IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Navigating Narrow Spaces: A Comprehensive Framework for Agricultural Robots

Geesara Kulathunga[2,1*], Abdurrahman Yilmaz[1*], Zhuoling Huang[1], Ibrahim Hroob[1], Jaspreet Singh[1], Leonardo Guevara[1], Grzegorz Cielniak[1] and Marc Hanheide[1]

***Abstract*— Navigating within narrow spaces is a fundamental challenge in robotics, requiring precise localisation, localisation error recovery, dynamic path planning, and adaptive control for effective manoeuvring. This paper presents a modular and perception-driven navigation framework designed for constrained environments, focusing primarily on agricultural applications. The proposed method integrates a multi-step point cloud processing pipeline for robust local perception, including pole detection, boundary line estimation, and trajectory refinement to ensure safe and precise traversal by refining initial trajectories based on detected environmental constraints and dynamically adapting to kinematic limitations. Experimental validation in a real strawberry polytunnel demonstrates superior trajectory accuracy and control stability compared to state-of-the-art navigators, achieving an average lateral deviation of 0.08 ± 0.01 m. The adaptive trajectory tracking and regulated pure pursuit control of the framework contribute to consistent navigation, even under increased velocity constraints, outperforming the resilient timed elastic band (RTEB) and model predictive path integral (MPPI) methods. This modular and generalisable framework offers significant potential for advancing autonomous navigation in narrow-space applications.**

I. Introduction

Navigating within narrow spaces is a critical and fundamental challenge in various robotics applications, requiring precise localisation, robust path planning, and dynamic control to operate effectively in constrained environments [1]. These scenarios often involve limited spatial manoeuvrability, where robots must navigate through constrained pathways, crop rows in agricultural fields [2], densely packed environments, or tight industrial storage aisles [3]. Navigation becomes even more challenging due to the localisation issues that may arise from weak GNSS signals [4], the dynamic or cluttered nature of the environment [5], or the reliance on low-accuracy global localisation methods [6].

This paper presents a multi-step navigation framework designed for narrow-space traversal, with a primary demonstration in agricultural settings. The proposed approach addresses limitations in global localisation with a robust point cloud-based local perception system and a trajectory refinement strategy to ensure safe and precise navigation, even for non-holonomic robots, e.g. car-like ground vehicles.

[1]The authors are with the Lincoln Centre for Autonomous Systems (LCAS), University of Lincoln, LN6 7TS, UK {ayilmaz, zhuang, ihroob, jasingh, lguevara, gcielniak, mhanheide}@lincoln.ac.uk, [2]AURIGA AI LTD, W1S 3AT, London, UK {geesara.k}@auriga.xyz

*Both authors contributed equally.

As illustrated in Fig. 1, the proposed framework processes raw point cloud data to identify key elements of the agricultural infrastructure, including poles that mark the left and right boundaries, intermediate goals derived from the initial reference goal, and a refined trajectory aligned with the centreline. This refined trajectory maintains proximity to the detected poles while continuously adapting to environmental constraints. The robot is guided smoothly between crop rows or other constrained pathways, enabling accurate and safe navigation.

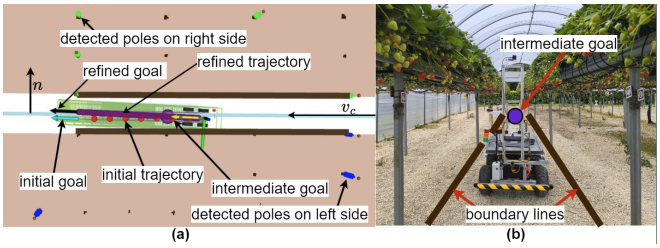

Fig. 1. Narrow space navigation strategy for an autonomous robot in agricultural environments: (a) A top-down view illustrating key elements of the navigation process. (b) The real-world scenario depicts the robot navigating between plant rows

The **contributions** of the study are as follows: 1). **Perception-driven navigation framework:** Development of a modular framework for narrow-space traversal, integrating real-time local perception and adaptive path refinement to enhance navigation robustness under low localisation accuracy. 2). **Point cloud-based environmental perception and boundary estimation:** A multi-step pipeline that processes raw point cloud data for enhanced scene understanding. This includes **ground removal** [7], [8], **elevation filtering**, **outlier filtering** [9], and **clustering** to extract structural elements. An efficient **pole detection algorithm** combining principal component analysis (PCA)-based orientation analysis and bounding box constraints—enables robust boundary estimation even under occlusions and sensor noise. 3). **Trajectory refinement and adaptive tracking:** A non-linear **optimisation-based** path refinement method that ensures adherence to environmental constraints. This is coupled with a **regulated pure pursuit controller** [10], ensuring kinematically feasible, dynamically stable path tracking even under increased velocity constraints.

II. Related Works

Autonomous navigation in agricultural environments, particularly in horticulture, presents significant challenges due

TABLE I

COMPARISON OF POLE DETECTION METHODS IN AGRICULTURAL SETTINGS

Method	Detection Accuracy	Robustness to Occlusion	Adaptability to Agricultural Environments
RANSAC [16]	High for clean data	Low (fails with partial occlusion)	Low (struggles with irregular terrain)
Clustering [17]	Moderate (sensitive to point density)	Moderate (fails in sparse point clouds)	Moderate (evenly spaced structures)
Vision [18]	High in clear visual conditions	Moderate (affected by lighting, shadows)	Low (issues with dust and foliage)
Proposed	High (temporal and spatial filtering)	High (effective with partial occlusion)	High (variable terrain and crop row shapes)

to narrow spacing between crop rows, reduced measurement accuracy of RTK-GNSS systems in covered areas (e.g., polytunnels and glasshouses), and noisy LiDAR measurements caused by environmental clutter [11]. Although advancements in visual sensors, GNSS integration, and machine learning have been proposed for crop row detection, these systems often depend heavily on precise sensor calibration and stable environmental conditions (e.g. changing light conditions), making them less effective in irregular or cluttered spaces [12]. Additionally, vision-based systems are sensitive to lighting variations, which may reduce their reliability in low-light or shadowed conditions. Table I summarises common pole detection methods in agri-robotics, highlighting their accuracy, robustness to occlusion, and adaptability.

Several robust algorithms for crop row detection and navigation using affordable cameras and GNSS systems have shown the potential to handle field variations, like growth stages, lighting conditions, and weed densities [13], [14]. These variations are inherent in agricultural fields and present significant navigation challenges, particularly in narrow, confined spaces between crop rows. Gai et al. [15] introduced a navigation method for dense canopy environments where GNSS signals are weak or unavailable. However, many of these approaches were designed for open spaces and lack robust planning and tracking for precise trajectory following in the constrained environments typically found in agricultural fields.

Autonomous navigation in structured environments, such as urban roads, has seen significant progress through advancements in computer vision and deep learning [19]–[21]. Lane detection algorithms in such contexts leverage high-contrast road markings, strong geometric priors, and dense semantic information from multimodal sensors like cameras and LiDAR [22]. However, these assumptions do not hold in agricultural environments, where the terrain is uneven, foliage can occlude key features, and lane-like structures (e.g., rows of trees or crops) are often irregular or discontinuous.

The reviewed approaches illustrate significant advancements in narrow-space navigation across agricultural and other robotics applications. However, limitations remain in handling localisation uncertainties and complex path-following scenarios. Existing methods often trade off adaptability and precision, underscoring the need for a more comprehensive and efficient navigation framework. Motivated by these gaps, this work introduces a robust and efficient navigation strategy integrating path refinement, boundary constraint generation, and precise kinematic control, specifically designed for navigating constrained agricultural rows and other narrow environments.

III. METHODOLOGY

The primary objective of this work is to develop a framework for autonomous navigation and operational planning in narrow semi-structured environments, such as orchard rows or vineyard corridors, where vertical features, e.g., tree trunks, posts, or similar structures tend, to be loosely aligned along either side of the robot's path. These environments are assumed to be relatively even in terrain and structurally consistent, enabling effective ground segmentation and feature extraction. The system tackles several key challenges, illustrated in Fig. 1(c): (1) constructing a reliable representation of the environment, (2) segmenting and classifying significant features for safe and efficient navigation, and (3) adapting operational constraints to real-time decision-making.

To achieve this, a multi-step point cloud processing pipeline is employed, using data captured from sensors such as LiDAR or depth cameras to construct a spatial-temporal voxel map [7]. The process begins with ground removal using Patchwork++ [8], a critical step for filtering out ground points and isolating meaningful environmental features. As Livox LiDARs [23] are utilised in this work providing affordable sensing but often introducing substantial false positives (e.g., Fig. 2(a)) some points may be misclassified as ground. To mitigate this, elevation filtering is applied to retain points within a specific vertical range, focusing attention on objects of interest and refining the dataset to preserve only accurate non-ground points.

At system initialisation, the robot is assumed to have a rough pose estimate, either from onboard GNSS-IMU localisation or manual input, with the initial heading approximately aligned with the general direction of travel. Sensor extrinsics are pre-calibrated, and all data streams, including point clouds and pose estimates, are assumed to be reasonably synchronised in time, although minor delays or offsets are tolerable within the pipeline.

A. Problem Formulation

At each time step t, the system receives a point cloud $\mathcal{P}_t \subset \mathbb{R}^3$ and the robot pose $\mathbf{x}_t = [x, y, \theta]^\top$, from which the heading vector $\mathbf{v}_c = [\cos(\theta), \sin(\theta)]^\top$ is derived. The goal is to detect vertical pole-like structures in the environment and fit a centerline that can guide the robot safely and efficiently. To begin, the point cloud is cropped to a horizontal field of view bounded by angles $\theta_{\min}$ and $\theta_{\max}$, and filtered along the vertical axis using a lower threshold t_z. Each cluster

$\mathcal{C}_i$ is evaluated using PCA, and a candidate is retained if its dominant axis is aligned with the vertical axis within a threshold ω_θ, its spatial extent is within a maximum pole width w_{pole}, and it passes a Z-score-based outlier test with threshold z_m. The centroids of such filtered clusters form a set of pole candidates $\mathcal{P}_{\text{pole}} \subset \mathbb{R}^2$. Each pole $\mathbf{p}_i$ is classified as left or right of the robot based on the sign of $\mathbf{v}_c \times (\mathbf{p}_i - \mathbf{p}_t)$, and retained only if its perpendicular distance to $\mathbf{v}_c$ is less than r_w, and its directional deviation β_i from $\mathbf{v}_c$ is less than ω_β. After side segmentation and filtering, we obtain $\mathbf{p}_{ls}$ and $\mathbf{p}_{rs}$, representing valid left and right pole sets, respectively. The final trajectory refinement is to minimize the alignment between $\tau_{refined}$ and τ_{init}, and all poles satisfy the filtering constraints $\theta_i < \omega_\theta$, $\beta_i < \omega_\beta$, and $d_\perp < r_w$. This formulation yields a robust and geometry-aware method for estimating the centerline in structured environments with incomplete or noisy data.

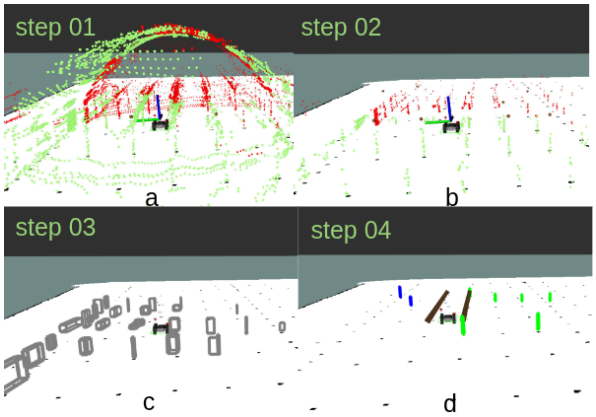

Fig. 2. Multi-step point cloud processing and boundary line estimation pipeline: (a) Raw point cloud with back (green) and front (red) points, (b) Low-height points after ground removal and elevation filtering, (c) Object clustering with bounding boxes, (d) Boundary line estimation with poles (green and blue) and estimated line (brown) for in-row navigation.

B. Point cloud cluster extraction

The non-ground point cloud data (Fig. 2(b)) is first downsampled using a voxel grid filter with a voxel size of 1 cm, reducing point density to enhance computational efficiency. Enhanced Euclidean cluster extraction [24] is then applied—augmented with multi-threading, pole detection, global path integration, and advanced filtering—to segment the point cloud into spatially coherent clusters. For each resulting cluster $\mathcal{C}$, Algorithm 1 performs two filtering steps. First, Principal Component Analysis (PCA) [25] is used to reorient the cluster along its principal axes and detect structural outliers based on deviation from the dominant orientation. Second, Z-score analysis [9] identifies statistical outliers by comparing point deviations from the cluster mean. These steps collectively remove noise and non-structural elements, yielding refined, non-ground clusters $\bar{\mathbf{C}}$, as described in Algorithm 1.

C. Pole detection from clusters

The cleaned point cloud is further clustered using voxel grid clustering to generate bounding boxes and identify segmented poles, defining spatial limits for navigation (see Fig. 2(c)). Pole identification involves bounding box analysis of clusters processed by Algorithm 1. For each cluster $\bar{\mathcal{C}}_i$, an axis-aligned bounding box is computed using the minimum and maximum points along x, y, and z axes $\bar{\mathcal{C}}^{min}, \bar{\mathcal{C}}^{max}$. Clusters with width exceeding a predefined threshold, w_{pole}, in the x- and y-direction ($\Delta x = |\bar{\mathcal{C}}^{max} - \bar{\mathcal{C}}^{min}|_x$, $\Delta y = |\bar{\mathcal{C}}^{max} - \bar{\mathcal{C}}^{min}|_y$) are rejected to retain pole-like structures.

For dimensionally valid clusters, PCA is used to evaluate orientation by computing the covariance matrix:

$$\Sigma = \frac{1}{|\bar{\mathcal{C}}|}\sum_{c\in\bar{\mathcal{C}}}(c-\mu)(c-\mu)^\top, \quad \mu = \frac{1}{|\bar{\mathcal{C}}|}\sum_{c\in\bar{\mathcal{C}}} c$$

where μ is the cluster centroid. Eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ and eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ are derived, with $\lambda_1 \geq \lambda_2 \geq \lambda_3$. The principal component $\mathbf{v}_1$ indicates the direction of maximum variance.

Algorithm 1 Point cloud cluster filtering with PCA

Require: $\mathbf{C} \leftarrow \{\mathcal{C}_i\}_{i=1}^N$, $\theta_{\min}, \theta_{\max}, t_z, z_m$ ▷ clusters and hyperparameters
Ensure: $\bar{\mathbf{C}}$ ▷ filtered clusters

procedure CLUSTER FILTERING
  for cluster $\mathcal{C}_i \in \mathbf{C}$ **do**
    $\lambda_1, \lambda_2, \lambda_3 \leftarrow \text{PCA}(\mathcal{C}_i)$ ▷ compute eigenvalues
    $\theta \leftarrow \arccos\left(\frac{\mathbf{v}_1^\top \mathbf{z}}{\|\mathbf{v}_1\|\|\mathbf{z}\|}\right)$ ▷ angle with vertical axis
    if $\theta_{\min} \leq \theta \leq \theta_{\max}$ **then**
      add $\mathcal{C}_i$ to $\bar{\mathbf{C}}$ ▷ retain cluster
      continue
    end if
    $\mu_i, \sigma_i \leftarrow \text{mean}(\mathcal{C}_i)$, $\text{cov}(\mathcal{C}_i)$ ▷ mean and variance
    for $c_i \in \mathcal{C}_i$ **do**
      $z_{score} \leftarrow \frac{|c_i - \mu_i|}{\sigma_i}$ ▷ compute Z-score
    end for
    $o_c \leftarrow$ number of points with $z_{score} > t_z$
    if $\frac{o_c}{|\mathcal{C}_i|} \leq z_m$ **then** ▷ outlier ratio, e.g., $z_m = 0.6$
      add $\mathcal{C}_i$ to $\bar{\mathbf{C}}$
      continue
    end if
  end for
end procedure

Clusters with $\theta < \omega_\theta$ (set to $\pi/6$) are discarded, indicating insufficient vertical alignment. This PCA-based filtering ensures that only vertical, slim, pole-like objects are retained for downstream tasks.

D. Boundary line estimation

Once pole locations are identified, boundary line estimation proceeds by classifying poles as left or right of the centreline $\mathbf{v}_c$, and estimating boundary lines based on geometric constraints. This process manages noisy data while ensuring parallelism and the maximum allowable row width $r_w \in \mathbb{R}^+$.

The poles are denoted by $\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$, where each $\mathbf{p}_i \in \mathbb{R}^2$ for $i = 0, 1, \ldots, n$. These poles are categorized into two sets: $\mathbf{p}_{ls}$ and $\mathbf{p}_{rs}$, representing the left and right sides of the centerline, respectively. The centreline direction is defined by the target position $\mathbf{p}_t \in \mathbb{R}^2$ and heading angle θ_t: $\mathbf{v}_\mathrm{c} = [\cos(\theta_t),\ \sin(\theta_t)]^\top \in \mathbb{R}^2$. Boundary line estimation involves direct and approximate methods, with a validation step to ensure accuracy for precise row boundary estimation.

1) Pole classification and line formation: Poles are classified as left or right using the cross-product criterion:

$$\begin{cases} \mathbf{v}_c \times (\mathbf{p}_i - \mathbf{p}_t) > 0 & \text{(left-side pole)} \\ \text{otherwise} & \text{(right-side pole).} \end{cases}$$

After classification, a filtering step selects the first valid pole $\mathbf{p}_1$ based on its perpendicular (lateral) distance in the robot's body frame: $d_\perp = \frac{\|\mathbf{p}_1 \times \mathbf{v}_c\|}{\|\mathbf{v}_c\|} < r_w$. For each subsequent pole $\mathbf{p}_i$, the angle with $\mathbf{v}_c$ is computed: $\cos(\beta) = \frac{\mathbf{v}_i \cdot \mathbf{v}_c}{\|\mathbf{v}_i\| \cdot \|\mathbf{v}_c\|}$, where the fitted line is denoted by $\mathbf{v}_i$ (Fig. 3). If $\beta < \omega_\beta$, the pole $\mathbf{p}_i$ is considered aligned for line formation. This process is applied separately to the left and right poles to estimate the boundary lines. Here, the ω_β threshold is set to 20° in our experiments.

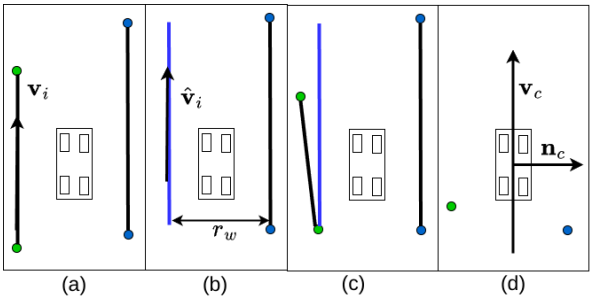

Fig. 3. Visual representation of four line estimation cases: (a) direct line estimation with aligned poles ensuring parallelism and proximity to the centreline ($\mathbf{v}_c$), (b) approximate line estimation using geometric constraints to project missing poles, (c) partial estimation with occluded poles and reconstructed lines, (d) line estimation process. Each case highlights key steps in the proposed method.

2) Direct boundary line estimation: Direct boundary lines are estimated using aligned poles (see Fig. 3). The selected boundary lines $\mathbf{v}_{\text{line}} := \bar{\mathbf{p}}_{ls}$ and $\mathbf{v}_{\text{line}} := \bar{\mathbf{p}}_{rs}$ are validated by: 1) proximity to the centreline: $d_\perp < r_w$, 2) parallelism to $\mathbf{v}_\mathrm{c}$ within tolerance, and 3) alignment with the normal vector: $\beta < \omega_\beta$. Only validated lines are retained for boundary line estimation.

3) Approximate boundary line estimation: When direct boundary line estimation fails, either due to the inability to detect a line on one side of the robot or misalignment between detected lines on both sides and centreline—approximate line estimation is employed. Using geometric constraints, missing poles and boundary lines are projected. For poles $\mathbf{p}_1, \mathbf{p}_2$, the projections are computed as: $\mathbf{p}_1^{\text{proj}} = \mathbf{p}_1 + r_w \cdot \mathbf{n}_c$, $\mathbf{p}_2^{\text{proj}} = \mathbf{p}_2 - r_w \cdot \mathbf{n}_c$. The cross-product ensures alignment: $(\mathbf{v}_c \times \mathbf{p}_1) \cdot (\mathbf{v}_c \times \mathbf{p}_1^{\text{proj}}) > 0$. If alignment is positive, the projections are accepted, and the line is computed as: $\hat{\mathbf{v}}_{line} = \mathbf{p}_1^{\text{est}} - \mathbf{p}_2^{\text{est}}$, $\hat{\beta} = \frac{|\hat{\mathbf{v}}_{line}^T \mathbf{n}_c|}{\|\hat{\mathbf{v}}_{line}\| \cdot \|\mathbf{n}_c\|}$. If $\hat{\beta} < \omega_{\hat{\beta}}$, the line is a valid approximation of the boundary line. This approach ensures robust estimation, even in the presence of occlusions or noisy data. In our experiments, the $\omega_{\hat{\beta}}$ threshold is set to 5°.

E. Boundary constraints construction

Let the estimated left and right poles be $\mathbf{p}_{ls} = [\mathbf{p_0^l}, \mathbf{p_1^l}]$ and $\mathbf{p}_{rs} = [\mathbf{p_0^r}, \mathbf{p_1^r}]$, respectively. The middle line is computed as the average of the start and end points of the left and right lines: $\mathbf{p}_{ms} = \left[\frac{\mathbf{p}_0^l + \mathbf{p}_0^r}{2}, \frac{\mathbf{p}_1^l + \mathbf{p}_1^r}{2}\right]$. The direction vector $\bar{\mathbf{v}}_c$ of the middle line is given by normalizing the vector difference between the start and end points: $\bar{\mathbf{v}}_c = (\mathbf{p}_1^m - \mathbf{p}_0^m)/\|\mathbf{p}_1^m - \mathbf{p}_0^m\|$. The boundary constraints are determined by computing an outward normal vector for each line relative to the target pose $\mathbf{p}_t$. For any line segment defined by $\mathbf{p}_0$ and $\mathbf{p}_1$, the direction vector is $\bar{\mathbf{v}}_{\text{line}} = \mathbf{p}_0 - \mathbf{p}_1$, and the vector to the target is: $\bar{\mathbf{v}} = (\mathbf{p}_1 - \mathbf{p}_t)/\|\mathbf{p}_1 - \mathbf{p}_t\|$. The corresponding normal vector is

$$\bar{\mathbf{n}} = \begin{bmatrix} -\bar{\mathbf{v}}_{\text{line},y} \\ \bar{\mathbf{v}}_{\text{line},x} \end{bmatrix}, \quad \bar{\mathbf{n}} \leftarrow \begin{cases} -\bar{\mathbf{n}}, & \text{if } \bar{\mathbf{v}}_{\text{target}} \cdot \bar{\mathbf{n}} \leq 0, \\ \bar{\mathbf{n}}, & \text{otherwise.} \end{cases}$$

Finally, the boundary constraints are represented by the matrix inequality $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, where each row corresponds to a boundary line:

$$\mathbf{A}\mathbf{x} \leq \mathbf{b} \leftrightarrow \begin{bmatrix} \bar{\mathbf{n}}_{x,\mathrm{l}} & \bar{\mathbf{n}}_{y,\mathrm{l}} \\ \bar{\mathbf{n}}_{x,\mathrm{r}} & \bar{\mathbf{n}}_{y,\mathrm{r}} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \leq \begin{bmatrix} \bar{\mathbf{n}}_l \cdot \mathbf{p}_1^l \\ \bar{\mathbf{n}}_r \cdot \mathbf{p}_1^r \end{bmatrix} \tag{1}$$

Here, $\bar{\mathbf{n}}_l$ and $\bar{\mathbf{n}}_r$ are the left and right normal vectors, and the robot position is $[x, y] \in \mathbb{R}^2$.

F. Trajectory refinement

Given the target position $\mathbf{p}_t$, boundary matrix $\mathbf{A}$ and vector $\mathbf{b}$, along with the robot's current position $\mathbf{p}_c \in \mathbb{R}^2$, the initial trajectory τ_{init} is estimated using cubic spline interpolation between $\mathbf{p}_c$ and $\mathbf{p}_t$. The reference trajectory τ_{ref} is constructed by interpolating between $\mathbf{p}_c$ and the projection of $\mathbf{p}_t$ onto the centreline, $\mathbf{p}_t^{proj}$, defined as $\mathbf{p}_t^{proj} = \mathbf{p_t} + \Big(((\mathbf{p}_0^m - \mathbf{p_t}) \cdot \bar{\mathbf{v}}_c)/(\bar{\mathbf{v}}_c \cdot \bar{\mathbf{v}}^\top{}_c)\Big) \cdot \bar{\mathbf{v}}_c$. The goal is to refine τ_{init} to be as close as possible to τ_{ref} while ensuring the kinematic feasibility (see Fig. 4). This is formulated as **a soft constraint-based non-linear optimisation problem**, minimising

$$\min_{\mathbf{x},\mathbf{y},\theta,\delta} \quad J = \lambda_t \sum_{i=1}^{n-1} (x_i - x_i^{\text{ref}})^2 + (y_i - y_i^{\text{ref}})^2 +$$
$$\lambda_s \sum_{i=1}^{n-3} \|x_{i+2} - 2 \cdot x_{i+1} + x_i\|^2 + \|y_{i+2} - 2 \cdot y_{i+1} + y_i\|^2,$$

subject to:

$$x(0) = x^{\text{ref}}(0), \quad x_{i+1} = x_i + \Delta t \cdot v_i \cos(\theta_i), \tag{2a}$$
$$y(0) = y^{\text{ref}}(0), \quad y_{i+1} = y_i + \Delta t \cdot v_i \sin(\theta_i), \tag{2b}$$
$$\theta(0) = \theta^{\text{ref}}(0), \quad \theta_{i+1} = \theta_i + \Delta t \cdot v_i \frac{\tan(\delta_i)}{l_b}, \tag{2c}$$
$$0 \leq v_i \leq v_{\max}, \quad -\delta_{\max} \leq \delta_i \leq \delta_{\max}, \tag{2d}$$

and boundary constraints from Eq. (1) apply to all waypoints.

The initial state constraints ensure the trajectory starts at the reference state. The robot's motion model provides kinematic constraints, while physical limits bound the velocity and steering angle. The trajectory must also stay within the boundary lines. The optimisation problem is solved using CasADi's [26] non-linear solver Ipopt [27], with initial guesses based on τ_{init}, and $\tau_{refined}$ is the optimised trajectory.

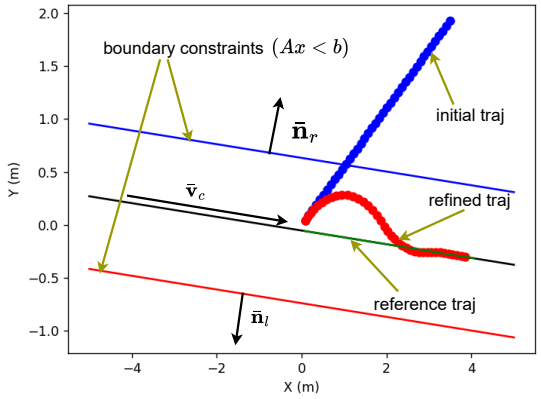

Fig. 4. Visual representation of the transformation from the initial trajectory to the refined trajectory while adhering to the constraints.

G. Regulated PurePursuit controller

A modified Pure Pursuit controller [10] is deployed for path tracking, offering improved performance with Ackermann-steered vehicles compared to a traditional PID controller [28]. Ackermann steering introduces nonholonomic constraints and a limited turning radius, which can lead to instability in PID-based heading or lateral control, especially in curved paths. In contrast, Pure Pursuit employs a geometric method that steers towards a look-ahead point, better aligning with the kinematics of Ackermann vehicles, resulting in smoother and more stable trajectories. By adapting the look-ahead distance based on speed or curvature, Pure Pursuit enhances performance, mitigating the instability and overshoot commonly associated with PID control.

The modified pure pursuit controller guides the robot along the middle of the row. The controller calculates control commands based on the robot's current position and a look-ahead point $\mathbf{p}_l = (x_l, y_l)$ on the reference trajectory $\tau_{refined}$. The look-ahead point is chosen such that the Euclidean distance from the robot's position $\mathbf{p}_c$ satisfies $\|\mathbf{p}_l - \mathbf{p}_c\| = l_d$, where l_d is the look-ahead distance.

The curvature κ of the circular arc connecting the robot to the lookahead point is calculated as $\kappa = \frac{2\hat{y}_l}{l_d^2}$, where $\hat{y}_l$ is the lateral offset of the lookahead point in the robot's local coordinate frame, given by $\hat{y}_l = \sin(\theta_c)(x_l - x_c) - \cos(\theta_c)(y_l - y_c)$. The radius of curvature r is then $r = \left|\frac{1}{\kappa}\right|$. If $r < r_{\text{min}}$, the linear velocity v_{linear} is scaled for curvature constraints as: $v_{\text{curvature}} = v_{\text{linear}} \cdot \left(1 - \frac{|r - r_{\text{min}}|}{r_{\text{min}}}\right)$, where v_{linear} is the initial linear speed (e.g., 0.5 m/s). The final linear velocity is then adjusted to $v_{\text{linear}} = \max\left(v_{\text{curvature}}, v_{\text{min}}\right)$, where v_{min} is the minimum allowable speed. The velocity is clamped to the valid range $v_{\text{linear}} \in [0, v_{\text{desired}}]$, where v_{desired} is the maximum linear velocity. The final signed velocity is $v_{\text{linear}} = \text{sign} \cdot v_{\text{linear}}$, where $\text{sign} \in \{-1, 1\}$ determines the direction of motion. The angular velocity is computed as $\omega = v_{\text{linear}} \cdot \kappa$. The control commands (v, ω) are then applied to the robot to follow the path. These steps are repeated until the robot reaches the goal, ensuring safe navigation in dynamic or curved environments. Finally, in agricultural settings, many objects cannot be considered obstacles for trajectory planning. To address this, we have implemented a dedicated safety checker that detects and classifies obstacles using LiDAR projection combined with a camera-LiDAR fusion approach. However, this safety checker is not included in this work, as it falls outside the scope of the current study.

IV. Experimental Setup and Results

A. Experimental Setup

Experiments were conducted in both simulated and real-world agricultural environments to evaluate the proposed navigation framework. Real-world trials took place at the using the AgileX Hunter 2.0 platform (Fig. 1(b)) equipped with a Trimble RTK-GNSS and front/rear Livox 3D LiDARs. These occurred in a controlled testbed with eight strawberry rows featuring occlusions like overlapping leaves, missing poles, and uneven terrain, simulating real field challenges. In the Gazebo simulation environment, a 30 m polytunnel with 1.3 m-wide rows is replicated to provide a realistic virtual setting for testing and validation. After prior simulated trials (Fig.5(c)), the data from the following experiments were collected from the real-world setting.

B. Reliability and Repeatability in Polytunnel Navigation

The robot's navigation performance in strawberry polytunnels was tested using trajectory tracking and velocity profiles during in-row navigation. A single trial, shown in Fig. 5, was repeated 10 times and analysed to assess consistency and reliability. In Fig.5(a), the robot demonstrates smooth and repeatable navigation with minimal lateral deviation from row centrelines. The real-world environment, Fig.5(b), featured varied textures and obstacles, such as vegetation and uneven surfaces, which were successfully managed.

Fig. 6 shows velocity profiles for a single trial highlighting periodic motion. The linear velocity exhibited stop-and-go behaviour typical of row-by-row navigation, while the angular velocity captured turning manoeuvres at row ends.

The robot maintained precise alignment with an average lateral deviation of 0.08 m and a σ of 0.01 m, even amid disturbances such as uneven terrain or sensor noise. The average linear velocity was steady at 0.2 m/s (σ = 0.007 m/s), suitable for tasks requiring uniform speed (e.g. spraying). The angular velocity averaged 0.05 rad/s (σ = 0.005 rad/s), reflecting accurate and smooth directional adjustments. Overall, the results show the reliability and repeatability of the system. Low variability across all metrics highlights the suitability of

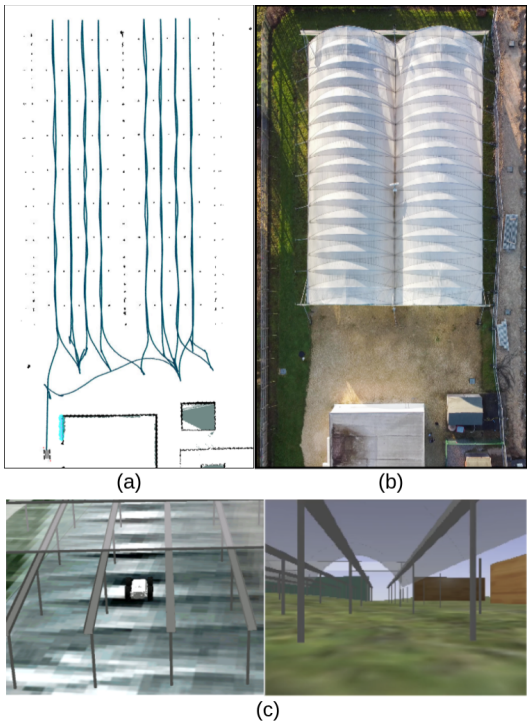

Fig. 5. Results of an in-row navigation test: (a) robot trajectory showing consistent path maintenance in narrow strawberry polytunnel corridors; (b) close-up of the real polytunnel; (c) simulated environment setup for validating the developed algorithm

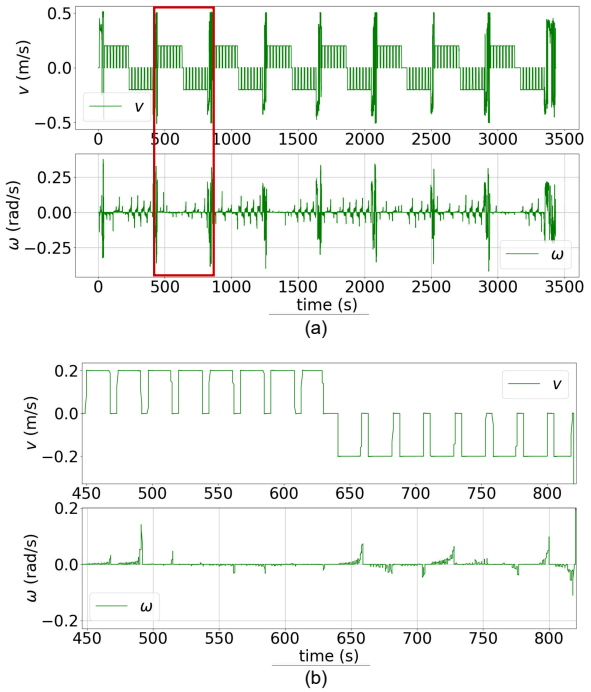

Fig. 6. Linear and angular velocity profiles during in-row navigation: (a) showing periodic motion patterns; the red box marks a region for detailed analysis. (b) Close-up of the highlighted region, illustrating velocity transitions and stability.

the navigation algorithm for structured, row-based farming environments.

C. Evaluating navigation at varying velocities

Figure 7 compares the performance metrics of the trajectory (total distance travelled ($\sqrt{x^2+y^2}$) and the lateral deviation of the reference trajectory) under three velocity constraints: $v_{\max}$ = 0.2, 0.35 and 0.5 (m/s). The left plot shows the total distance covered for forward and backward motions. Across all velocities, the robot closely follows the reference trajectory, with minor variations observed as $v_{\max}$ increases. Transition points (e.g., around index 100) remain smooth, demonstrating consistent path-following performance. The right plot highlights lateral deviations. Forward motion at $v_{\max} = 0.2$ shows slightly higher deviations, particularly near transitions, likely due to stricter kinematic constraints. In contrast, $v_{\max} = 0.5$ achieves the lowest deviations, maintaining better accuracy. All settings keep deviations under 0.15 m, with backward motions stabilising further, though $v_{\max} = 0.2$ continues to show slightly larger deviations. Shaded regions mark forward and backward phases, emphasizing that transitions tend to amplify deviations, especially at lower velocities. Higher velocity settings ensure smoother transitions and better overall accuracy. These results demonstrate the algorithm's robustness across varying velocities.

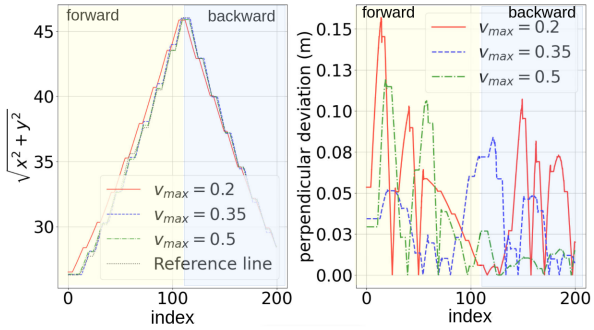

Fig. 7. Trajectory performance for different maximum velocities v_{max}) during forward (yellow) and backward (blue) motion. The left plot shows the total distance travelled ($\sqrt{x^2+y^2}$) aligned with the reference trajectory, while the right plot depicts lateral deviations. The index represents each time instance at which the lateral deviation is estimated.

D. Performance comparison with alternative planners

To evaluate the performance of the proposed narrow-space navigation planner, we compared it with two state-of-the-art approaches: Model Predictive Path Integral (MPPI) [29] and Resilient Timed Elastic Band (RTEB) [30]. MPPI uses a sampling-based approach to optimise trajectories over a prediction horizon, considering path following, control effort, and obstacle avoidance. RTEB extends the traditional Timed Elastic Band (TEB) [31] to handle environmental uncertainties and recovery behaviours, minimising deviations from the reference path.

Figure 8 shows the velocity profiles of all planners. While RTEB follows the reference line with minimal deviation, it struggles to generate a trajectory if the robot deviates slightly from the row centre (as time 50-500 s). In contrast, the proposed planner remains robust in handling such deviations, making it more reliable than RTEB. MPPI also

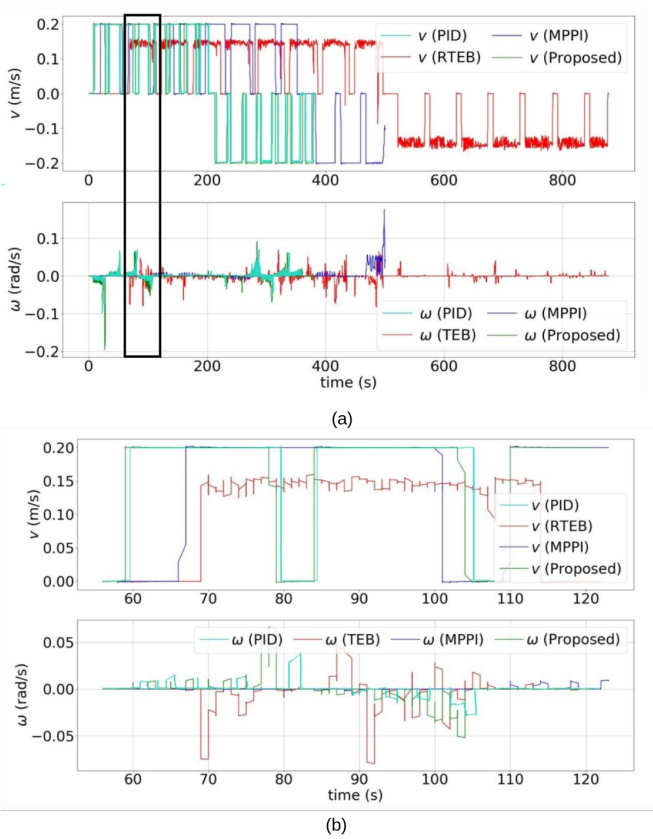

Fig. 8. Comparison of linear and angular velocity profiles for RTEB, MPPI, PID, and the proposed planner. Subfigure (b) provides a close-up view of the highlighted region in subfigure (a)

exhibits noticeable deviations, particularly towards the end of the trajectory (as time 450-500 s), indicating instability compared to the other methods.

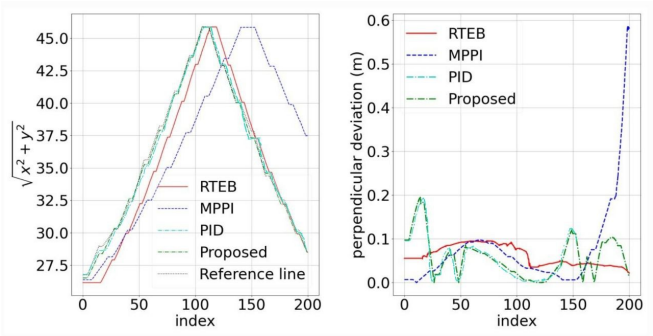

Fig. 9. The lateral deviation during in-row navigation is shown in Fig. 8. The index represents each time instance at which the lateral deviation is estimated.

The lateral deviation analysis shown in Fig. 9 further highlights the proposed planner's superior accuracy, with an average deviation approximately 10% lower than RTEB and 25% lower than MPPI. It shows the lowest lateral deviation, while RTEB's deviations are larger. MPPI demonstrates significant lateral instability, especially where the trajectory diverges from the reference path, suggesting it is less suitable for precision-critical tasks.

The velocity profiles provide additional insights into control performance. The proposed planner maintains consistent linear velocity and well-regulated angular velocity, ensuring smooth motion. RTEB, however, exhibits oscillations in both, indicating abrupt control changes that could affect efficiency. MPPI displays erratic velocity variations and higher amplitude oscillations, reflecting poor control stability.

In summary, the proposed planner outperforms RTEB and MPPI in trajectory precision, lateral deviation, and control stability, proving it more reliable for in-row navigation in agricultural environments. MPPI's erratic behaviour and lack of stability make it less suitable for such tasks. While the proposed planner demonstrates superior accuracy and stability compared to RTEB and MPPI, some limitations remain. The framework's dependency on clear structural elements, such as poles for boundary estimation, may reduce its effectiveness in unstructured or occluded environments. Additionally, while operating efficiently at 5–10 Hz, real-time applications requiring higher update rates may benefit from further computational optimisations. The trajectory-smoothing process prioritises stability, which, in some cases, may result in slower responses to sudden dynamic obstacles compared to purely sampling-based planners like MPPI.

E. Impact of pole detection and line estimation on navigation performance

Pole detection and boundary line estimation are critical components of the proposed in-row navigation planner, ensuring that the robot operates within defined boundary constraints. Key metrics were recorded to evaluate the accuracy and variability of line estimation, including correct left and right boundary line estimations, reuse of previously estimated lines, and failed line estimations.

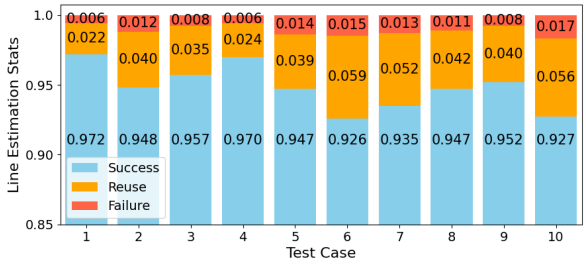

Fig. 10. Performance metrics of in-row navigation across ten test cases for pole detection and line estimation.

Fig. 10 illustrates the normalised line estimation statistics across all test cases, showing the proportion of successfully estimated lines (sky blue), reused lines (orange), and failed line estimations (red). Despite pole detection varying from 92 to 99 per metre – likely influenced by environmental factors such as occlusions – the line estimation remained consistent, averaging 226 lines per metre in all trials. Additionally, line estimation failures remained below 2% in all test cases, proving system robustness. When estimation failures occurred, the topological map manager (TMM) played a crucial role

by integrating pole location data to refine boundary line estimations. This iterative correction process, supported by TMM, effectively mitigated GNSS drift and enhanced the overall reliability of the detection and estimation framework.

V. Conclusion

This paper introduces a novel framework for autonomous navigation in narrow spaces, highlighting its successful in-row navigation application in agricultural environments. The suggested method offers robust and precise navigation in demanding settings by utilising point cloud processing, efficient pole identification, and refined trajectory tracking under linear boundary limitations and kinematic constraints. Extensive studies show that it outperforms competing planners, e.g., RTEB and MMPI, in terms of reliability, repeatability, and path accuracy, indicating its adaptability and generalisability.

Acknowledgment

This study was supported by the Innovate UK-funded project, Agri-OpenCore [grant number 10041179]. The conference expenses were covered by AURIGA AI LTD.

References

[1] H. Azpúrua, M. Saboia, G. M. Freitas, L. Clark, A.-a. Aghamohammadi, G. Pessin, M. F. Campos, and D. G. Macharet, "A survey on the autonomous exploration of confined subterranean spaces: Perspectives from real-word and industrial robotic deployments," *Robotics and Autonomous Systems*, vol. 160, p. 104304, 2023.

[2] A. Choudhary, Y. Kobayashi, F. J. Arjonilla, S. Nagasaka, and M. Koike, "Evaluation of mapping and path planning for non-holonomic mobile robot navigation in narrow pathway for agricultural application," in *2021 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2021, pp. 17–22.

[3] A. Yilmaz, A. D. Vit, I. H. Savci, H. Ocakli, and H. Temeltas, "Reference cage architecture for autonomous docking of mobile robots in automotive production systems," *The International Journal of Advanced Manufacturing Technology*, vol. 129, no. 7, pp. 3497–3511, 2023.

[4] D. Yue, S. Shang, K. Feng, H. Wang, X. He, Z. Zhao, N. Zhang, B. Zuo, and D. Wang, "Research on the model of a navigation and positioning algorithm for agricultural machinery based on the IABC-BP network," *Agriculture*, vol. 13, no. 9, p. 1769, 2023.

[5] N. AbuJabal, M. Baziyad, R. Fareh, B. Brahmi, T. Rabie, and M. Bettayeb, "A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots," *Sensors (Basel, Switzerland)*, vol. 24, no. 24, p. 8089, 2024.

[6] M. Elsanhoury, P. Mäkelä, J. Koljonen, P. Välisuo, A. Shamsuzzoha, T. Mantere, M. Elmusrati, and H. Kuusniemi, "Precision positioning for smart logistics using ultra-wideband technology-based indoor navigation: A review," *IEEE Access*, vol. 10, pp. 44 413–44 445, 2022.

[7] S. Macenski, D. Tsai, and M. Feinberg, "Spatio-temporal voxel layer: A view on robot perception for the dynamic world," *International Journal of Advanced Robotic Systems*, vol. 17, no. 2, 2020. [Online]. Available: https://doi.org/10.1177/1729881420910530

[8] H. Lim, O. Minho, and H. Myung, "Patchwork: Concentric zone-based region-wise ground segmentation with ground likelihood estimation using a 3d lidar sensor," *IEEE Robotics and Automation Letters*, 2021.

[9] A. E. Curtis, T. A. Smith, B. A. Ziganshin, and J. A. Elefteriades, "The mystery of the Z-score," *Aorta*, vol. 4, no. 04, pp. 124–130, 2016.

[10] S. Macenski, S. Singh, F. Martín, and J. Ginés, "Regulated pure pursuit for robot path tracking," *Autonomous Robots*, vol. 47, no. 6, pp. 685–694, 2023.

[11] A. N. Sivakumar, M. V. Gasparino, M. McGuire, V. A. H. Higuti, M. U. Akcal, and G. Chowdhary, "Demonstrating cropfollow++: Robust under-canopy navigation with keypoints," in *Robotics: Science and Systems 2024*, Delft, Netherlands, 2024.

[12] J. Shi, Y. Bai, Z. Diao, J. Zhou, X. Yao, and B. Zhang, "Row detection BASED navigation and guidance for agricultural robots and autonomous vehicles in row-crop fields: methods and applications," *Agronomy*, vol. 13, no. 7, p. 1780, 2023.

[13] R. De Silva, G. Cielniak, G. Wang, and J. Gao, "Deep learning-based crop row detection for infield navigation of agri-robots," *Journal of Field Robotics*, vol. 41, no. 7, pp. 2299–2321, 2024.

[14] W. Winterhalter, F. Fleckenstein, C. Dornhege, and W. Burgard, "Localization for precision navigation in agricultural fields—beyond crop row following," *Journal of Field Robotics*, vol. 38, no. 3, pp. 429–451, 2021.

[15] J. Gai, L. Xiang, and L. Tang, "Using a depth camera for crop row detection and mapping for under-canopy navigation of agricultural robotic vehicle," *Computers and Electronics in Agriculture*, vol. 188, p. 106301, 2021.

[16] P. Biber, U. Weiss, A. Albert, D. Schröder, and A. Zell, "Navigation in horticultural fields using the tree trunks as landmarks," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1630–1637.

[17] N. Chebrolu, P. Lottes, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5671–5676.

[18] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool, "Deepfruits: A fruit detection system using deep neural networks," in *Sensors*, vol. 16, no. 8. MDPI, 2016, p. 1222.

[19] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art," *Foundations and Trends in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[20] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *CVPR*. IEEE, 2017, pp. 1907–1915.

[21] D. Neven, B. D. Brabandere, S. Georgoulis, M. Proesmans, and L. Van Gool, "Towards end-to-end lane detection: an instance segmentation approach," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 286–291.

[22] M. Meyer and G. Kuschk, "Sensor fusion for joint 3d object detection and semantic segmentation," in *CVPR Workshops*, 2019.

[23] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 3126–3131.

[24] H. Liu, R. Song, X. Zhang, and H. Liu, "Point cloud segmentation based on euclidean clustering and multi-plane extraction in rugged field," *Measurement Science and Technology*, vol. 32, no. 9, p. 095106, 2021.

[25] A. Maćkiewicz and W. Ratajczak, "Principal components analysis (pca)," *Computers & Geosciences*, vol. 19, no. 3, pp. 303–342, 1993.

[26] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "Casadi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, pp. 1–36, 2019.

[27] H. Pirnay, R. López-Negrete, and L. T. Biegler, "Optimal sensitivity based on ipopt," *Mathematical Programming Computation*, vol. 4, pp. 307–331, 2012.

[28] Y. Chen, Y. Shan, L. Chen, K. Huang, and D. Cao, "Optimization of pure pursuit controller based on pid controller and low-pass filter," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 3294–3299.

[29] A. B. Steve Macenski, "nav2 mppi controller," https://index.ros.org/p/nav2_mppi_controller/, 2024, accessed: 2024-12-04.

[30] G. Kulathunga, A. Yilmaz, Z. Huang, I. Hroob, H. Arunachalam, L. Guevara, A. Klimchik, G. Cielniak, and M. Hanheide, "Resilient timed elastic band planner for collision-free navigation in unknown environments," *Journal of Field Robotics*, 2025.

[31] Y. Roussel and et al., "Dynamic replanning using the time elastic band," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2250–2255.