), Medium (M), and High (H). All membership functions are triangular, with 50% overlap, defined over the same universes of discourse used by the IBVS controller. Their ranges and triangular shapes are illustrated in Fig. 8.

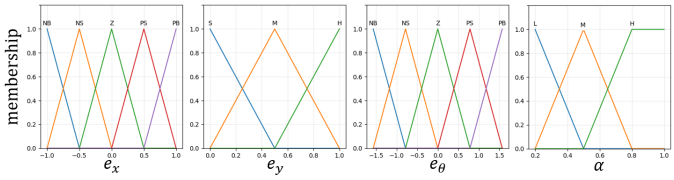


Fig. 8. Fuzzy membership functions for the errors e_x, e_y, e_θ, and the output α.

The rules detailed in Table 1 are designed to mitigate risks by reducing the robot velocity. During the column primitive actuation, high caution is required to prevent overshooting or losing the line. Consequently, the velocity is intentionally capped at "Medium", and large column errors or angular errors force the velocity to "Low". During the row primitive actuation, "High" velocity is permitted only when the robot is aligned ($e_x \approx 0, e_\theta \approx 0$). As misalignment occurs, speed is reduced to combat error growth and maintain the crop line alignment.

Table 1. Velocity modulation rule base.

e_θ	**Column Primitive** (Input e_y)			**Row Primitive** (Input e_x)				
	S	**M**	**L**	**NB**	**NS**	**Z**	**PS**	**PB**
N	M	L	L	L	L	M	L	L
Z	M	M	L	L	M	H	M	L
P	M	L	L	L	L	M	L	L

The fuzzy module is computationally lightweight, involving only triangular membership evaluations and a small rule set, introducing a negligible processing load relative to the line detection module. The fuzzy modulation is evaluated at every control cycle and applied directly to the robust IBVS controller. As the fuzzy output is bounded $[0.2, 1.0]$, the fuzzy system does not affect the sign of the control signal or insert new discontinuities into the system, actuating as a damper by reducing robot velocity.

3. RESULTS AND DISCUSSION

The line detection module took less than ten minutes to train, utilizing half-precision to improve speed and memory usage. In the training GPU, each inference takes $5\,ms$, while in the robot CPU, an Intel Core i5-8259U, about $55\,ms$. Evaluating the 225 images of the test group, an average error of 6% for each border value is found, which leads to an average error of 8 degrees for θ. Selecting only images for the column primitive, the average border value error is 14%, which resolves to an average error of 20 degrees for θ. For these images, the model presented a conservative tendency, predicting lines with smaller angular values, which can be explained by the smaller number of similar images available in the dataset.

Initially, the impact of the Fuzzy Velocity Module on the system control performance is evaluated through a custom deterministic simulation of a simplified robot model. A high desired velocity of $1.5\,ms^{-1}$ is imposed on the system under three starting conditions. In Fig. 9, an initial position with large e_x and e_θ errors is configured. In Scenario (i), the image features are fully contained within the row primitive. The controller is able to correct the robot position even without fuzzy modulation. However, improvements are observed with its activation. Specifically, the fuzzy modulation results in a reduced maximum overshoot and a shorter settling time, reaching steady earlier. Without it, e_x surpasses 0.9, which could lead to the loss of the crop-row if 1.0 is reached with high e_θ.

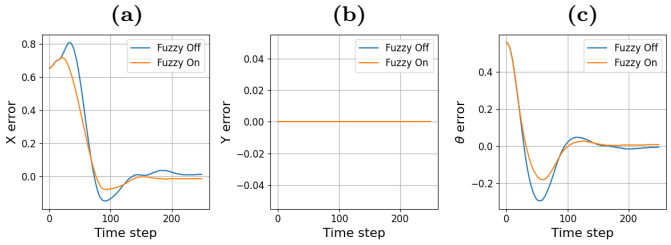


Fig. 9. Error variables comparison for the row primitive start position for Scenario (i).

In Scenario (ii), an initial position is defined so that both primitives are required. The system is able to complete the maneuver without fuzzy modulation; however, Fig. 10 demonstrates that the fuzzy module provides a superior damping effect. Specifically, a reduced peak overshoot and attenuated oscillations are observed with fuzzy modulation, as the lower velocity prevents the system from overcorrecting during primitive transitions. The objective of these first two scenarios is to display ideal conditions in which the previous system is still functional while highlighting that the fuzzy module improves performance. By initiating the simulation with an unfavorable heading position in Scenario (iii), a system failure is triggered without fuzzy modulation, as shown in Fig. 11. In this case, the robot fails to turn in time, causing the crop-row to exit the camera's field of view.

Fig. 12 shows the linear speed for all three scenarios. The observed overshoot in the error graph triggered a reduction in speed, as expected. Scenarios (ii) and (iii) show some brief discontinuities, present due to the primitives switching. This same discontinuity can be seen for the angular

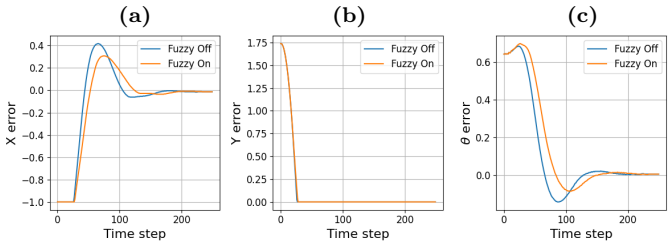


Fig. 10. Error variables comparison for the first column primitive start position for Scenario (ii).

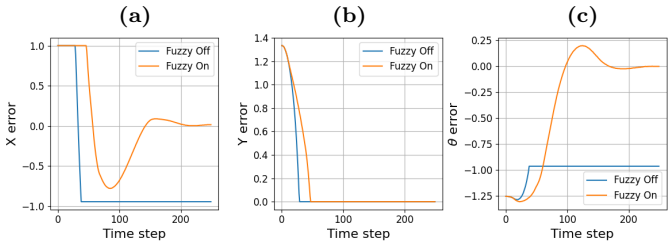


Fig. 11. Error variables comparison for the second column primitive start position for Scenario (iii).

speed in Fig. 13. They are present even without the fuzzy modulation, being a byproduct of the controller switching approach, and it can be alleviated with further control gain tuning. Scenario (iii) fails without fuzzy modulation due to the crop-row going outside of the camera view, as the robot cannot turn in time.

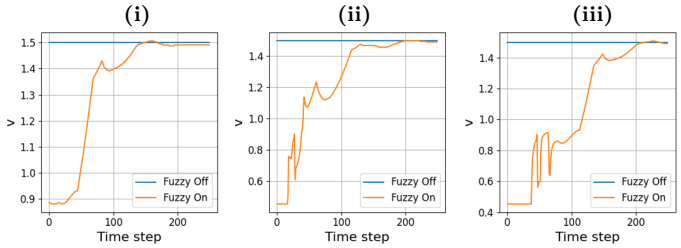


Fig. 12. Linear velocities for each test scenarios.

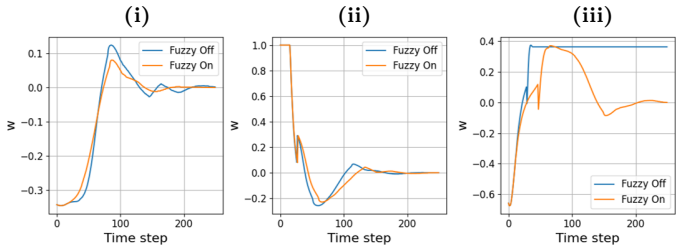


Fig. 13. Angular velocities for each test scenarios.

The final test is executed in a Gazebo simulation environment created using the OpenDroneMaps software to construct a 3D model from the real world field from a UAV survey. Fig. 14 shows the simulation environment with a grass field scenario and a differential-drive mobile robot.

From the position shown in Fig. 14, the robot ran with three velocities. The first and second runs have the maximum velocity set to $0.65\,ms^{-1}$, with and without the fuzzy modulation. This value was chosen empirically as the maximum velocity in which the robot was able to perform the entry maneuver without the fuzzy module. The third run have its maximum velocity set to $1.0\,ms^{-1}$, which was selected to match the non fuzzy modulated performance.

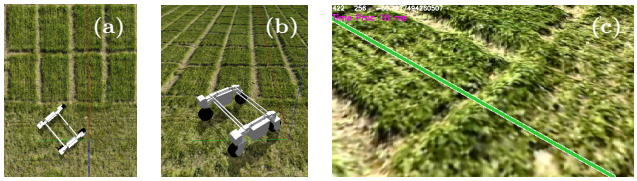


Fig. 14. Gazebo environment: (a) Top view; (b) Robot and field view; (c) Robot camera view with predicted line.

As observed in Fig. 15a, run two (orange) outperformed the baseline, run one (blue), by executing a trajectory with a smaller overshoot. Fig. 15b shows that the robot heading converged with a smaller overshoot than run one (blue). Eventual heading deviance can occur since the simulated field is uneven. This can be seen in run two (orange) in Fig. 15b, but the robot is able to recover.

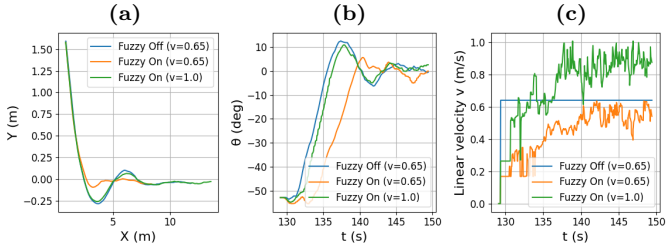


Fig. 15. Robot readings for the three tests in the Gazebo simulation: (a) Robot trajectory; (b) Robot reading; (c) Robot linear velocity.

Fig. 15c shows the desired velocity set for the robot during the maneuver. With the fuzzy modulation, the robot velocity starts constrained, increasing when the error is small, as defined in the fuzzy rules. Since the Deep learning line detection has an expected error, this causes the image feature errors to be noisy. As such, the robot linear velocity to became noisier, similar to the angular velocity, due to the reliance on the deep learning predicted line. In the third run (green), the maximum velocity is increased until the trajectory performance is similar to run one (blue), where the robot is now able to reach higher speeds. The highest successful maximum speed observed for this test scenario was $1.1\,ms^{-1}$, and the maximum allowable speed by the simulated Thorvald robot is $1.5\,ms^{-1}$.

4. CONCLUSION

In this work, a camera-based crop-row following scheme was presented and evaluated, successfully integrating a deep-learning-based line detection mechanism with a robust Image-Based Visual Servoing (IBVS) control architecture. The proposed fuzzy velocity modulation enhanced the robot capabilities, either by improving the traversal trajectory through reduced transient error or by enabling higher operational speeds. The pipeline demonstrated real-time feasibility, executing at 10 Hz on a low-power mobile CPU. Nevertheless, a more robust computational platform is recommended to support higher-frequency control loops and more complex model architectures. Future research should focus on enhancing the detection model's performance, specifically exploring the application of modern deep-learning structures such as Vision Transformers (ViT) for improved feature extraction. Furthermore, the integration of vision-based line following with global localization systems should be investigated; such sensor fusion could provide necessary redundancy and improve line prediction reliability. Crucially, the system must now be validated in real-world field environments. Factors such as dynamic lighting variations, seasonal changes in crop morphology, and robot motion disturbances may induce transient errors in the detection module. Consequently, advanced filtering techniques should be evaluated to identify the optimal trade-off between induced control delay and the requisite noise rejection. By overcoming these fundamental navigational challenges, autonomous mobile robotic platforms can better facilitate agricultural research, work, and production.

REFERENCES

Affonso, F., Tommaselli, F.A.G., Capezzuto, G., Gasparino, M.V., Chowdhary, G., and Becker, M. (2025). CROW: A self-supervised crop row navigation algorithm for agricultural fields. *J. Intell. Robot. Syst.*, 111(1), 28.

Ahmadi, A., Nardi, L., Chebrolu, N., and Stachniss, C. (2020). Visual servoing-based navigation for monitoring row-crop fields. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 4920–4926.

Bach, S.H. and Yi, S.Y. (2022). An efficient approach for line-following automated guided vehicles based on fuzzy inference mechanism. *J. Robot. Control*, 3(4), 395–401.

Barbosa, G.B.P., da Silva, E.C., and Leite, A.C. (2021). Vision-based autonomous crop row navigation for wheeled mobile robots using super-twisting sliding mode control. In *Eur. Conf. Mob. Robots (ECMR)*, 1–6.

Cherubini, A., Chaumette, F., and Oriolo, G. (2008). An image-based visual servoing scheme for following paths with nonholonomic mobile robots. In *10th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, 108–113.

da Costa, I.F., Leite, A.C., and Caarls, W. (2025). Data set diversity in crop row detection based on CNN models for autonomous robot navigation. *J. Field Robot.*, 42(2), 525–538.

de Silva, R., Cielniak, G., and Gao, J. (2024). Vision based crop row navigation under varying field conditions in arable fields. *Comput. Electron. Agric.*, 217, 108581.

de Silva, R., Cielniak, G., Wang, G., and Gao, J. (2023). Deep learning-based crop row detection for infield navigation of agri-robots. *J. Field Robot.*, 41(7), 2299–2321.

Nkwocha, C.L. and Wang, N. (2025). Deep learning-based semantic segmentation with novel navigation line extraction for autonomous agricultural robots. *Discov. Artif. Intell.*, 5(1), 73.

Oliveira, A.I.S., Carvalho, T.M., Martins, F.F., Leite, A.C., Figueiredo, K.T., Vellasco, M.M.B.R., and Caarls, W. (2019). On the intelligent control design of an agricultural mobile robot for cotton crop monitoring. In *12th Int. Conf. Dev. eSyst. Eng. (DeSE)*, 563–568.

Ponnambalam, V.R., Bakken, M., Moore, R.J.D., Gjevestad, J.G.O., and From, P.J. (2020). Autonomous crop row guidance using adaptive multi-ROI in strawberry fields. *Sensors*, 20(18), 5249.

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer-Verlag.

Xaud, M.F.S., From, P.J., and Leite, A.C. (2025). Robust visual servoing and CNN-based thermal imaging for sugarcane row following with a skid-steering mobile robot. *IEEE Access*, 13, 143166–143195.