Computer Vision-Based Autonomous Navigation for UAV Vineyard Row Following

# Computer Vision-Based Autonomous Navigation for UAV Vineyard Row Following

Riccardo Enrico
*Department of Mechanical and Aerospace Engineering, Politecnico di Torino,*
Corso Duca degli Abruzzi 24, 10129 Torino, Italy.
riccardo.enrico@polito.it

Petre Ricioppo
*Department of Mechanical and Aerospace Engineering, Politecnico di Torino,*
Corso Duca degli Abruzzi 24, 10129 Torino, Italy.
petre.ricioppo@polito.it

Mauro Mancini
*Department of Mechanical and Aerospace Engineering, Politecnico di Torino,*
Corso Duca degli Abruzzi 24, 10129 Torino, Italy.
mauro.mancini@polito.it

Stefano Primatesta
*Department of Mechanical and Aerospace Engineering, Politecnico di Torino,*
Corso Duca degli Abruzzi 24, 10129 Torino, Italy.
stefano.primatesta@polito.it

***Abstract*—This study investigates the implementation of a Robotics operating system (ROS) enabled Unmanned Aerial Vehicle (UAV) simulation for vineyard row following. The primary motivation stems from the imprecision of GPS positioning in agricultural areas, where high accuracy is crucial for tasks such as crop irrigation. The proposed approach utilizes a computer vision algorithm to accurately determine the vineyard row position and, therefore, adjust the UAV navigation. This algorithm, combined with ROS, is tested on a PX4-enabled UAV. The findings indicate that our method offers significantly improved precision over the blind positioning obtained through a Graphical user interface (GUI) based ground control system which does not take into account the position of the vineyard row with respect to the UAV but only the end position manually selected by the user.**

***Index Terms*—Precision agriculture, Computer Vision, Unmanned Aerial Vehicles, Autonomous Navigation**

## I. Introduction

Recently, a rapid population increase coupled with the urbanization of rural areas has been observed. Therefore, the production of an increasing amount of high quality food against an ever decreasing availability of agricultural land is one of the most pressing issues of our time. The exploration of techniques and strategies to produce the required amount of food with a limited amount of land is within the objectives of Precision Agriculture (PA). PA leverages various technologies such as Internet of Things (IoT), Artificial Intelligence (AI) and robotics. While IoT and AI are mainly used for farm monitoring and management, robotics is used for many other tasks, including land preparation, planting, plant treatment, harvesting and yield estimation [1].

Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) play complementary roles in Precision Agriculture, helping optimize various farming tasks. UAVs are typically used for aerial monitoring, collecting data on crop health, soil conditions, and field mapping [2].

UGVs, meanwhile, handle ground-level tasks like planting, weeding, and pesticide application, which require greater load capacity and direct interaction with the crops [3]. UGVs often operate in point-to-point mode, efficiently moving between designated spots in the field to perform specific tasks [4] [5]. They also employ coverage strategies, ensuring that entire areas are traversed for uniform tasks like spraying or harvesting [6] [7].

This paper focuses on the autonomous navigation of UAVs for agricultural applications where the drones have to follow rows of vineyards for irrigation purposes. In particular, we address the problem of localization accuracy using GPS, which is a major issue concerning the development of autonomous robotic systems for PA applications. Indeed, UAVs in PA are required to perform tasks which require an high degree of localization precision [2], GPS signal is typically weak in rural areas which can produce highly inaccurate localization estimations and thus invalidate the robot's mission. For the specific PA application analyzed in this paper, a poor GPS signal can result in the failure to spray pesticides onto vineyard rows.

To enhance the localization accuracy of civil GPS devices, researchers proposed Real-Time Kinematics (RTK-GPS) techniques, but they incur additional costs due to the necessary hardware, such as the base station. Furthermore, while RTK improves UAV localization on the map, it does not address the offset that may exist between the map available at the ground

This study was carried out within the "National Research Centre for Agricultural Technologies – AGRITECH" and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – Avviso n. 3138 del 16/12/2021, Codice Programma CN00000022). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

This publication is part of the project PNRR-NGEU which has received funding from the MUR – DM 117/2023.

Additionally, this research activity has been supported by funds from the project "Sistema di spruzzatura basato su drone con serbatoio anti-sloshing per applicazione in agricoltura di precisione", funded by the Compagnia di San Paolo (PoC Instrument - Linea 1, PoC Launchpad, 2023).

control station and the actual position of the vineyard row.

In this work, the GPS-based autonomous navigation of a Quadrotor UAV is enhanced by integrating a computer vision algorithm. The efficacy of such an approach is validated through the use of Gazebo simulations [8] in which the simulation environment includes a real-world point cloud acquired during a monitoring task of a real vineyard. The simulation environment offers a Software in the Loop approach (SITL), in which the algorithms employed in the simulation are the same as those running on a real system's flight stack. Another rationale for implementing the simulation in Gazebo is its integration with both the Robotic Operating System (ROS) [9] leveraged by the algorithms and the PX4 Autopilot [10].

The ROS 2-PX4 approach has previously been investigated for unmanned aerial vehicles (UAVs) computer vision. However, it has been employed for the inspection of pipes [11] rather than for the purposes of following the rows of vines as described in this study.

This paper is organized as follows. Section II presents the methodology adopted to develop the autonomous navigation strategy, and also describes the organization of the data flow between the models and algorithms used. Section III details the simulation scenarios and reports the results of the numerical simulations. In particular, two different simulation scenarios were developed as explained below. The first scenario proposes a comparison between our proposed method and the GPS-based method. The results in Section III show that our proposal improves the positioning error compared to a vineyard row where positioning error is present.

The second scenario is provided in subsection III-B and aims to show the dependence of the vineyard row tracking algorithm on the quality of the point cloud scan and consequently of the simulation environment.

## II. Methodology

### A. Simulation environment

The simulation environment has been created using the Gazebo simulator, incorporating an actual point cloud scan from a vineyard located in the Laimburg Experimental Center [12].

The point cloud scan of the vineyard row was generated using a multispectral camera. The UAV generates reference data by simulating a depth camera, such as the Intel RealSense camera [13]. The depth camera itself is simulated using the depth camera plugin available for Gazebo.

The simulated world quality depends greatly on the input scan data. Figure 4(b) illustrates that while some sections allow to have a clear distinction between the vineyard rows due to the high scan quality. In other instances, however, the quality degrades, resulting in the merging of individual rows. This is demonstrated in the second simulation scenario (subsection III-B), which serves to highlight the dependency of the vineyard row following algorithm on the quality of the data utilized to develop the simulation environment.

### B. ROS 2 - Autopilot integration

Figure 1 depicts the general organization adopted by the algorithms as ROS 2 nodes. The development process and the manner in which they operate is explained in detail in this section.

Part of the ROS 2 nodes have been developed with the ROS 2 Python API (`rclpy`) in order to take advantage of the computer vision and filtering libraries, while the control nodes have been developed with the ROS 2 C++ API (`rclcpp`).

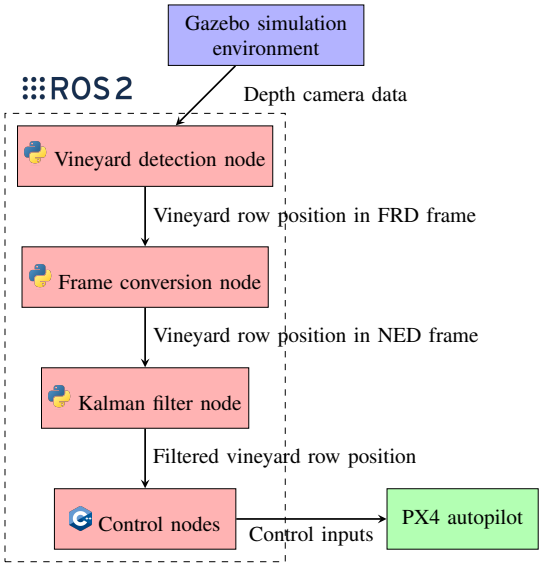



Fig. 1. Flowchart showing the organization and data flow of the ROS 2 nodes.

The ROS-enabled detection algorithm is integrated through the usage of offboard mode to the PX4 Autopilot. This enables to send control inputs to the simulated UAV through ROS 2 nodes.

As illustrated in Figure 1 the data pertaining to the centroid position of the vineyard row is expressed in the Forward-Right-Down (FRD) frame. As both the PX4 autopilot and the ROS 2 controller nodes require all data to be expressed in the global North-East-Down (NED) frame [14], the position information is translated from the FRD body frame of the quadrotor to the NED inertial frame using a ROS 2 node which employs the ROS transformation library (TF2).

### C. Vineyard Row Detection

The vineyard row centroid identification is performed within the Vineyard Row Detection Node. The implemented method is described in Algorithm 1.

The point cloud acquired from the depth camera is subjected to a transformation process whereby the terrain is flattened into a planar surface. This is achieved by identifying the terrain plane using the RANSAC algorithm, as presented in [15]. Subsequently, the modified point cloud is converted into a Height Map. Areas exceeding a threshold of 0.5 meters are uniformly colored, as only the vineyard row elements exceed this threshold. Once the Height Map has been generated, it becomes possible to detect the color area corresponding to

**Algorithm 1** Vineyard Row Detection with RANSAC pseudocode

**Input**: $P$, $N_{\max}$, $\epsilon$
Initialize best plane $\hat{\mathbf{P}} = [a, b, c, d]$
**while** $n < N_{\max}$ **do**
  Randomly select 3 points $\{p_i, p_j, p_k\} \subset \mathbf{P}$
  Compute the plane $\hat{\mathbf{P}} = [a, b, c, d]$ passing through $\{p_i, p_j, p_k\}$
  Calculate the distance of each point $p \in \mathbf{P}$ to the plane $\hat{\mathbf{P}}$
  Determine inliers: $\{p \mid |ap_x + bp_y + cp_z + d| < \epsilon\}$
  **if** number of inliers is greater than current best model **then**
    Update the best plane $\hat{\mathbf{P}}$
  **end if**
  Increment iteration counter $n \leftarrow n + 1$
**end while**
Transform point cloud into height map $H$
Identify colored region corresponding to vineyard row
Detect edges and centroid $(x_c, y_c)$ of the vineyard row
**Output:** Centroid coordinates $(x_c, y_c)$ and row boundaries

the vineyard rows and determine its centroid. This is achieved using the OpenCV Python library, as presented in [16].

The use of the RANSAC algorithm and the OpenCV library enables the process to be computationally efficient, thereby facilitating real-time application.

### D. Kalman filter

Once the data incoming from the detection algorithm has been translated to a suitable reference frame, the vineyard position correction data is filtered using a Kalman filter. The development of such a node is accomplished through the utilization of the Filterpy Python library [17].

The state vector, in Equation (1), adopted in the Kalman filter incorporates a state representing the centroid of the vineyard row. $x$ and $y$ represent the position of the center of the vineyard row section within the depth camera field of view, which varies over time. $\psi$ is the yaw angle of the vineyard row, expressed in the NED frame is assumed to be constant and, therefore, the prediction model does not vary this state. However, this state is continuously updated by the computer vision algorithm, as shown in (2) related to the input data array. Finally, $v_x$ and $v_y$ indicate the fictitious velocity of the vineyard row centroid. This value is imposed by the user, through a ROS 2 parameter, as this corresponds to the velocity of the UAV during the task completion.

Equation (3) and (5) show respectively the state transition model and the observation model which highlight the relation between the true state and the sensor input. Moreover, in Equation (3) and (5) the process and measurement noise are present, which covariance matrix is shown in Table I.

$$x = \begin{bmatrix} x & y & \psi & v_x & v_y \end{bmatrix}^T \tag{1}$$

$$z = \begin{bmatrix} x & y & \psi \end{bmatrix}^T \tag{2}$$

$$x_k = Fx_{k-1} + Bu_k + w_k \tag{3}$$

$$F = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = 0 \tag{4}$$

$$z_k = Hx_k + v_k \tag{5}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{6}$$

### E. Kalman filter parameters

Table I illustrates the parameters selected for the Kalman filter. The objective of this selection was to ensure that the algorithm exhibited a high degree of trust in the readings received from the sensor, while still allowing for a certain degree of filtering. Since the sensor update frequency is constrained by the computational time of the detection algorithm, which operates at 2 Hz, it necessitates a balance between accurate state estimation and responsiveness to new measurements. The chosen parameters, including a prediction frequency of 10 Hz, enable the Kalman filter to interpolate between sensor updates effectively.

As previously stated, a ROS 2 parameter is used to regulate the XY-plane velocity of the UAV. This parameter is also shared by the control node, in which the XY-velocity is enforced as a saturation value. The velocity value utilized in the simulations shown in this paper is 2 m/s.

TABLE I
VALUES OF THE PARAMETERS OF THE KALMAN FILTER.

| Parameter | Value |
|---|---|
| Q - Process uncertainty/noise cov. matrix | diag(2,2,10,10,10) |
| R - Measurement uncertainty/noise cov. matrix | diag(1e-3,1e-3,1e-3) |
| Prediciton frequency | 10 Hz |
| Sensor update frequency | 2 Hz |
| $v_{xy}$ - UAV velocity | 2 m/s |

### F. Kalman filter results

As it is shown in Figure 2 the Kalman filter maintains a constant-heading prediction of the position of the vineyard row. A sensor update step is performed each 5 filter prediction steps. Due to the tuning of the filter parameters, namely the Q and R matrices, Table I, the filter prioritizes sensor measurements thus effectively allowing for the filter to change the vineyard row $\psi$ value.

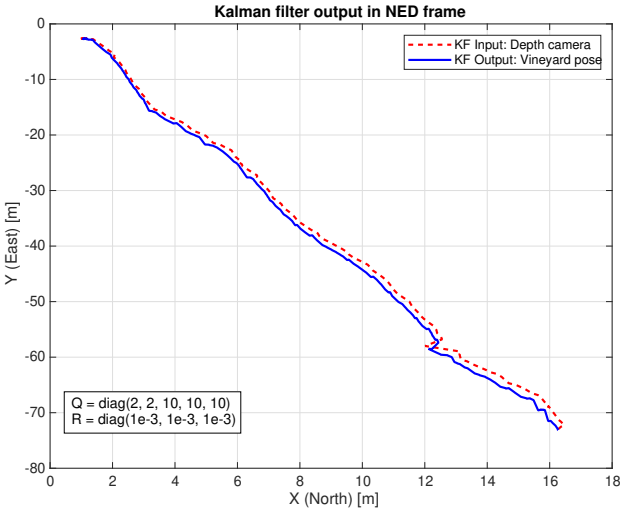



Fig. 2. Kalman filter estimation of the position of the vineyard row in the NED frame.

### G. ROS 2 Control of the UAV position

The ROS 2 nodes have been designed and developed to manage an input of desired position defined by a user either by employing the graphic user interface (GUI) of a ground control station e.g. QGroundControl or directly using the command line interface (CLI). Thus, the UAV is controlled by leveraging the PX4 offboard mode.

In our method, the ground control station is employed to locate the end position of the vineyard row on the available map, although up to this point, an offset between the map-selected reference and the vineyard row's real location is present.

This is actively corrected through the ROS 2 algorithms that, as explained in the previous sections, detect the correct location of the vineyard row from the depth camera, control the positioning of the UAV, and ensure correct trajectory tracking.

The reference position offset information incoming from the previous ROS 2 nodes is filtered using the Kalman filter algorithm (Figure 1), then this new reference is managed by a ROS 2 action server which orchestrates the detection and the control algorithms.

Moreover, the ROS 2 action employs a finite state machine structure to ensure the code is easy to read and develop, thus enabling easier future developments.

The control algorithms manage communication between the ROS-enabled algorithms and the PX4 autopilot. From the updated position reference, control inputs are computed by a PD velocity controller and sent as velocity setpoints to the autopilot. The communication to the latter established through the uXRCE-DDS middleware, enabling low-level control of the actuators.

The proposed method is compared to the method which employs only the reference from the ground control station but no active computer vision correction, this is used as the baseline in the result section (Figure 3).

## III. Results

Our research activity aimed to demonstrate that using an actively corrected reference point, gathered from a depth camera and sent to a UAV via ROS 2, enhances vineyard row tracking compared to the traditional method, which employs a single reference position at the end of the row and relies only on GPS for correct positioning.

### A. Simulation scenario 1

The simulation was conducted using a meticulously selected subset of the available point cloud, in which the vineyard rows were clearly delineated. As displayed in the results shown in Figures 3(b) and 4(a), the identification and following performance increases significantly in the presence of a higher quality point cloud.

In Figure 3(a), the UAV's position is plotted against the vineyard row's ground-truth position in a case without the proposed correction. Both trajectories are obtained by exploiting the simulator. Figure 3(b) shows the same image with position correction enabled. Table II and Figure 3 show a substantial decrease in the mean error in the case in which the correction is enabled, in comparison to a simulation where it is not.

TABLE II
Root Mean Square Error (RMSE) for UAV position in two cases.

| Test case | RMSE [m] |
|---|---|
| No correction | 1.059 |
| Correction enabled | **0.265** |

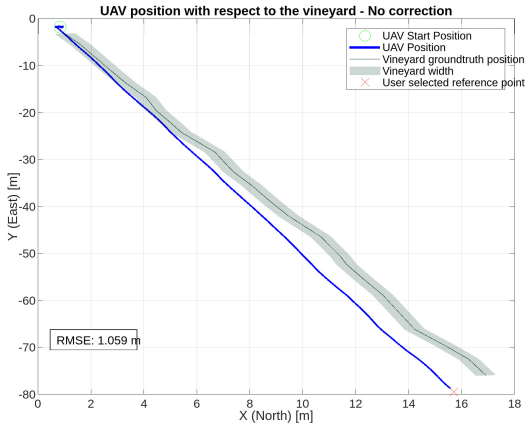
The comparison highlights the imprecision of the map used by the control station software in contrast to the detection algorithms which instead actively corrects the position of the quadrotor based on the vineyard row identification.

Furthermore, in the uncorrected case, the UAV shows less directional change due to the singular reference point, as opposed to continuous correction which is present in the other case.

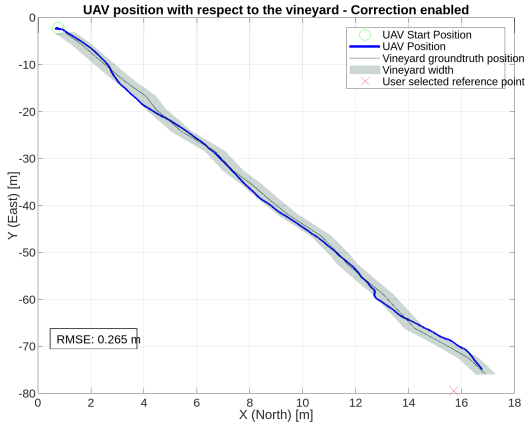
Moreover, Figure 3 shows that the user-selected reference point is blindly followed in the case without correction (Figure 3(a)). In contrast, in our method (Figure 3(b)) the reference point, while it being the same and being wrong compared to the true position of the vineyard row end position is corrected by the algorithm.

### B. Simulation scenario 2

The second scenario evaluates and highlights the dependency of the position correction on a good quality of the point cloud used to generate the simulation environment. As displayed in Figure 4(a) the UAV maintains a good tracking of the vineyard row. However, Figure 4(b) displays two vineyard rows that appear joined together, this error prevents the detection algorithm from identifying the real vineyard row center. The Kalman filter parameters are left unchanged from the previous simulation scenario, to asses the tracking performance depending on the incoming data quality. Despite

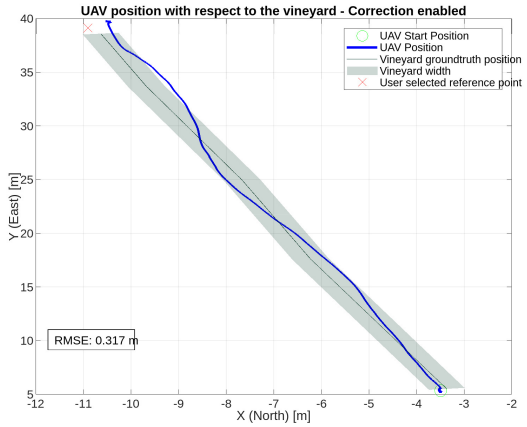



(a) UAV Position with respect to the vineyard row true position. Case without position correction.

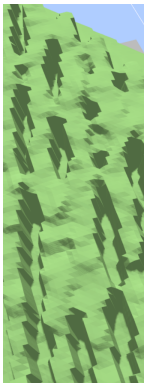



(b) UAV position with respect to the vineyard row true position. Case with position correction.

Fig. 3. UAV Positioning error comparison in case in which the position correction is enabled (3(b)) and not (3(a)).




(a) Simulation scenario 2 - Kalman filter vineyard row position estimation with respect to the true position of the vineyard row.



(b) Simulation environment. Showing the joined rows due to poor scan quality.

Fig. 4. Figures showing the outcome of the vineyard row following task (4(a)) and the simulation environment on which this result was obtained (4(b)).

a noticeable performance decline, the mean positioning error is still lower than the average vineyard row width, thanks to the robustness provided by the Kalman filter node.

## IV. Conclusion

This paper presents a method for improving the precision of vineyard row following tasks for a UAV in precision agriculture by actively correcting the position reference. The simulation results demonstrate that the correction algorithm, despite being susceptible to the quality of the incoming point cloud, offers a superior solution to the single reference point method.

Figure 4 illustrates a decline in the tracking performance when compared to what has been shown in 3(b). As previously stated, the Kalman filter enhances tracking performance in scenarios where the vineyard row detection is not as precise. As indicated in the results section, the parameters were not varied in order to evaluate the dependency of the tracking performance on the input data quality itself. An alternative approach could be to vary the Kalman filter parameters, thus enabling the node to assign a lower importance to the sensor inputs in favor of the prediction. Alternatively, the outliers (i.e., the positions pertaining to other vineyard rows) might be discarded using metrics such as the Mahalanobis distance [18].

Future research will concentrate on enhancing the Kalman filter estimation and adapting its parameters in accordance with the quality of the incoming data. Furthermore, the results of this study have demonstrated a correlation between the quality of the point cloud and the subsequent performance of the vineyard row following the algorithm. Consequently, future tests will be conducted using more precise and high-quality point clouds.

In this work, a point cloud from a real-world application was used to generate a simulation environment. In future work,

a depth camera will be employed to evaluate the ability of the proposed approach to integrate with depth sensor data in outdoor conditions, assessing its robustness and suitability for real-world applications.

Furthermore, a more integrated approach will be developed between the correction algorithm and the ground control station, with the objective of enhancing the user-friendliness of the correction algorithm.

## REFERENCES


[1] L. F. P. Oliveira, A. P. Moreira, and M. F. Silva, "Advances in agriculture robotics: A state-of-the-art review and challenges ahead," *Robotics*, vol. 10, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2218-6581/10/2/52

[2] P. Surico, N. Bloise, S. Primatesta, and G. Guglieri, "Design and stability analysis of an agricultural sprayer uas integrated with an anti-sloshing tank," in *2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE, 2023, pp. 788–793.

[3] A. Khadatkar, C. R. Mehta, and C. P. Sawant, "Application of robotics in changing the future of agriculture," *Journal of Eco-friendly Agriculture*, vol. 17, no. 1, pp. 48–51, Aug. 2022. [Online]. Available: https://acspublisher.com/journals/index.php/jefa/article/view/7871

[4] M. Mancini, E. I. Trombetta, D. Carminati, and E. Capello, "Adaptive sliding mode control with artificial potential field for ground robots in precision agriculture," in *2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. IEEE, 2023, pp. 325–330.

[5] P. Ricioppo, D. Celestini, and E. Capello, "Generalization of reinforcement learning through artificial potential fields for agricultural ugvs," in *2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, 2023, pp. 386–391.

[6] S. Chakraborty, D. Elangovan, P. L. Govindarajan, M. F. ELnaggar, M. M. Alrashed, and S. Kamel, "A comprehensive review of path planning for agricultural ground robots," *Sustainability*, vol. 14, no. 15, 2022. [Online]. Available: https://www.mdpi.com/2071-1050/14/15/9156

[7] D. Celestini, S. Primatesta, and E. Capello, "Bio-inspired complete coverage path planner for precision agriculture in dynamic environments," in *2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*, 2023, pp. 777–782.

[8] "Gazebo," https://gazebosim.org/home.

[9] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[10] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240.

[11] Y. M. R. da Silva, F. A. A. Andrade, L. Sousa, G. G. R. de Castro, J. T. Dias, G. Berger, J. Lima, and M. F. Pinto, "Computer Vision Based Path Following for Autonomous Unmanned Aerial Systems in Unburied Pipeline Onshore Inspection," *Drones*, vol. 6, no. 12, p. 410, Dec. 2022.

[12] "Centro di sperimentazione Laimburg," https://www.laimburg.it/it/default.asp.

[13] "Intel® RealSense™ computer vision - depth and tracking cameras," https://www.intelrealsense.com/, Jul. 2024, accessed: 2024-9-23.

[14] "Controller diagrams, px4 guide (main)," https://docs.px4.io/main/en/flight_stack/controller_diagrams.html.

[15] J. M. Martínez-Otzeta, I. Rodríguez-Moreno, I. Mendialdua, and B. Sierra, "Ransac for robotic applications: A survey," *Sensors*, vol. 23, no. 1, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/1/327

[16] J. Howse, *OpenCV computer vision with python*. Packt Publishing Birmingham, UK, 2013, vol. 27.

[17] "FilterPy 1.4.4 documentation," https://filterpy.readthedocs.io/en/latest/.

[18] A. Minervini, S. Primatesta, and G. Guglieri, "Enhanced altitude estimation for unmanned aerial vehicles in a GNSS-denied environment," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, Jun. 2024, pp. 1177–1183. [Online]. Available: https://ieeexplore.ieee.org/document/10556904