Journal of Intelligent & Robotic Systems (2025) 111:28
https://doi.org/10.1007/s10846-025-02219-2

REGULAR PAPER

CROW: A Self-Supervised Crop Row Navigation Algorithm for Agricultural Fields

Francisco Affonso[1] · **Felipe Andrade G. Tommaselli**[1] · **Gianluca Capezzuto**[1] · **Mateus V. Gasparino**[2] · **Girish Chowdhary**[2] · **Marcelo Becker**[1]

Received: 30 April 2024 / Accepted: 28 December 2024

Abstract
Compact robots operating beneath the crop canopy present great potential for a range of autonomous and remote tasks, including phenotyping, soil analysis, and cover cropping. Under-canopy navigation presents unique challenges, such as the need for a navigation system that can traverse diverse crop types, navigate despite sensory obstructions, and manage sensory noise effectively. Aiming to solve this problem in a scalable manner, we present a novel navigation method that uses a self-supervised neural network tailored for row-following in under-canopy plantations for mobile robots. Our method, termed CROW (**C**rop-**ROW** navigation), integrates perception, waypoint generator, and control components, and is capable of handling variations in luminosity, topology, types of plantations, and plant growth stages. By using a Deep Learning-based approach to interpret LiDAR scans, we convert the detected rows of crops into lines, establishing waypoints for the controller based on fundamental geometric principles. To address the computational complexity inherent in standard Model Predictive Controller solvers, we employ a Constrained Iterative Linear Quadratic approach. Our system has been validated in both simulated and real-world environments, demonstrating successful navigation through 115-meter corn rows with little to no intervention, *i.e.*, requiring only 3 ± 3 interventions per row experiment.

Keywords Mobile robotics · Navigation · Deep Learning · Optimal Control

1 Introduction

Autonomous farm vehicles have immense importance in increasing farm productivity through either automation of the existing machinery or the introduction of brand-new autonomous robotic systems. In this background, accurate navigation along the crop rows is a fundamental skill requisite for many tasks in farming, such as precision agriculture, planting, and harvesting of the crop rows. Along with the specific challenges that this kind of task represents, the deployment of robots in agricultural environments implies extra difficulty, due to the seasonality of crops, changes in land topography, and most importantly, the fact that more than one type of crop is interacted with.

It becomes logical to apply robots performing hazardous operations in challenging outdoor conditions - such as, for instance, farming or exploration of wild remote areas. That explains why the increasing use of robots in such kinds of operations is something worth noticing [1–3]. However, to ensure safety in tough terrains, say in large-scale commodity row crops, robots have to be deployed that can detect the paths that can be covered. In other words, small robots may ensure that the effective performance of various tasks in such situations is secured. Larger equipment is not applicable in these assignments, since they are designed either for open-field or over-canopy operations, respectively. From the above argument, therefore, it is very clear that autonomous mobile robots that are small, compact, and cheap are best suited for executing essential tasks to achieve high performance in phenotype data acquisition, crop scouting, under-canopy cover crop planting, and under-canopy weeds removal [4–7].

Our system represents an extension of the groundwork laid in prior research [8], which introduced a modular LiDAR-based crop-row following system. This extension aims to address inherent challenges encountered in such environments, including the presence of naturaldisturbances

Francisco Affonso and Felipe Andrade G. Tommaselli contributed equally to this work.

Extended author information available on the last page of the article like rough terrain and fluctuations in crop morphology. By adopting a modular architecture that separates perception, waypoint generation, and control, our system utilizes LiDAR-derived point cloud data to anticipate crop row configurations. These predicted row lines are then employed to generate waypoints based on geometric principles, enabling the creation of an optimized row-following path in terms of control.

Unlike previous methods [3, 5, 9, 10], we present a framework that exploits a self-supervised neural network to convert the detected row crops into lines, according to the robot's relative heading and placement in the row. Our approach is flexible, utilizing powerful state-of-the-art techniques in field robot navigation, including the fact that the perception is made in terms relative to the robot, *i.e.*, locally. Likewise, preceding works [11–13], do not solve existing problems of loss of GNSS reliability for small robots navigating under the canopy, our implemented navigation module evaluates the central axis between planting rows for generating the path. This guarantees modularity for scenarios where global positioning signals are unreliable, outperforming GNSS-only navigation. It is also important to mention that our Model Predictive Control (MPC) poses computation challenges owing to the high cost of the solvers available, presenting, therefore, an impractical implementation in embedded systems [14–17]. To deal with this problem, we use a faster solver that locally linearizes the dynamic model, termed Iterative Linear Quadratic Regulator (iLQR), so that the processing can be performed efficiently and its solution can be rapidly sought (Fig. 1).

Fig. 1 Representation of our solution that autonomously navigates in farm row crops. By using perception through LiDAR and planning trajectories, our system can provide safe navigation in agricultural scenarios

The main contributions of this paper are as follows:

- A self-supervised perception module that utilizes LiDAR data to accurately interpret the under-canopy environment by detecting crop rows, outperforming heuristic approaches such as [3];
- CROW, a complete LiDAR-based navigation system for under-canopy row-following, operates exclusively on local data interpretation, eliminating the need for odometry information;
- A fully integrated navigation solution, combining a deep learning perception module, a waypoint generator grounded in geometric principles and an optimal control based on the iterative Linear Quadratic Regulator (iLQR).

2 Related Work

The majority of existing works in agricultural environments focus on two main types of navigation methods: satellite-based global positioning system (GPS) path tracking and row-following navigation. In this section, we review these methods, highlighting how our approach leverages state-of-the-art techniques and addresses their limitations.

2.1 Classical Navigation

In classical navigation systems, sensors play a pivotal role in mapping environments through SLAM [18], which meticulously records all environmental boundaries and obstacles, enabling autonomous robot navigation between predetermined points. However, a comprehensive assessment of this method exposes notable shortcomings when deployed in outdoor settings. To address these limitations, Global Navigation Satellite Systems (GNSS) come into play, utilizing reference points to chart a trajectory for the robot [19].

Promising solutions here include the use of wide and well-established GNSS technology outdoors since it would give an absolute position estimate of the vehicle if a clear signal from the satellites is received. When so coupled with Real-Time Kinematic hardware, this technology gives positioning that is centimeter-level accurate and reliable. In time, according to the GNSS and RTK technologies, there comes the autonomous navigation system into the picture these days, which is being implemented by different types of agricultural robotic systems [10, 20]. Despite its advantages, GNSS is limited in an environment with occlusions, even with RTK corrections.

The inadequacy of these methods in such environments stems from the fact that agricultural fields often feature irregular obstacles of varying sizes, luminosity, and other factors that introduce considerable noise into the process of creating a high-fidelity map. Additionally, the rugged terrain necessitates a local perception of the surroundings to avoid unforeseen obstacles that could potentially damage the robot, rendering navigation solely reliant on satellite systems impractical.

2.2 Row Following Navigation

Exteroceptive sensors designed to navigate between crop rows have received a lot of attention from researchers [3, 5, 21–23]. These methods leverage distinctive features present in regularly planted crop rows, enhancing navigation accuracy and reliability. Even so, they still present significant challenges, such as the use of RGB images for navigation under-canopy. The need to calculate absolute distances from images makes conventional approaches that rely on extracting geometry from single images difficult, generally requiring frequent calibrations and not achieving the same accuracy as systems that use LiDAR. Furthermore, variations in visual appearance over different days and seasons restrict the effectiveness of heuristic methods employing detection algorithms, while the similarity in visual features contributes to positional inaccuracies in SLAM algorithms [24–26].

Specifically, in the under-canopy row-following task, sensors such as LiDAR are adept at detecting and following crop rows by precisely determining the distance from the rows and the angle relative to the row, using this to track specified row-relative pose [3, 5, 12, 27]. Ultrasonic [28] and infrared sensors [29, 30] can also offer robust solutions in environments with varying lighting conditions [31]. This capability is essential for navigation beneath the canopy where light levels can fluctuate greatly [32]. However, in some cases, methods that use heuristics are not robust enough to keep robots navigating without errors. Consequently, despite its higher cost, LiDAR provides a simple solution for object detection, distance estimation, and performance in low lighting compared to other sensors in the same class, such as cameras and radars [33].

2.3 Learned Navigation in Agricultural Environments

Employment of some of these sophisticated sensors with advanced machine learning algorithms to identify and track crop rows is a primary method not only in under-canopy row following but also in other areas of agricultural robotics [34–36]. A major advantage of this approach is its exceptional adaptability [37]; current machine learning models can be trained to recognize and adjust to various crop types and row configurations.

The framework proposed in Sivakumar et al. [9] takes a supervised learning approach, where a network model is trained for better discrimination between traversable and untraversable areas and shown to be effective for autonomous navigation. In a similar way, Sivakumar et al. [38] uses a keypoint-based under-canopy navigation system, which processes RGB images to generate heatmaps for semantic keypoints. However, this requires manual labeling, which is tedious and time-consuming. Moreover, the reliance on labeled datasets can limit the system's robustness, particularly when visual cues change significantly. In contrast to these approaches, Huang et al. [39] presents an end-to-end learning-based row-following system specifically designed for structured apple orchards. The system maps raw pixel data directly to driving commands, eliminating the need for multiple subtasks such as boundary detection and manual labeling. We instead leverage a probability distribution to create a row predictor that can interpret the under-canopy environment and predict waypoints for good navigation.

3 System Design

This paper introduces CROW, a system designed for row-following navigation, illustrated in Fig. 2. CROW seamlessly integrates the robustness and generalization capabilities of Self-supervised Deep Learning with the swift solving capabilities of an iterative Linear Quadratic Regulator, this system empowers our robotic platform, Terrasentia.

Consisting of three essential modules, the system begins with a Deep Learning perception module (Section 3.2). This component uses a LiDAR scan to analyze row crop data and identify local lines through a self-supervised trained neural network approach (Sections 3.3 and 3.4). Subsequently, a waypoint generator determines a future setpoint for the controller to pursue, using the perception crop lines (Section 3.5). Finally, the system utilizes a Nonlinear Model Predictive Controller to steer itself toward the designated waypoint through a series of actions. The optimization problem is effectively solved using a constrained iLQR solver (Section 3.7), with a kinematic model (Section 3.6) to model the system's motion.

To design our navigation system, we make the following assumptions the leverages the planting structure in traditional agricultural environments:

- The environment remains static without moving obstacles, enabling the robot to follow the next waypoint and apply control without encountering collisions;
- The crops are planted with a consistent distance between the rows;
- The row curves can be approximated as local lines to generate waypoints with minimal distances.

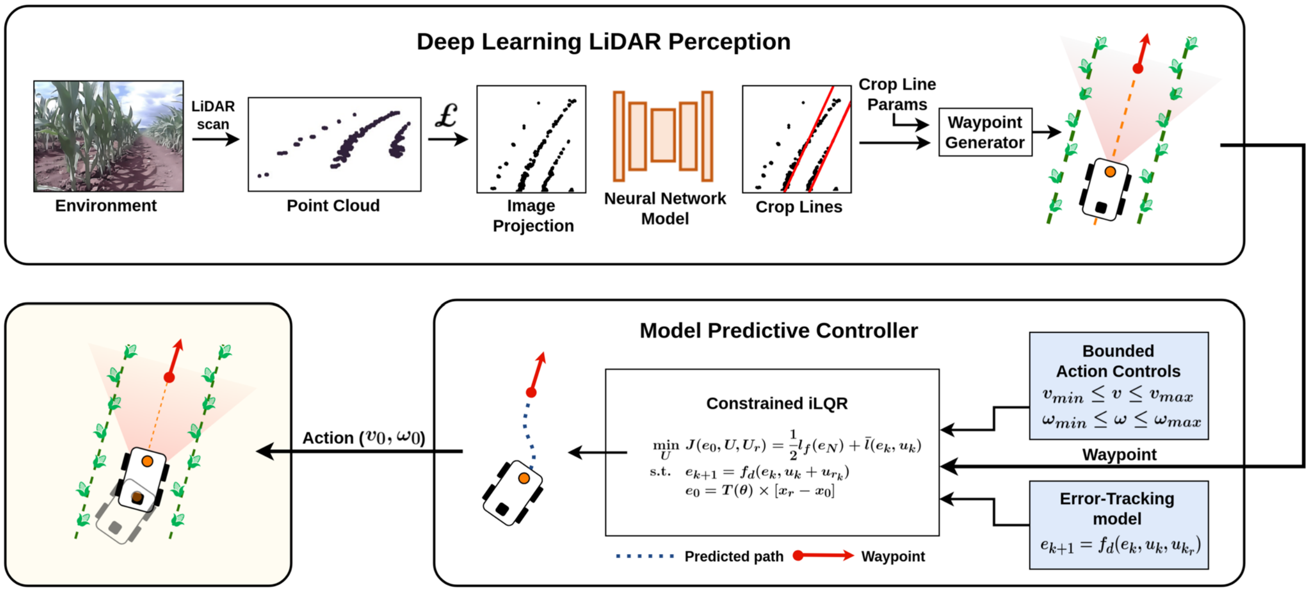

Deep Learning LiDAR Perception
Environment
LiDAR scan
Point Cloud
£
Image Projection
Neural Network Model
Crop Lines
Crop Line Params
Waypoint Generator
Model Predictive Controller
Constrained iLQR
$\min_{U} J(e_0, U, U_r) = \frac{1}{2} l_f(e_N) + \bar{l}(e_k, u_k)$
s.t. $e_{k+1} = f_d(e_k, u_k + u_{r_k})$
$e_0 = T(\theta) \times [x_r - x_0]$
Bounded Action Controls
$v_{min} \leq v \leq v_{max}$
$\omega_{min} \leq \omega \leq \omega_{max}$
Waypoint
Error-Tracking model
$e_{k+1} = f_d(e_k, u_k, u_{k_r})$
Predicted path
Waypoint
Action (v_0, ω_0)

Fig. 2 CROW outlines perception, waypoint generator, and controller modules. It starts with a LiDAR scan of a covered crop area, producing an image after a £ transformation of the point cloud. This image feeds into a neural network to approximate crop data into lines, used to generate waypoints. These waypoints guide the Model MPC, which uses a constrained iLQR solver to navigate the row

3.1 Robot Platform

Terrasentia, developed by Earthsense Inc., is a skid-steer robot tailored for effective high-throughput field phenotyping data collection in agriculture. To enhance this data collection process, the platform receives various solutions for plantation navigation [3, 5, 9, 12]. Additionally, our system utilizes the Hokuyo URG-04LX-UG01 LiDAR, boasting a maximum range of 30m with a scan speed of 25ms. For algorithm execution, the platform is equipped with an NVIDIA Jetson AGX Orin computer.

3.2 Perception

The system detailed in this document targets generic agricultural environments, such as corn plantations, and is adaptable to corridor-type environments with parallel delimiters. The LiDAR sensor requires surfaces for laser reflection, making this a crucial prerequisite for the Perception system.

With the goal of implementing an Optimal Control strategy alongside a Local Perception system, we developed the Waypoint Generator, a middle-man system that generates setpoints for the Controller based on the Perception data. Inspired by Higuti et al. [3], our system advances the perception framework using Deep Learning, shifting away from earlier heuristic methods. We selected Perception Crop Lines as the predicted output of the Perception system because they enable the generation of geometric-based setpoints. Figure 3 illustrates the LiDAR data visualization, while the second figure depicts the corresponding image projection.

The parameters of interest include distances dL and dR, and azimuth ϕ, represented in Fig. 4. However, this information is intrinsically encoded in the Crop Lines, which

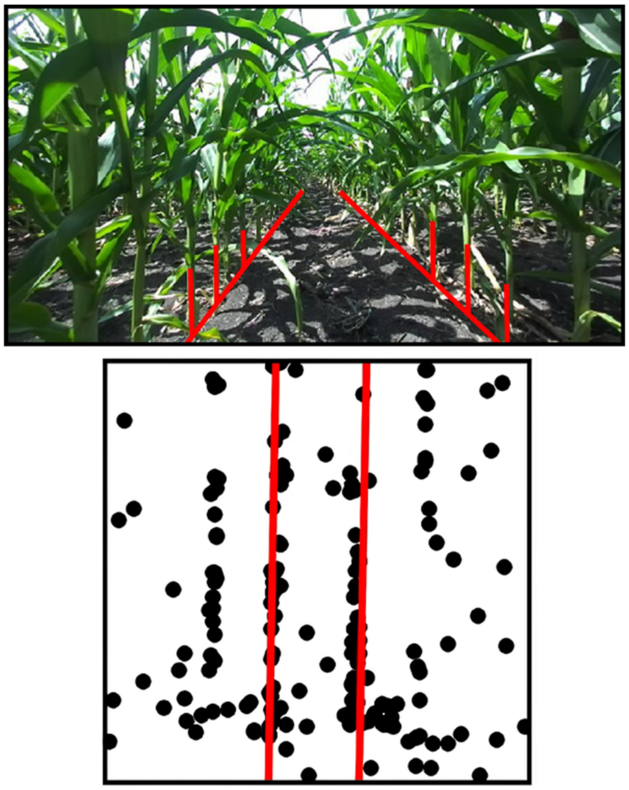


Fig. 3 Characterization of the under-canopy environment, showing the Robot's camera view on the left and the corresponding LiDAR reading on the right

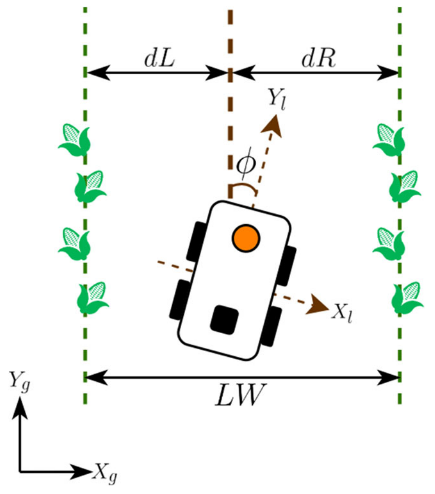


Fig. 4 Schematic of TerraSentia's navigation in the global coordinate system G and local relative to the robot L

contributes to the setpoint generation. The desired perception output, defining the reference path, are the line equations parameters m_1, m_2, b_1, and b_2. The line equation format adopted for the ξ *line* is represented by Eq. 1.

$$y_\xi = m \cdot x_\xi + b. \tag{1}$$

In future sections, the inference error will be defined as the Error function calculated from the predicted and reference parameter values across all four parameters. Given that one of the assumptions in system design is the parallelism between the crop rows, in this work, $m_1 = m_2$ is assumed.

Since LiDAR readings serve as the primary source of information, the focus of environmental characterization revolves around data noise. This noise can arise from various factors, including bumpy terrain, uneven leaf distribution, asymmetrical crop rows, sensor occlusion, and others. In Fig. 3 it is possible to visually understand these noise generation factors.

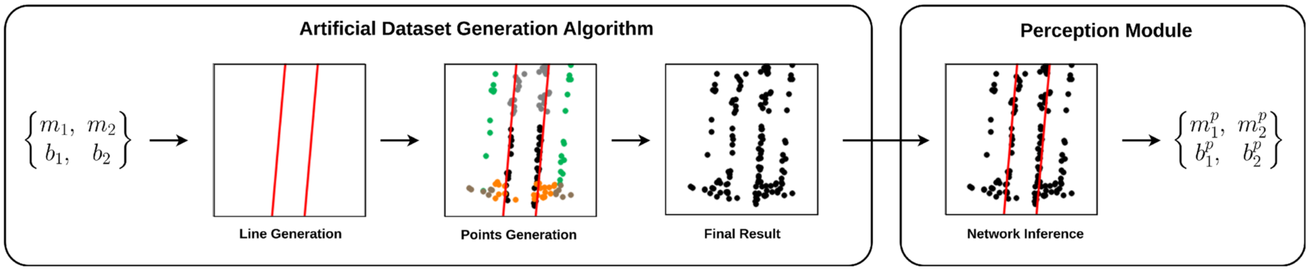
Heuristic methods, as discussed in Higuti et al. [3], face two primary challenges: noise from readings and sensor occlusion, particularly in cases of partial occlusion where some useful data is still available. These issues not only motivated the use of neural networks in CROW, but also render manual labeling impractical, as crop rows are often not defined by the two main lines directly due to these challenges.

Furthermore, it is difficult to gather a high-quality dataset, since manually controlling the robot in a tight space requires expertise and can potentially damage the crop. Consequently, collected datasets tend to be sparse and uneven, resulting in significant biases in the neural network. To address this problem, we developed a self-supervised learning pipeline that creates labeled artificial input data, enabling the neural network to effectively manage noise with a balanced, extensive, and diverse dataset. We explored different architectures and techniques to ensure that such approach not only converges to a solution but also demonstrates feasibility with real-world data. The details of these self-supervised learning architectures and techniques are presented in Section 3.3, while the method for generating artificial data is discussed in Section 3.4.

We propose using an image of the projected LiDAR point cloud as input for the Perception system, via a, denotated, £ transformation that formats the image appropriately. Projecting the initial meters of the point cloud onto an image allows the neural network to leverage additional information and ideally develop spatial intuition, as shown in Fig. 3 for example. Consequently, while this approach requires slightly more computational power, experimental tests showed improved performance with image input (Fig. 5).

3.3 Self-Supervised Learning

Due to project constraints that make manual labeling impractical, we propose a two-step approach that utilizes self-supervised learning. This method involves an algorithm that leverages prior human interpretations to produce large artificial datasets. These datasets consist of labeled parameters from Eq. 1 and their corresponding images that show projected point clouds. The images are entirely generated by the



Artificial Dataset Generation Algorithm
$\begin{Bmatrix} m_1, & m_2 \\ b_1, & b_2 \end{Bmatrix}$
Line Generation
Points Generation
Final Result
Perception Module
Network Inference
$\begin{Bmatrix} m_1^p, & m_2^p \\ b_1^p, & b_2^p \end{Bmatrix}$

Fig. 5 The Artificial Dataset Generation pipeline for the Perception system, starting with an angle θ. From this angle, the label parameters $\{m_1, m_2, b_1, b_2\}$ are calculated algebraically. Points are generated based on these label lines to produce the final image based on the label parameters. The network then performs inference on this image for training, evaluation or testing, resulting in the parameters $\{m_1^p, m_2^p, b_1^p, b_2^p\}$

Table 1 Practical results for architecture selection

Architecture	Params	Final Error[1]	Inference Time
MobileNet	3.5M	0.037	32.77 ms
EfficientNet	5.3M	0.061	50.67 ms
ResNet18	11.7M	0.109	101.28 ms
VggNet16	138.4M	0.446	239.49 ms

[1]Final Error Mean for 100 runs

algorithm and are not derived from computational simulations or any physical related model.

The prediction is made by a single neural network trained offline, representing a self-supervised learning approach. The training and inference processes are essentially the same. During each training iteration, a random batch is provided to the neural network, and the quality of its predictions is compared to the labels from our artificial datasets, initiating the optimization process that trains the model.

Through extensive practical tests, a predefined encoder architecture was chosen to build the neural network. The base results in Table 1 show that the MobileNet V2 architecture is the most suitable for the task. Three other notable architectures considered during the selection were VggNet16, ResNet18, and EfficientNet.

The average error was assessed using a distinct test set to prevent bias in the analysis. The test was conducted 100 times on various *batches*, assessing the average error to gauge the quality of the final model achieved. The inference time aligns with this approach using the same test set.

The MobileNet architecture, known for its efficiency in computational power and performance [40], was adapted by adding a Convolutional Layer to match input dimensions. In this specific application, sequential linear layers and one-dimensional batch normalization were introduced after the final convolutional layer of MobileNet V2. This adjustment, represented by the Classifier in Fig. 6, transforms the network's output into four prediction values, typically using four float values, with the option to reduce to three if the results are highly reliable.

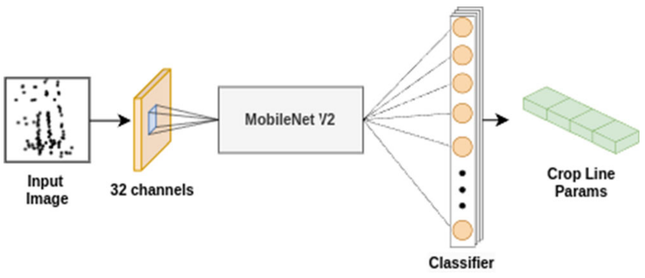


Fig. 6 Schematic diagram of the MobileNet V2 adaptation, illustrating the input and output pipelines. On the input side, the projected image is processed through a convolutional layer to modify the image dimensions for input on the MobileNet V2 framework. The MobileNet's output then passes through a Integration Layer, which returns 4 values known as the Crop Line Parameters

The L1 Loss function was utilized as the cost function (and error estimator) for the network training, which functions similarly to an MSE cost function but is less sensitive to outliers. The L1 Loss showed improved performance due to the presence of noise and misreadings in the LiDAR data.

3.4 Artificial Datasets

The algorithm for creating the artificial dataset utilizes previous human analysis to randomly place points within a fixed probabilistic distribution, as the environmental elements tend to remain constant. We define the line inclination angle θ to range from -25° to 25°, producing 50 image packs for each θ value. The total number of images is determined by the number of images per pack, with a higher number of images in packs centered around the 0 value to enhance reliability in this region, where the network is generally more sensible.

In Fig. 5 it is possible to visualize the algorithm pipeline to generate one image for a given θ angle. This process can be described by the following steps:

1. Line Generation: In this step, two lines are generated with inclination θ, and the distance between them is randomly chosen from a specified range. The labels $\{m_1, m_2, b_1, b_2\}$ are stored;
2. Points Generation: The algorithm generates points in specific regions based on a probability distribution derived from Random or Normal functions, using the label line as a reference. This process includes various stochastic parameters to ensure data uniqueness;
3. Final Result: The final output can be visualized in Fig. 5. In this phase, random noise is typically added to the image, with 0 to 50 points distributed randomly.

The network will use this image for inference during training or evaluation, predicting the values $\{m_1^p, m_2^p, b_1^p, b_2^p\}$. These predictions can then be used to calculate the error against the reference values $\{m_1, m_2, b_1, b_2\}$.

Focusing on the Points Generation step, Figure 7 visually illustrates the split nuclei regions, each accompanied by a unique probability distribution. As mentioned earlier, our approach to point generation was centered on identifying patterns in these regions where points tend to cluster. Consequently, the variation between images will stem from the number and arrangement of points in these specific regions. We utilized five regions for this purpose.

- Black: This region identifies the main crop lines where points tend to concentrate. It serves as the primary source of information for crop line identification in human classification;

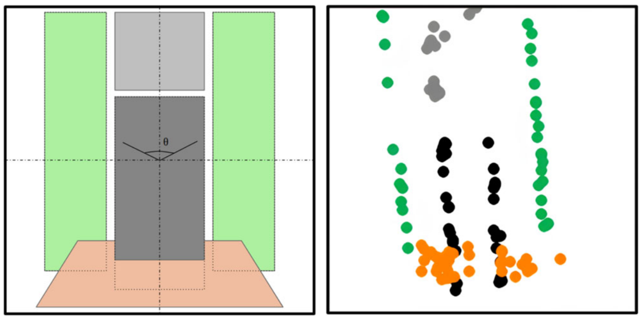


Fig. 7 The left image characterizes regions for image generation, with each region defined by specific rules based on Random or Normal functions that use the label line as a reference (the called probability distribution). The right image displays a sample from this distribution, with $\theta = -4°$

- Green: This region pertains to adjacent lines, which may or may not be present. In the constrained under-canopy environment, these adjacent lines can interfere with row reading;
- Orange: This lower region typically contains LiDAR noise from leaves near the sensor, making it difficult to extract reliable information. However, the authors noted a connection between some readings in this region and the Black region. Consequently, these regions overlap to enable the network to recognize this relationship and mitigate misinformation from the noisy data;
- Grey: The upper grey area consists of distant points that are not particularly useful for inference. Nonetheless, they often correlate with the number of points in the black areas, indicating that when main lines are obscured, the quantity of points in the grey area generally increases. This behavior can help the neural network gather more information about the environment when the main lines are not dependable.

Each nucleus has patterns that enable point placement via the Random or Normal function. The Normal function is used for clustering points in small groups to preserve the stochastic behaviors in the fixed skeleton; otherwise, the Random function is applied.

The creation of artificial datasets, despite the many practical and efficient benefits, poses the challenge of ensuring that the artificially generated dataset accurately represents the real environment. For this, empirical tests are crucial in ensuring that the hyperparameters for generating these points are properly tuned, resulting in representative images. The optimal model developed in this work required 21000 artificially generated labeled images for training. A small sample of the dataset illustrating the previously described patterns is shown in image Fig. 8.

To address challenging edge cases in input readings, such as sensor occlusion, we propose incorporating noise into network training at the final stage of the artificial image generation. Figure 9 depicts this process: the left plot displays the generated image after the already presented pipeline of Lines and Points generation, while the right plot shows the final version with added noise.

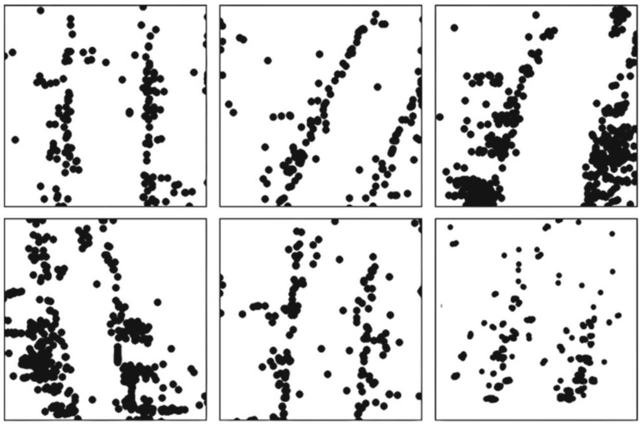


Fig. 8 Sample from the artificial dataset used for training in this work, featuring 8 images with distinct configurations that illustrate the structured corridor problem, adapted to our environment's rules

Figure 10 illustrates the prediction system's performance in occlusion scenarios. The left image shows a prediction that slightly veers left, following the crop lane lines, demonstrating a strong prediction. In the subsequent image, sensor occlusion results in limited data points; despite this, the network still provides a coherent left-leaning prediction. The next image depicts the situation immediately after the occlusion, where the network continues on the same path.

This suggests that the network effectively utilizes various environmental cues to generate coherent predictions, even when the central lane lines are obscured. It's worth noting that the network currently lacks a memory component, meaning each prediction is based solely on the available information.

As a result, we observe a significant improvement in effectiveness across different scenarios, with the same model performing notably better in diverse environments. This thesis is supported by the Results section, which presents

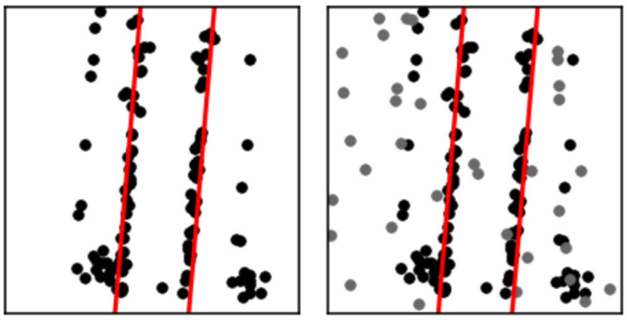


Fig. 9 Process of noise addition, where the gray points in the right image represent an example of random noise throughout the image

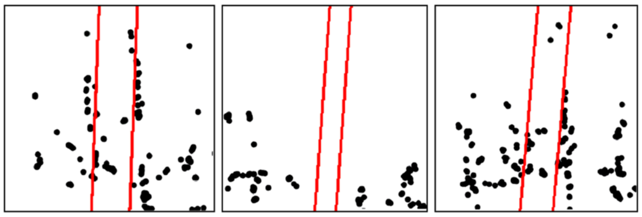


Fig. 10 Real-world test showing the evolution of the prediction through time of the system over a moment with partial occlusion of the LiDAR caused by a leaf

compelling Perception outcomes from both Gazebo simulations and real-life tests, all utilizing the same model.

In addition, artificial dataset generation for neural network training is a common practice and, as shown in the referenced work, enables training in cases of Dataset Shift. In such scenarios, the dataset's immutability cannot be guaranteed, leading to the need for frequent retraining with new datasets. These challenges are discussed in [41, 42], supporting the approach taken.

The works [43–45] support the approach of generalizing neural network learning through the addition of noise. The maximum amount of noise to be added is tested empirically and varies between different batches of the dataset.

3.5 Waypoint Generator

To bridge the perception and controller modules, we employed a waypoint generator. This generator utilizes geometric principles to approximate the central axis among planting rows, derived from inference lines of perception, to generate future waypoints for row-following, as illustrated in Figure 11.

Since the inference lines are referenced from the robot's position, the central axis is calculated as the average between the left and right lines.

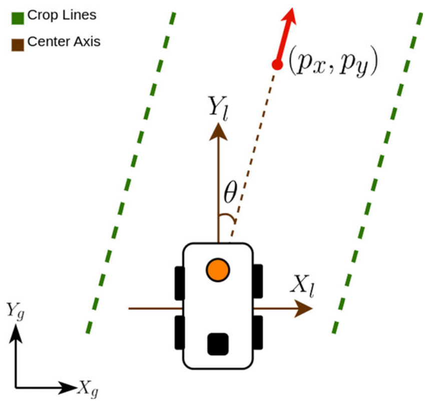


Fig. 11 Representation of the derivation of a local waypoint. In the figure, (p_x, p_y) indicates the position of the waypoint, and θ represents the heading angle

Using the central axis and Euclidean distance to account for potential distortions between the axes, we calculate a set point at a distance D from the robot. This forms the basis for determining the waypoint coordinates in the local frame:

$$\begin{cases} y_{px} = m \cdot x_{px} + b \\ D = \sqrt{x_{rm}^2 + y_{rm}^2} \end{cases} \tag{2}$$

where m and b represent the slope and intercept of the center axis, respectively. Additionally, the subscript px denotes image pixel coordinates, and rm denotes row meter coordinates.

Solving this system provides the position in row coordinates, already in local frame coordinates $(p_x, p_y) = (x_{rm}, y_{rm})$, with the heading angle obtained using $\theta = \arctan 2(x_{rm}, y_{rm})$.

Finally, we established a local waypoint to serve as a reference for the controller to follow the crop row.

3.6 Kinematic Model

In this paper, we utilize the nonlinear form of the robot error-tracking model [46]. This employs a unicycle kinematic model to determine changes in the robot's position and orientation, given by the equations:

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v \cdot \cos\theta \\ v \cdot \sin\theta \\ \omega \end{bmatrix} \tag{3}$$

where $x = [p_x \;\; p_x \;\; \theta]$ is the state vector containing the position and orientation of the mobile robot and $u = [v \;\; \omega]$ is the action control vector containing the linear and angular speeds.

With these definitions, it is possible to obtain an error state $e \in \mathbb{R}^3$ between the pose of the waypoint and the robot. Introducing a reference state $x_r = [p_{x_r} \;\; p_{y_r} \;\; \theta_r]$ and a reference action control $u_r = [v_r \;\; \omega_r]$, the error state is obtained as:

$$e = T(\theta) \times [x_r - x] \tag{4}$$

where $T(\theta)$ is the transformation matrix shown below:

$$T(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

After applying the transformation, the error state vector comprises longitudinal, lateral, and angular errors. This adjustment yields a kinematic model that circumvents the singularities present in the unicycle model when the linear velocity approaches zero, as discussed in González-Sierra et al. [47].

Associating the derivation of the error state $\dot{e}$ with the kinematic relations of the unicycle model, we can obtain the dynamic model based on the error tracking.

$$\dot{e} = f(e, u, u_r) = \begin{Bmatrix} v_r \cos e_3 + \omega e_2 - v \\ v_r \sin e_3 + \omega e_1 \\ \omega_r - \omega \end{Bmatrix} \tag{6}$$

The development of this equation allows us to present a nonlinear state space of this error-tracking model, discretized using a sampling time (Δt).

$$e_{k+1} = f_d(e_k, u_k, u_{r_k}) = e_k + \left(\begin{bmatrix} -1 & e_2 \\ 0 & e_1 \\ 0 & -1 \end{bmatrix} u + \begin{bmatrix} \cos e_3 & 0 \\ \sin e_3 & 0 \\ 0 & 1 \end{bmatrix} u_r \right) \Delta t \tag{7}$$

Remark 1 For controllability, it's necessary that $v_r \neq 0$ or $\omega_r \neq 0$, as the system is linearized around the current state.

3.7 Controller

The controller employed for the robot utilizes a Nonlinear Model Predictive Controller (NMPC), which employs a Constrained Iterative Linear Quadratic Regulator (iLQR) as its solver [48].

For prediction, we denote a trajectory $\{E, U, U_r\}$ containing the sequence of states $E = \{e_0, e_1, ..., e_k\}$, the sequence of action controls $U = \{u_0, u_1, ..., u_{k-1}\}$, and the sequence of reference action controls $U_r = \{u_{r_0}, u_{r_1}, ..., u_{r_{k-1}}\}$, over the prediction horizon. Moreover, we introduce inequality constraint functions to ensure bounded action control, which can be generalized as:

$$g_i(u_k) \leq 0, \quad k = 0, 1, ..., N-1 \tag{8}$$

where i is the index of the inequality constraints. Finally, we can present our optimization problem:

$$\begin{aligned} \min_{U} \quad & J(e_0, U, U_r) = \frac{1}{2} l_f(e_N) + \frac{1}{2} \sum_{k=0}^{N-1} l(e_k, u_k) \\ \text{s. t.} \quad & e_{k+1} = f_d(e_k, u_k + u_{r_k}) \\ & e_0 = T(\theta) \times [x_r - x_0] \\ & g_i(u_k) \leq 0, \\ & k = 0, 1, ..., N-1 \end{aligned} \tag{9}$$

where the total cost is defined as the sum of running costs l and terminal cost l_f over a horizon N.

The running and terminal costs follow linear quadratic functions, mirroring the principles of the Linear Quadratic Regulator [49]. These cost functions are defined as:

$$\begin{aligned} & l_f(e_N) := \|e_N\|^2_{Q_f} \\ & l(e_k, u_k) := \|e_k\|^2_Q + \|u_k\|^2_R \end{aligned} \tag{10}$$

where Q_f, Q, and R represent positive definite weight matrices for the final state, state, and action control, respectively.

3.7.1 Barrier Method

To handle inequality constraints, the barrier method was used [50]. This method serves as a penalty mechanism that expands the feasible region of the optimization problem to ϕ, but imposes a substantial cost on the cost function for points lying outside this region.

$$I_\phi(U) = \begin{cases} 0, \text{ for } U \in \phi_i \\ \infty, \text{ otherwise} \end{cases} \tag{11}$$

where

$$\phi := \{U | g_i(u_k) \leq 0, k = 0, 1, ..., N-1\} \tag{12}$$

Continuing, this set isn't differentiable, which is a requirement for calculating gradients and Hessians in iLQR steps. Therefore, we employ a logarithmic barrier function to approximate it, ensuring desirable properties such as continuity, differentiability, and convexity.

We can define a new running cost by incorporating this function:

$$\bar{l} = \frac{1}{2} \sum_{k=0}^{N-1} l(e_k, u_k) - \frac{1}{t} \sum_{i=1}^{j} \log(-h_i(u_k)) \tag{13}$$

where the parameter t determines the accuracy of the approximation. As t approaches infinity, the function becomes more accurate with ϕ.

Furthermore, the optimization problem is reformulated as:

$$\begin{aligned} \min_{U} \quad & J(e_0, U, U_r) = \frac{1}{2} l_f(e_N) + \bar{l}(e_k, u_k) \\ \text{s. t.} \quad & e_{k+1} = f_d(e_k, u_k + u_{r_k}) \\ & e_0 = T(\theta) \times [x_r - x_0] \\ & k = 0, 1, ..., N-1 \end{aligned} \tag{14}$$

3.7.2 Backward Pass

This setup allows us to solve Eq. 14 using the principle of differential dynamic programming (DDP). Thus, Bellman's

Principle of Optimality defines a state value function V_k representing the minimum cost-to-go at each state:

$$\begin{aligned} V_k &= \min_U \left\{ \bar{l}(e_k, u_k) + V_{k+1}(f_d(e_k, u_k)) \right\} \\ V_N &= l_f(e_N) \end{aligned} \tag{15}$$

The cost remaining in the k_{th} iteration depends on the $(k+1)th$ iteration, establishing a connection between steps that enables the backward pass to compute the values of V_k, starting with the last known remaining cost V_N.

Furthermore, deviations of e_k and u_k are introduced as δe_k and δu_k, respectively, to allow us to linearize the dynamic model over a current state and action control:

$$\delta e_{k+1} \approx A(e_k, u_k)\delta e_k + B(e_k, u_k)\delta u_k \tag{16}$$

where $A \equiv \frac{\partial f_d}{\partial e}$ and $B \equiv \frac{\partial f_d}{\partial u}$.

Using the iLQR and DDP methods, we introduce a perturbed action-value function, by conducting a second-order Taylor expansion centered at the current state e_k and action u_k, using the linearized kinematic model. This expansion enables us to estimate how the action value evolves when the state and action variables undergo minor adjustments.

$$\begin{aligned} \delta Q_k(\delta e_k, \delta u_k) \approx \frac{1}{2} \begin{bmatrix} \delta e_k \\ \delta u_k \end{bmatrix}^T \begin{bmatrix} Q_{ee} & Q_{eu} \\ Q_{ue} & Q_{uu} \end{bmatrix} \begin{bmatrix} \delta e_k \\ \delta u_k \end{bmatrix} + \\ + \begin{bmatrix} Q_e \\ Q_u R \end{bmatrix}^T \begin{bmatrix} \delta e_k \\ \delta u_k \end{bmatrix} \end{aligned} \tag{17}$$

where

$$\begin{aligned} Q_e &= \bar{l}_e + A_k \ (V_{k+1})_e \\ Q_u &= \bar{l}_u + B_k \ (V_{k+1})_e \\ Q_{ee} &= \bar{l}_{ee} + A_k^T \ (V_{k+1})_{ee} \ A_k \\ Q_{ue} &= \bar{l}_{ue} + B_k^T \ (V_{k+1})_{ee} \ B_k \\ Q_{uu} &= \bar{l}_{uu} + B_k^T \ (V_{k+1})_{ee} \ A_k \end{aligned}$$

here the subscripts denote the partial derivative concerning the corresponding variable.

With these equations, it is possible to find the optimal δu_k minimizing the perturbed Q-function.

$$\delta u_k^* = \arg\min_{\delta u_k} \delta Q_k(\delta e_k, \delta u_k) \tag{18}$$

Solving Eq. 18 we can determine the backward pass gains to find the optimal correction action control:

$$\delta u_k^* = d_k + K_K \delta e_k \tag{19}$$

with

$$\begin{aligned} d_k &= -Q_{uu}^{-1} Q_u \\ K_k &= -Q_{uu}^{-1} Q_{ux} \end{aligned} \tag{20}$$

Having this solution, we can plug it back into the equation of the Q-perturbed function to calculate the difference in the state value function and previous gradients and Hessians:

$$\begin{aligned} \Delta V &= \frac{1}{2} d^T Q_{uu} d + d^T Q_u \\ (V_k)_e &= Q_e - K^T Q_{uu} d + K^T Q_u + Q_{ue}^T d \\ (V_k)_{ee} &= Q_{ee} + K^T Q_{uu} K + K^T Q_{ue} + Q_{ue} K \end{aligned} \tag{21}$$

These derivatives can be used to calculate the preceding optimal correction action controls, for all trajectory U, following the idea behind the backward pass.

3.7.3 Forward Pass

After completing the backward pass, the next step is the forward pass. Here, we update the nominal trajectory by integrating forward from the initial state. Additionally, we use a backtracking search parameter α to ensure convergence.

$$\begin{aligned} u_k &= \hat{u_k} + K_k(e_k - \hat{e}_k) + \alpha d_k \\ e_{k+1} &= f_d(e_k, u_k) \end{aligned} \tag{22}$$

where the hat indicates the nominal value of the respective variable.

3.7.4 Implementation

It's important to note that during this process, certain Q-terms might lose positive definiteness. To address this issue, regularization is applied to the matrices Q_{uu} and Q_{ux} by incorporating an identity matrix multiplication (ρI) into the computations with the gradients of the kinematic model. Consequently, the updated Q-terms ($\tilde{Q}uu$ and $\tilde{Q}ux$) are utilized to calculate the backward pass gains [15].

Hence, this algorithm operates at a frequency to predict the path between the robot's pose and the waypoint, and it applies the first action control to the robot. The pseudocode of the constrained iLQR is outlined in Algorithm 1. It's worth noting that in the actual algorithm, several stopping conditions are incorporated, such as a tolerance threshold for ΔV and regularization.

Algorithm 1 Iterative Linear Quadratic Regulator

```
Input: $U, e_0$
Output: $u_0$
1: $E, J \leftarrow$ rollout($e_0, U$)
2: for $i = 0, 1, ..., i_{max}$ do
3:     $K, d, \Delta V \leftarrow$ BackwardPass(E, U)
4:     for $\alpha_i : \alpha$ do
5:         $(E, U, J)_{new} \leftarrow$ForwardPass($K, d, \alpha_i$)
6:         if $|J_{new}| < |J|$ then
7:             $(E, U, J) \leftarrow (E, U, J)_{new}$
8:         end if
9:     end for
10: end for
```

4 Experimental Results

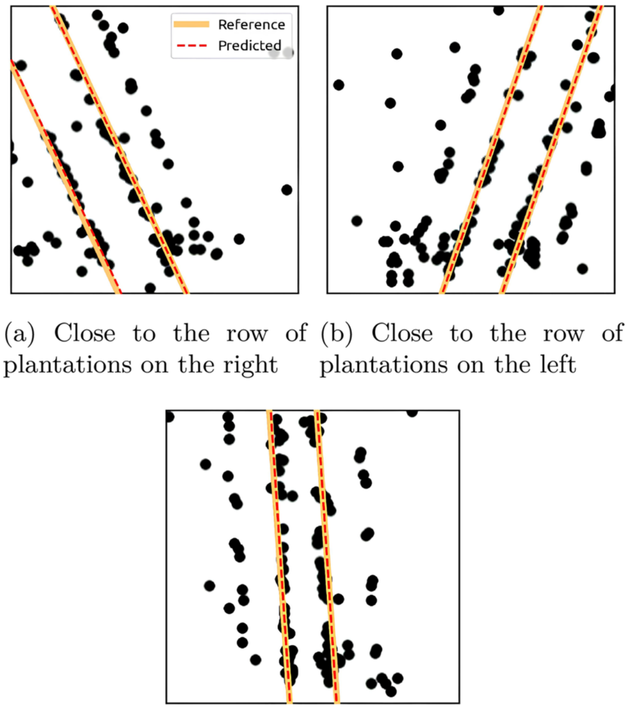
In this section, we present the experimental results obtained from both simulation and real-world scenarios, utilizing the TerraSentia platform (Fig. 1) to apply our methodology. Additionally, we validate our system by comparing it with a state-of-the-art method in simulation and demonstrating its functionality in real crop environments.

4.1 Perception

The Perception system receives raw LiDAR data and outputs the crop lines in the line equation format $\{m_1, m_2, b_1, b_2\}$. It is worth noting that, in the bellow experiments, the predictions with the neural network were made assuming $m_1 = m_2$.This correction enhances the efficiency of neural network training and has demonstrated positive results empirically.

To evaluate inference accuracy, the error was calculated from prediction outputs on two datasets: the training set with 21,108 images and the testing set with 11,440 images. The L1 Loss function, the default for this works training, was used to ensure result reproducibility. Table 2 presents the mean μ and standard deviation ρ of the Absolute Error metric for each dataset.

Figure 12 illustrates the quality of the neural network's predictions. The image depict examples of predictions from three distinct positions: near the right plantation, near the left plantation, and at the center. These examples reveal that the prediction line closely follows the reference line, indicating the network's high accuracy across all this extreme scenarios. The absolute errors for the images are as follows:



(a) Close to the row of plantations on the right (b) Close to the row of plantations on the left

(c) Centered in the middle of the hallway

Fig. 12 Different prediction extremes for the Neural Network inference, compared to the reference

- (a) $m_1 : 0.411, b_1 : 0.175, b_2 : 0.286;$
- (b) $m_1 : 0.380, b_1 : 0.552, b_2 : 0.501;$
- (c) $m_1 : 0.619, b_1 : 0.768, b_2 : 0.373.$

Comparing the values of Fig. 12 to those in Table 2, it is evident that all fall within the range defined by the table's mean and standard deviation.

By refining the analysis scope, it is possible to link the errors values to real-world measurements. Using b_1 and b_2 outputs enables us to evaluate the actual deviation in meters these errors would cause, based on the same assumptions as the deployment code. To accurately depict the error geometrically, the difference between the prediction and the label (called Absolute Loss) is preferable to L1 Loss, as L1 Loss distorts the geometric interpretation of the value.

The conversion from b_{pixel} to b_{meters} can be achieved with the following expression:

$$b_{meters} = \frac{H_{meters}}{H_{pixels}} \cdot b_{pixels} \tag{23}$$

Table 2 Inference Accuracy (μ,ρ)

Dataset	m_1	b_1	b_2
Train	(0.489, 0.308)	(0.488, 0.327)	(0.521, 0.355)
Test	(0.563, 0.321)	(0.558, 0.360)	(0.602, 0.373)

H_{meters} and H_{pixels} denote the Horizon in meters and pixels, respectively. This metric reflects the maximum value from the images. In this project, all images have a height of 224 pixels, and the experimental Horizon for testing was set at 3 meters. Thus, the conversion factor is $3/224$.

The Test set errors (μ,ρ) for b_1 and b_2 were (0.7473, 0.4821) and (0.806, 0.499) centimeters, respectively. The test set typically performs worse than the train set but better reflects real-case scenarios. Thus, most errors are statistically below 1.5 centimeters, which is almost negligible given the 1.8-meter row gap.

4.2 Controller

Before presenting the full system results, we conducted a benchmark of the controller by comparing our implementation of the Model Predictive Controller, which uses a Constrained Iterative Linear Quadratic Regulator (iLQR) as the solver, with the state-of-the-art IPOPT solver [17]. It is important to highlight that while our constrained iLQR employs the kinematic model developed in Section 3.6, the IPOPT was configured with the unicycle model. This comparison allowed us to identify the advantages of our controller relative to other state-of-the-art methods.

The comparison, shown in Table 3, was measured by running the system in simulation, which is described in more detail in Section 4.3. The table displays the RAM usage (from a total of 8GB) and the time taken to compute each control action. Both metrics are crucial for ensuring that our system does not overburden the robot's hardware with the controller module and can generate control actions more frequently, making the robot more robust.

Considering that the time to complete the simulation run is nearly identical across controllers, with each outperforming the other by only a few seconds (3% of the total runtime) across the experiments, we can analyze the data in Table 3 under the assumption that both controllers performed equivalently for our intended purpose.

Analyzing the results, it is worth highlighting that, in all cases-taking the average of both straight and curved scenarios-our method computes the control actions faster. However, the most significant observation is the difference in RAM usage during these calculations. The iLQR solver only utilized 0.28%-0.30% of the available memory, whereas IPOPT consumed between 17% and 30%. This substantial reduction in resource consumption makes our method far more efficient.

Table 3 Comparison between our iLQR and IPOPT

	Straight		Curved	
	iLQR	IPOPT	iLQR	IPOPT
Av. Memory [%]	**0.28**	10.50	**0.30**	14.30
Av. Solver [ms]	**17**	55	**30**	55

4.3 Simulation

To conduct our experiments, we used the Gazebo simulator [51], which is renowned for its flexibility in robotics simulation. Specifically, we employed the Virtual Maze Field package [52] to create plantation environments as close as possible to the real ones; hosting the virtual TerraSentia, which was equipped with identical sensors to those found on the actual robot. The setup could allow for rigorous navigation system testing under controlled yet realistic conditions. To thoroughly assess the system's adaptability and accuracy, some variations of the plantation parameters were made, such as plant density, row spacing, and crop height. The adaptations allowed one to simulate absolutely the principal types of agricultural environments and system performance with the whole diversity of scenarios from mild to most challenging ones.

In our first set of experiments, we compare our method with a baseline [3], with both systems navigating through straight and curved plantations, enabling us to assess the accuracy of trajectory generation relative to a reference. Given our assumption of consistent row spacing, the reference line is defined as the center line with an equal distance from each row crop. In this scenario, the task was to navigate autonomously through the entire crop row with the robot starting position close to the crop lane entrance, aligned with the rows, making it easier to start planning the path between the planting lanes.

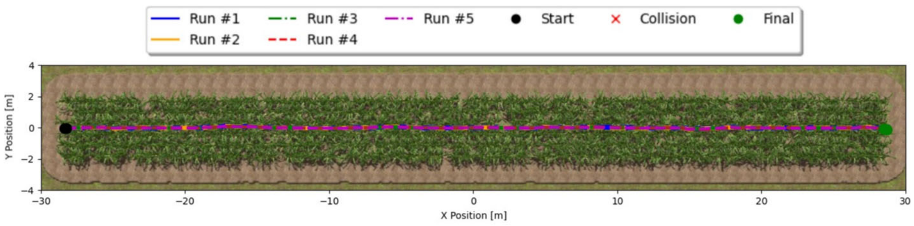
The trajectories generated by our system and the baseline (Higuti et al. [3]) are depicted in Fig. 13. Our system exhibits a higher degree of accuracy compared to the baseline. Particularly in curved scenarios, our system is capable of generating trajectories without collisions, whereas the baseline fails to do so. Furthermore, each test was conducted five times, and the results are presented in Table 4.

In straight scenarios, CROW achieved a 100% collision-free rate, covering an average distance of 57.5 meters, whereas the Higuti et al. [3] method had a 20% collision rate, with an average of 50.56 meters covered. This demonstrates CROW's superior accuracy and efficiency. In the more complex curved scenarios, CROW again showed better results, maintaining a 100% collision-free rate and covering 44 meters on average. The Higuti et al. [3] method, however, struggled significantly, with an 80% collision rate and only 24.33 meters covered on average.

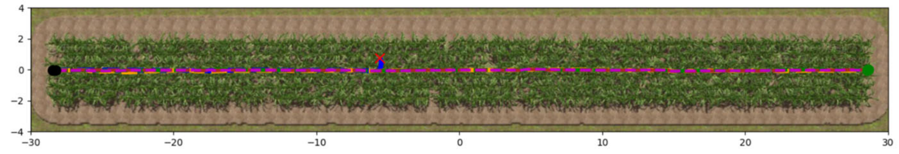
4.4 Field Validation

Following the validation of the proposed method through simulation, we proceeded to conduct field validation using an actual robotic platform. We utilized the facilities of the Illinois Autonomous Farm at the University of Illinois Urbana-Champaign, which features rows with varying shapes, distances between crops and sizes of the rows.

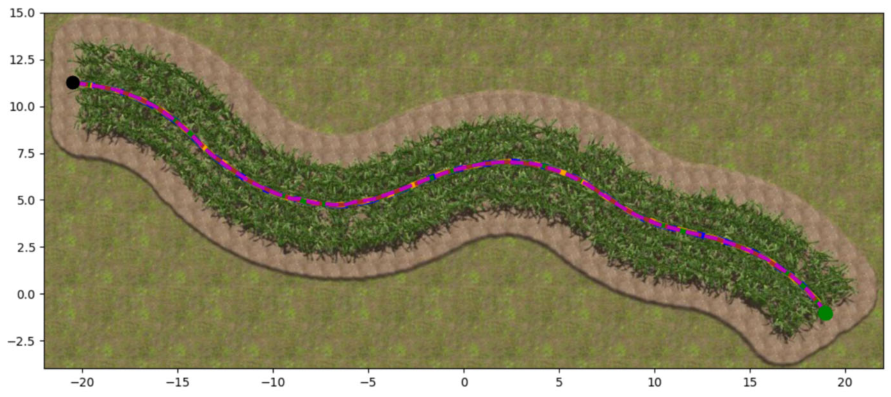
In the outdoor setting, we employed a Real-Time Kinematic (RTK) system as the ground truth for analyzing the trajectories, as our system does not require a localization module to navigate through the crops.



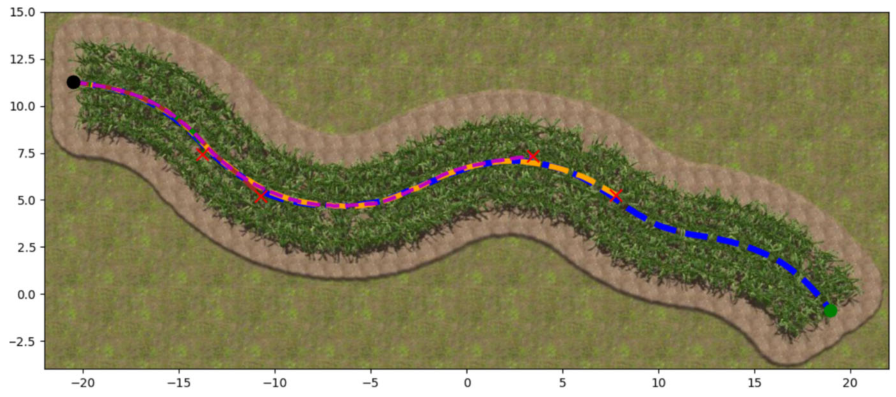
(a) Experiment with CROW in a straight-row scenario.



(b) Experiment with Higuti et al. [?] in a straight-row scenario.



(c) Experiment with CROW in a curved-row scenario.



(d) Experiment with Higuti et al. [?] in a curved-row scenario.

Fig. 13 Comparison between CROW and Higuti et al. [3] on a straight and curved crop row. In this experiment, the methods run five times in the same crop row. The plot shows the navigated path with collision locations

Table 4 Comparison between CROW and Higuti et al. [3] for straight and curved crop rows

Run #	Straight Row				Curved Row			
	CROW		Higuti et al. [3]		CROW		Higuti et al. [3]	
	Coll.	Distance [m]	Coll.	Distance [m]	Coll.	Distance [m]	Coll.	Distance [m]
1	–	57.50	✓	22.83	–	44.00	–	44.00
2	–	57.50	–	57.50	–	44.00	✓	31.06
3	–	57.50	–	57.50	–	44.00	✓	8.28
4	–	57.50	–	57.50	–	44.00	✓	11.87
5	–	57.50	–	57.50	–	44.00	✓	26.42
Average	**0**	**57.5**	**20%**	**50.56**	**0.0**	**44.0**	**80%**	**24.326**

Continuing, we conducted 6 experiments across different rows measuring 115 meters, during which we recorded the number of collisions and the time taken to complete each run. For each collision, the controller was halted, and the robot's locomotion resumed once it was realigned within the row. Finally, the numerical results can be seen in Table 5.

Additionally, some of the trajectories used to generate these results, based on RTK-recorded positions, are shown in Fig. 14 along with their corresponding collisions and start points. We excluded some trajectories to avoid overcrowding the GPS plot, as repeated rows made it difficult to interpret.

These results demonstrate that our method successfully navigates real crop rows without the need to train the perception module on real-world data, achieving an average distance of 34.5 meters per collision, with 3±3 collisions over the runs.

However, real-world noise, such as weeds within the rows and other natural obstacles, introduces challenges in interpreting the LiDAR data for detecting crop lines. These obstacles were less prevalent in runs 3 and 4, as observed in the front camera footage from those experiments, which also had fewer collisions. Thus, we can establish a direct correlation between the difficulty of each row, i.e., the number of natural features in the under-canopy environment, and the number of collisions experienced by CROW.

Table 5 Field Validation Results

Run #	Coll.	Av. Dist. p/ Coll. [m]	Traj. Time [s]
1	5	23.00	235
2	4	28.75	245
3	0	–	155
4	1	115.0	172
5	4	28.75	252
6	6	19.17	278
Average	**3.33**	**34.5**	**222.83**

Furthermore, given that our system struggles in these challenging conditions, it is possible to integrate a recovery module, as implemented in Gasparino et al. [12]. This addition could enable the system to navigate under-canopy environments more effectively, thereby enhancing its overall performance.

It is important to note that the satellite images shown in Fig. 14 are not aligned with the exact dates of our experiments. As a result, the auxiliary image is provided to approximate the appearance of the crops during the experimental period.

5 Conclusion

We presented CROW, a row-following navigation system designed for compact robots, validated on the Terrasentia Platform within under-canopy environments. Our system uses a LiDAR-based self-supervised Deep learning perception, capable of identify across various crop types and detecting crop rows amidst fluctuating noise levels. This perception data informs the generation of waypoints for the robot, which is controlled by an MPC framework that integrates an error-tracking model with a constrained iLQR solver.

Our method was evaluated in both simulated and real-world environments, demonstrating robust performance under diverse conditions. In simulation, CROW excelled in navigating curved crop rows, a challenge that many contemporary approaches struggle with, achieving successful navigation without any collisions.

In real-world tests, however, a performance gap emerged due to the presence of noise, such as weeds and other natural obstacles within the crop rows. This noise introduced challenges in interpreting the LiDAR data, leading to more collisions in noisier rows. Despite these challenges, CROW was able to navigate real crop rows without requiring training on real-world data, showcasing its adaptability and strength.

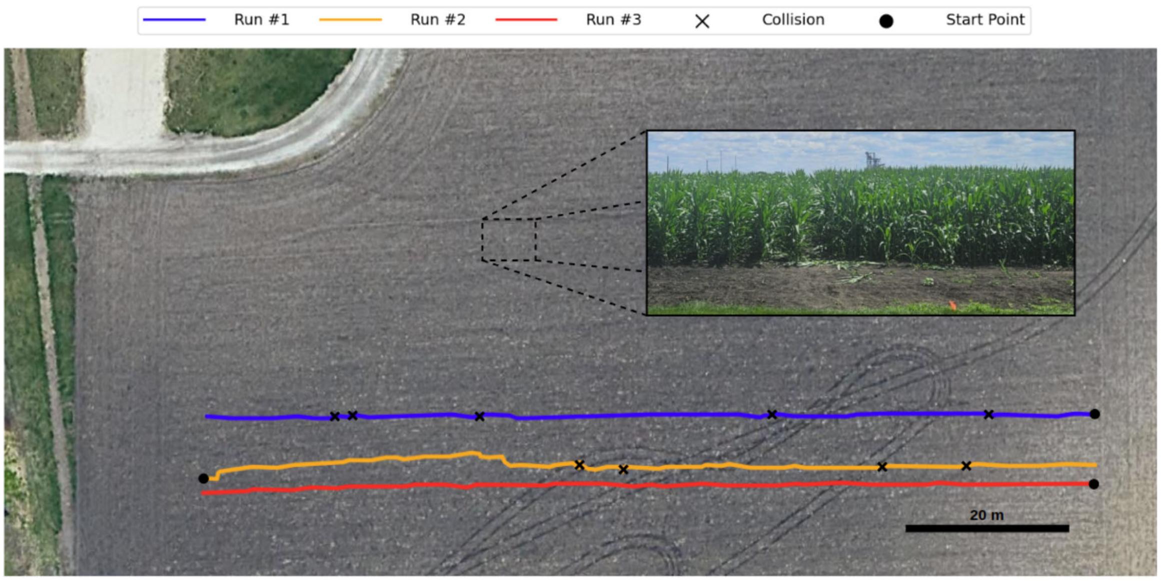


Fig. 14 GPS trajectories of CROW at the Illinois Autonomous Farm. The accompanying image depicts the plantations on the date of the experiments

Notably, the system achieved an average of 34.5 meters per collision, with 3 ± 3 collisions across our experiments.

Acknowledgements This work was supported by São Paulo Research Foundation (FAPESP) grant no. 2022/08330-9, 2022/03339-8 and 2022/08433-2. Brazilian National Research Council (CNPq) grants no. 308092/2020-1.

Author Contributions All authors contributed to the methodology, development and validation of this study. Controller and System Idealization: Francisco Affonso. Perception System: Felipe Andrade G. Tommaselli. Experimental Setup: Gianluca Capezzuto. Supervision: Mateus V. Gasparino, Girish Chowdhary and Marcelo Becker

Funding This work was supported by São Paulo Research Foundation (FAPESP) grant no. 2022/08330-9, 2022/03339-8 and 2022/08433-2.

Data Availability The artificial dataset used to train the model is available online: https://drive.google.com/drive/folders/1IeAhgs2SlV-Fgol1CqfxPrg_Yv2mBH2O?usp=sharing. This folder also includes the corresponding labels for the dataset and the trained model, which were used in the presented results.

Code availability All this code work was synthesized in the official CROW repository on GitHub: https://github.com/faffonso/CROW.

Declarations

Conflicts of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethics approval Not applicable.

Consent to participate Informed consent was obtained from all participants.

Consent for publication Participant consented to the submission of this article.

References

1. English, A., Ross, P., Ball, D., Corke, P.: Vision based guidance for robot navigation in agriculture, 1693–1698 (2014) https://doi.org/10.1109/ICRA.2014.6907079
2. Emmi, L., Le Flécher, E., Cadenat, V., Devy, M.: A hybrid representation of the environment to improve autonomous navigation of mobile robots in agriculture. Precision Agric. **22**, 1–26 (2021). https://doi.org/10.1007/s11119-020-09773-9
3. Higuti, V., Velasquez, A., Magalhaes, D., Becker, M., Chowdhary, G.: Under canopy light detection and ranging-based autonomous navigation. Journal of Field Robotics. **36**(3), 547–567 (2019). https://doi.org/10.1002/rob.21852
4. Mueller-Sim, T., Jenkins, M., Abel, J., Kantor, G.: The robotanist: A ground-based agricultural robot for high-throughput crop phenotyping. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 3634–3639 (2017). https://doi.org/10.1109/ICRA.2017.7989418
5. Velasquez, A., Higuti, V., Valverde Gasparino, M., V Sivakumar, A.N., Becker, M., Chowdhary, G.: Multi-sensor fusion based robust row following for compact agricultural robots. Field Robotics. **2**, 1291–1319 (2022) https://doi.org/10.55417/fr.2022043

6. McAllister, W., Whitman, J., Varghese, J., Axelrod, A., Davis, A., Chowdhary, G.: Agbots 2.0: Weeding denser fields with fewer robots. In: Toussaint, M., Bicchi, A., Hermans, T. (eds.) Robotics. Robotics: Science and Systems. MIT Press Journals, United States (2020). https://doi.org/10.15607/RSS.2020.XVI.062
7. Shamshiri, R.R., Weltzien, C., Hameed, I.A., Yule, I.J., Grift, T.E., Balasundram, S.K., et al.: Research and development in agricultural robotics: A perspective of digital farming. International Journal of Agricultural & Biological Engineering. **11**(4), 1–14 (2018)
8. Pinto, F.A., Tommaselli, F.A.G., Gasparino, M.V., Becker, M.: Navigating with finesse: Leveraging neural network-based lidar perception and ilqr control for intelligent agriculture robotics. In: 2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE), pp. 502–507 (2023). https://doi.org/10.1109/LARS/SBR/WRE59448.2023.10332981
9. Sivakumar, A.N., Modi, S., Gasparino, M.V., Ellis, C., Velasquez, A.E.B., Chowdhary, G., Gupta, S.: Learned Visual Navigation for Under-Canopy Agricultural Robots. Preprint at https://arxiv.org/abs/2107.02792 (2021)
10. Kayacan, E., Young, S.N., Peschel, J.M., Chowdhary, G.: High-precision control of tracked field robots in the presence of unknown traction coefficients. Journal of Field Robotics. **35**(7), 1050–1062 (2018). https://doi.org/10.1002/rob.21794
11. Bergerman, M., Maeta, S.M., Zhang, J., Freitas, G.M., Hamner, B., Singh, S., Kantor, G.: Robot farmers: Autonomous orchard vehicles help tree fruit production. IEEE Robotics & Automation Magazine. **22**(1), 54–63 (2015). https://doi.org/10.1109/MRA.2014.2369292
12. Gasparino, M.V., Higuti, V.A.H., Sivakumar, A.N., Velasquez, A.E.B., Becker, M., Chowdhary, G.: Cropnav: a framework for autonomous navigation in real farms. In: 2023 IEEE International Conference on Robotics and Automation (ICRA), pp. 11824–11830 (2023). https://doi.org/10.1109/ICRA48891.2023.10160990
13. Cantelli, L., Bonaccorso, F., Longo, D., Melita, C.D., Schillaci, G., Muscato, G.: A small versatile electrical robot for autonomous spraying in agriculture. AgriEngineering. **1**(3), 391–402 (2019)
14. Plancher, B., Manchester, Z., Kuindersma, S.: Constrained unscented dynamic programming. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5674–5680 (2017). https://doi.org/10.1109/IROS.2017.8206457
15. Howell, T.A., Jackson, B.E., Manchester, Z.: Altro: A fast solver for constrained trajectory optimization. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 7674–7679 (2019). https://doi.org/10.1109/IROS40897.2019.8967788
16. Gill, P.E., Murray, W., Saunders, M.A.: Snopt: An sqp algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005). https://doi.org/10.1137/S0036144504446096
17. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006). https://doi.org/10.1007/s10107-004-0559-y
18. Aulinas, J., Petillot, Y., Salvi, J., Llado, X.: The slam problem: a survey **184**, 363–371 (2008). https://doi.org/10.3233/978-1-58603-925-7-363
19. Groves, P.D.: Principles of gnss, inertial, and multisensor integrated navigation systems, 2nd edition [book review]. IEEE Aerospace and Electronic Systems Magazine. **30**(2), 26–27 (2015) https://doi.org/10.1109/MAES.2014.14110
20. Rovira-Más, F., Chatterjee, I., Sáiz-Rubio, V.: The role of gnss in the navigation strategies of cost-effective agricultural robots. Comput. Electron. Agric. **112**, 172–183 (2015). https://doi.org/10.1016/j.compag.2014.12.017. Precision Agriculture
21. Gasparino, M.V., Higuti, V.A.H., Velasquez, A.E.B., Becker, M.: Improved localization in a corn crop row using a rotated laser rangefinder for three-dimensional data acquisition. J. Braz. Soc. Mech. Sci. Eng. **42**(11), 592 (2020). https://doi.org/10.1007/s40430-020-02673-z
22. Gu, Y., Li, Z., Zhang, Z., Li, J., Chen, L.: Path tracking control of field information-collecting robot based on improved convolutional neural network algorithm. Sensors. **20**(3), 797 (2020). https://doi.org/10.3390/s20030797
23. Gai, J., Xiang, L., Tang, L.: Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle. Comput. Electron. Agric. **188**, 106301 (2021). https://doi.org/10.1016/j.compag.2021.106301
24. Mur-Artal, R., Montiel, J.M.M., Tardós, J.D.: Orb-slam: A versatile and accurate monocular slam system. IEEE Trans. Rob. **31**(5), 1147–1163 (2015). https://doi.org/10.1109/TRO.2015.2463671
25. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multiview stereopsis. IEEE Trans. Pattern Anal. Mach. Intell. **32**(8), 1362–1376 (2010). https://doi.org/10.1109/TPAMI.2009.161
26. Xue, J., Zhang, L., Grift, T.E.: Variable field-of-view machine vision based row guidance of an agricultural robot. Computers and Electronics in Agriculture. **84**, 85–91 (2012) 10.1016/j.compag.2012.02.009
27. Malavazi, F.B.P., Guyonneau, R., Fasquel, J.-B., Lagrange, S., Mercier, F.: Lidar-only based navigation algorithm for an autonomous agricultural robot. Computers and Electronics in Agriculture. **154**, 71–79 (2018) 10.1016/j.compag.2018.08.034
28. Corno, M., Furioli, S., Cesana, P., Savaresi, S.M.: Adaptive ultrasound-based tractor localization for semi-autonomous vineyard operations. Agronomy **11**(2), 287 (2021). https://doi.org/10.3390/agronomy11020287
29. Xaud, M.F.S., Leite, A.C., From, P.J.: Thermal image based navigation system for skid-steering mobile robots in sugarcane crops. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 1808–1814 (2019). https://doi.org/10.1109/ICRA.2019.8794354
30. Xu, R., Li, C.: A review of high-throughput field phenotyping systems: Focusing on ground robots. Plant Phenomics. **2022** (2022) https://doi.org/10.34133/2022/9760269
31. Bai, Y., Zhang, B., Xu, N., Zhou, J., Shi, J., Diao, Z.: Vision-based navigation and guidance for agricultural autonomous vehicles and robots: A review. Comput. Electron. Agric. **205**, 107584 (2023). https://doi.org/10.1016/j.compag.2022.107584
32. Kim, K., Deb, A., Cappelleri, D.J.: P-agbot: In-row & under-canopy agricultural robot for monitoring and physical sampling. IEEE Robotics and Automation Letters. **7**(3), 7942–7949 (2022). https://doi.org/10.1109/LRA.2022.3187275
33. Wang, P.: Research on comparison of lidar and camera in autonomous driving. J. Phys: Conf. Ser. **2093**(1), 012032 (2021). https://doi.org/10.1088/1742-6596/2093/1/012032
34. Nellithimaru, A.K., Kantor, G.A.: Rols : Robust object-level slam for grape counting. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 2648–2656 (2019). https://doi.org/10.1109/CVPRW.2019.00321
35. Zhao, W., Wang, X., Qi, B., Runge, T.: Ground-level mapping and navigating for agriculture based on iot and computer vision. IEEE Access. **8**, 221975–221985 (2020). https://doi.org/10.1109/ACCESS.2020.3043662
36. Chen, S., Zhou, B., Jiang, C., Xue, W., Li, Q.: A lidar/visual slam backend with loop closure detection and graph optimization. Remote Sensing. **13**(14), 2720 (2021). https://doi.org/10.3390/rs13142720
37. Oliveira, L.F.P., Moreira, A.P., Silva, M.F.: Advances in agriculture robotics: A state-of-the-art review and challenges ahead. Robotics **10**(2), 52 (2021). https://doi.org/10.3390/robotics10020052
38. V Sivakumar, A.N., Valverde Gasparino, M., McGuire, M., Higuti, V., Akcal, M., Chowdhary, G.: Demonstrating cropfollow++:

Robust under-canopy navigation with keypoints. (2024). https://doi.org/10.15607/RSS.2024.XX.023
39. Huang, P., Zhu, L., Zhang, Z., Yang, C.: An end-to-end learning-based row-following system for an agricultural robot in structured apple orchards. Math. Probl. Eng. **2021**(1), 6221119 (2021). https://doi.org/10.1155/2021/6221119
40. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR. **abs/1704.04861** (2017) arXiv:1704.04861
41. Lu, Y., Chowdhery, A., Kandula, S., Chaudhuri, S.: Accelerating machine learning inference with probabilistic predicates. SIGMOD '18, pp. 1493–1508. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3183713.3183751
42. Kar, A., Prakash, A., Liu, M.-Y., Cameracci, E., Yuan, J., Rusiniak, M., Acuna, D., Torralba, A., Fidler, S.: Meta-sim: Learning to generate synthetic datasets. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
43. Song, H., Kim, M., Park, D., Shin, Y., Lee, J.-G.: Learning from noisy labels with deep neural networks: A survey. IEEE Transactions on Neural Networks and Learning Systems. **34**(11), 8135–8153 (2023). https://doi.org/10.1109/TNNLS.2022.3152527
44. Kim, J., Picek, S., Heuser, A., Bhasin, S., Hanjalic, A.: Make some noise. unleashing the power of convolutional neural networks for profiled side-channel analysis. IACR Transactions on Cryptographic Hardware and Embedded Systems. **2019**(3), 148–179 (2019) https://doi.org/10.13154/tches.v2019.i3.148-179
45. Goldberger, J., Ben-Reuven, E.: Training deep neural-networks using a noise adaptation layer. In: International Conference on Learning Representations (2017). https://openreview.net/forum?id=H12GRgcxg
46. Kayacan, E., Chowdhary, G.: Tracking error learning control for precise mobile robot path tracking in outdoor environment. Journal of Intelligent & Robotic Systems. **95**, 975–986 (2019). https://doi.org/10.1007/s10846-018-0916-3
47. González-Sierra, J., Aranda-Bricaire, E., Hernández-Martínez, E.G.: Trajectory tracking strategies with singularities avoidance for groups of unicycle-type robots. IFAC Proceedings Volumes. **44**(1), 5926–5931 (2011). https://doi.org/10.3182/20110828-6-IT-1002.01703. 18th IFAC World Congress
48. Ma, J., Cheng, Z., Zhang, X., Lin, Z., Lewis, F.L., Lee, T.H.: Local learning enabled iterative linear quadratic regulator for constrained trajectory planning. IEEE Transactions on Neural Networks and Learning Systems. **34**(9), 5354–5365 (2023). https://doi.org/10.1109/TNNLS.2022.3165846
49. Bemporad, A., Morari, M., Dua, V., Pistikopoulos, E.N.: The explicit linear quadratic regulator for constrained systems. Automatica **38**(1), 3–20 (2002). https://doi.org/10.1016/S0005-1098(01)00174-1
50. Forsgren, A., Gill, P.E., Wright, M.H.: Interior methods for nonlinear optimization. SIAM Rev. **44**(4), 525–597 (2002). https://doi.org/10.1137/S0036144502414942
51. Koenig, N., Howard, A.: Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), vol. 3, pp. 2149–21543 (2004). https://doi.org/10.1109/IROS.2004.1389727
52. Barthel, J., Bier, J., Friedel, T., Ruigrok, T., Tuschla, L., Essen, R.: Virtual Maize Field. https://github.com/FieldRobotEvent/virtual_maize_field

Francisco Affonso Undergraduate student in Mechatronics Engineering at the University of São Paulo, São Carlos School of Engineering. Member of LabRoM since 2022, where he began his Scientific Initiation project at the São Paulo Research Foundation (FAPESP) on Control Systems.

Felipe Andrade G. Tommaselli Undergraduate student in Electrical Engineering with emphasis on Computing at the University of São Paulo, São Carlos School of Engineering. Member of LabRoM since 2022, where he began his Scientific Initiation project at the São Paulo Research Foundation (FAPESP) on the Perception system outlined in this document.

Gianluca Capezzuto Undergraduate student in Electrical Engineering with emphasis on Computing at the University of São Paulo, São Carlos School of Engineering. Member of LabRoM since 2022, where he began his Scientific Initiation project at the São Paulo Research Foundation (FAPESP) on Ablative studies in Neural Networks.

Mateus V. Gasparino Ph.D. student at the Computer Science Department, University of Illinois at Urbana-Champaign, USA. He was awarded the M.Sc. degree in mechanical engineering and Bachelor's degree in mechatronics engineering from the University of Sao Paulo, Brazil. He is currently a graduate research assistant at the Distributed Autonomous Systems Laboratory (DASLab), and his research interests include perception, mapping, control, and learning for robots in unstructured outdoor environments.

Girish Chowdhary Associate professor and Donald Biggar Willet Faculty Fellow at the University of Illinois at Urbana-Champaign. He is the director of the Field Robotics Engineering and Science Hub (FRESH) at UIUC and the Director of the USDA/NIFA Farm of the Future. Girish holds a joint appointment with Agricultural and Biological Engineering and Computer Science, he is a member of the UIUC Coordinated Science Lab, and holds affiliate appointments in Aerospace Engineering and Electrical Engineering. He holds a PhD (2010) from Georgia Institute of Technology in Aerospace Engineering. He was a postdoc at the Laboratory for Information and Decision Systems (LIDS) of the Massachusetts Institute of Technology (2011-2013), and an assistant professor at Oklahoma State University (2013-2016). He also worked with the German Aerospace Center's (DLR's) Institute of Flight Systems for around three years (2003-2006). He is the winner of the Air Force Young Investigator Award, and several best paper awards, including a best systems paper award at RSS 2018 for his recent work on the agricultural robot TerraSentia.

Marcelo Becker Received his B.Sc. Mechanical Engineer (ME) degree with emphasis on Mechatronics in 1993 at University of São Paulo (USP), Brazil. He received his M.Sc. ME and D.Sc. ME degrees, respectively, in 1997 and 2000 at the State University of Campinas (Unicamp), Brazil. During his D.Sc. studies he spent 8 months as a guest student at the Institute of Robotics (IfR) - Swiss Federal Institute of Technology, Zurich (ETHZ). At that time he was involved in researches on obstacle avoidance and map building procedures for indoor mobile robots. From August 2005 until July 2006 he did a Sabbatical at the Autonomous System Lab (ASL) - Swiss Federal Institute of Technology, Lausanne (EPFL). There he was involved in researches on obstacle avoidance for indoor and outdoor mobile robots. From 2001 to 2008, he served as an associate professor at Pontifical Catholic University of Minas Gerais (PUC Minas) in Brazil. During 2002-2005, he also held the position of co-head of the Mechatronics Engineering Department and the Robotics and Automation Group (GEAR) at PUC Minas. Since 2008, he has been a Professor at the University of São Paulo (EESCUSP). As of 2022, he has taken on the role of coordinator at the Robotics Center at USP. His research work has been published in various conferences and journals, focusing on mechanical design and mobile robotics. While his research interests are diverse, his main areas of focus include mobile robots for agriculture, inspection robots for industry, design methodologies and tools, mechanical design applied to robots, and mechatronics.

Francisco Affonso[1] · **Felipe Andrade G. Tommaselli**[1] · **Gianluca Capezzuto**[1] · **Mateus V. Gasparino**[2] · **Girish Chowdhary**[2] · **Marcelo Becker**[1]

✉ Francisco Affonso
faffonso@usp.br

✉ Felipe Andrade G. Tommaselli
f.tommaselli@usp.br

✉ Marcelo Becker
becker@sc.usp.br

Gianluca Capezzuto
gianlucacapezzuto@usp.br

Mateus V. Gasparino
mvalve2@illinois.edu

Girish Chowdhary
girishc@illinois.edu

1 Robotics Lab. Mobile Robotics Group - LabRoM, Mechanical Engineering Department, University of São Paulo (EESC - USP), São Paulo, Brazil

2 Field Robotics Engineering and Science Hub (FRESH), Illinois Autonomous Farm, University of Illinois at Urbana-Champaign (UIUC), Champaign, IL, USA