Point Cloud Based Visual Planning and Servoing for Autonomous Vine Pruning

Fadi Gebrayel[1], Thomas O'Brien[2], Martin Mujica[1], and Patrick Danès[1]

***Abstract*—This paper addresses vision based autonomous vine pruning with robots. The complex structure of vines makes visual servoing difficult due to challenges in feature extraction and 3D pose estimation. A novel approach interweaving visual servoing and vision based planning is proposed. An online planner features a Nonlinear Model Predictive Control to compute on-the-fly suitable 3D waypoints. These are navigated by a Position Based Visual Servoing integrating Iterative Closest Point based point-cloud alignment. Qualitative and quantitative evaluations are conducted on live experiments where a Franka Emika manipulator embedding a eye-in-hand stereo camera performs a sequence of pruning actions on seven increasingly complex vine stocks, in spite of unexpected motion of the vine and/or the robot's base. Outdoor experiments are also carried out in strong wind conditions.**

I. INTRODUCTION

A. Context

Recent years have witnessed an important growth in robots working in unstructured complex environments. In particular, there has been a significant interest in autonomous robotic systems for agricultural tasks [1], [2]. For instance, a common approach in autonomous pruning and harvesting consists of one or several robotic arms, that perform complex motion tasks, based on advanced perception systems. In addition, recent advances in fruit segmentation and detection have led to new fruit harvesting results such as [3] or [4] for peppers and small fruits. Gursoy et al. [5] proposed a dual-arm harvesting robot that utilizes visual data to detect fruits and tree trunks. Authors in [6] and [7] also used multiple robot arms for fruit harvesting, one for perception and the other for manipulation, at the expense of increased cost.

Among the agricultural tasks, vine pruning is essential for vineyard management. It impacts grape quality and yield, but remains labor-intensive and ergonomically challenging. Similarly to harvesting, pruning also requires advanced sensor-based control to navigate in complex environments and to manage dynamic factors (e.g., wind, vibrations) as well as unexpected changes. However, vine pruning poses significant challenges to sensor-based control, as the extraction and tracking of features are hindered by the plant's repetitive and intricate structure. Consequently, no robotic solution has yet fully replaced manual pruning. In research laboratories, multiple experimental approaches have been explored. Botterill et al. [8] developed a prototype that integrates advanced computer vision, machine learning, and motion technologies. The authors highlight the challenges posed by the long chain of interdependent components, which complicate practical implementation. Similarly, the Bumblebee robot [9] incorporates a robust mechanical design composed of a UR5 robotic arm on a mobile base along with several exteroceptive sensors. Using 3D reconstruction, this system performs bud detection and cutting pose localization (based on pruning rules). Motion planning is used to define the movements based on an optimal sequence of predefined cuts.

Most of these works do not bring a satisfactory solution to the displacement of the robot arm and end-effector tool. Indeed, they generally execute planned motions in open-loop w.r.t. the environment. So, they cannot cope with unavoidable modeling errors, nor with vine relative shifts induced by wind, contact with the robots, vibrations or displacements of the mobile base. Nevertheless, robustness to such artifacts remains a key factor to a successful deployment of agricultural autonomous robots. Therefore, this work aims at proving how a suitably designed vision based control can bring an effective solution to vine pruning, even when unexpected movements of the vine and/or mobile base occur.

B. Related work

Vine pruning is a complex task, whose automation requires the interconnection of several components [8]: vine perception, cutting poses definition, robot motion, etc. The recent study [10] merges 2D segmentation with point clouds. The cutting poses are automatically generated, and motions for them are planned. However, the relative pose between the vine and the robot is assumed constant between the planning and the pruning, which is not realistic. In [11], once pruning points are localized in the global frame, a whole-body velocity planner generates the approach trajectory that coordinates the movements of the 2-DoF non-holonomic mobile base and the 7-DoF robotic arm. Nevertheless, no vision feedback is integrated into the closed-loop execution of this trajectory, so robustness to the aforementioned artifacts is not ensured.

The work of Yandun et al. [12] generates the robot's reference motion through a combination of reinforcement learning and inverse kinematics, yet execution remains in open-loop w.r.t. the environment. In contrast, Katyara et al. [13] focus on perception, cutting point learning, and motion planning while

Manuscript received: August 25, 2025; Revised: November 19, 2025; Accepted: January 31, 2026.

This paper was recommended for publication by Editor Hyungpil Moon upon evaluation of the Associate Editor and Reviewers comments.

*This work was jointly supported by "Défi Clé Robotique centrée sur l'humain" (funded by La Région Occitanie, France), LAAS-CNRS, Toulouse, France, and the University of Newcastle, Newcastle, Australia.

[1]Fadi Gebrayel, Martin Mujica, and Patrick Danès are with LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France. {fadi.gebrayel, martin.mujica, patrick.danes}@laas.fr

[2]Thomas O'Brien is with the University of Newcastle, Newcastle, Australia. thomas.obrien@uon.edu.au

Digital Object Identifier (DOI): see top of this page.

also accounting for vine rod stiffness with admittance control. The recent work [14] introduces a complete outdoor-tested pipeline featuring an efficient perception system. However, this advanced system does not ensure a 100% cutting success rate. One reason is that, as in most approaches, it relies on basic motion planning without execution in a closed loop with the environment, which may lead to task failure when unexpected wheel slip or shifts in the vine occur. In addition, Liu et al. [15] used labeled point clouds to reconstruct 3D models and identify cut points. The authors plan collision-free, manipulability-aware paths combined with position-based servoing to precisely reach the cut points. Their baseline results highlight the advantage of incorporating an in-hand camera for local observation. However, the method incurs longer computation times than simpler sensor based controls, struggles to balance collision avoidance with task completeness, and assumes static trees, neglecting dynamic changes.

To address the difficulty of the task, a related work [16] introduced vision based closed-loop control using point cloud data. The results showed the potential benefits of point cloud data usage to reach cutting poses, but no answer was brought to convergence from large initial errors nor to collision avoidance. To solve these issues, we propose to incorporate an online generation of a sequence of frame poses (or "waypoints") to the cutting pose. We draw inspiration from the monocular vision-based navigation system for mobile robots described by Cherubini et al. [17]. Unlike their method that relies on a 2D visual path from previously recorded images, our method computes an online 3D visual path, generated from the 3D model of the vine, to maximize vine visibility while minimizing the risk of collisions. The method will be shown to adapt to environmental changes, and allow the robot to reach the cutting pose from large initial errors.

The next section presents the pruning problem, summarizes the paper's contribution, and outlines its organization.

II. System overview

A. Notation

The following notation is used. I_n stands for the $n \times n$ identity matrix. The Euclidean and weighted norms of $x \in \mathbb{R}^n$ are denoted by $\|x\|$ and $\|x\|_P^2 = x^\top P x$, with $P = P^\top \in \mathbb{R}^{n \times n}$ a positive definite matrix. The homogeneous transform expressing F_j in the reference frame F_i is denoted by ${}^iH_j$, with its associated rotation matrix ${}^iR_j \in \mathrm{SO}(3)$ and translation vector ${}^it_j \in \mathbb{R}^3$. We denote a pose vector as ${}^is_j = \left[{}^it_j^\top, {}^i\Theta_j^\top\right]^\top \in \mathbb{R}^6$, where ${}^i\Theta_j \in \mathbb{R}^3$ represents the Euler angle parametrization of ${}^iR_j$. Unless otherwise specified, if a pose vector does not explicit its reference frame (e.g., as in "s_j") then, it is assumed to be expressed in the robot base frame F_B. The pseudo–inverse of any matrix A is denoted by A^+.

B. Autonomous Vine Pruning Problem Statement

Given a point cloud Ω_{env} of the complex vine environment (referenced w.r.t. the robot base frame F_B), the end-effector tool (EOT) of a robotic manipulator must be guided to a sequence $\mathbb{C} = \{s_j^*\}_{j=1}^J$ of J predefined reference cutting poses $s_j^* = \left[{}^Bt_j^{*\top}, {}^B\Theta_j^{*\top}\right]^\top$, where ${}^Bt_j^* \in \mathbb{R}^3$ and ${}^B\Theta_j^* \in \mathbb{R}^3$ denote the reference EOT cutting position and orientation with respect to F_B. The EOT must navigate the cutting poses by using the visual feedback from its attached eye-in-hand RGB-D camera.

C. Proposed Approach and Organization of the Paper

Previous study [16] introduced a position-based visual servoing (PBVS) method leveraging 3D pose estimation via the Iterative Closest Point (ICP) algorithm. This approach enabled successful cuts despite environmental changes in the vine and can be readily generalized to any structure for which a 3D model exists or can be acquired through scanning. However, it is prone to failure because the underlying ICP registration cannot be applied to distant point clouds occurring in complex scenes. To make the robot EOT navigate between different poses around the vine by visual feedback, we build upon and extend [17], by shifting the visual memory concept from 2D features to 3D point clouds. This leads to a visual navigation framework designed for vine pruning but adaptable to other tasks: harvesting, object grasping, etc. Our key contributions are the following:

- A *Nonlinear Model Predictive Control (NMPC) style planner* of a sequence of poses (or "waypoints") between any initial EOT pose to any reference cutting pose s_j^*, which are collision-free, maintain a consistent view of the vine and enable successful local ICP registration of the point clouds collected at these waypoints.
- An *ICP based PBVS (ICP-PBVS)* that navigates the robot along the above collision-free waypoint sequence even in the presence of errors in Ω_{env} or of unexpected online disturbances (e.g., robot or branch movements). This high-level reactive controller combines ICP based point cloud alignment with suitable setpoints defined on-the-fly so as to ensure the EOT convergence to s_j^*.

The system is overviewed in Figure 1. The NMPC planner takes the cutting poses $\{s_j^*\}_{j=1}^J$ and environment point cloud Ω_{env} as inputs to provide a sequence of point cloud references $\{\Omega_w\}$ which guide the EOT to the desired cutting pose. These point clouds are used by the high-level controller to generate velocity references to the low-level controller.

The remainder of the paper is organized as follows. Section III presents the design of the NMPC-style waypoint planner. Section IV details the ICP-PBVS. Section V showcases experimental results on real vines, including dynamic and challenging visibility conditions, as well as outdoor evaluations. Finally, Section VI concludes the paper, discusses limitations and potential directions for future research.

III. NMPC Waypoint Planner

The aim is to generate a sequence of W waypoints

$$\mathbb{W} = \{s_w\}_{w=1}^W \tag{1}$$

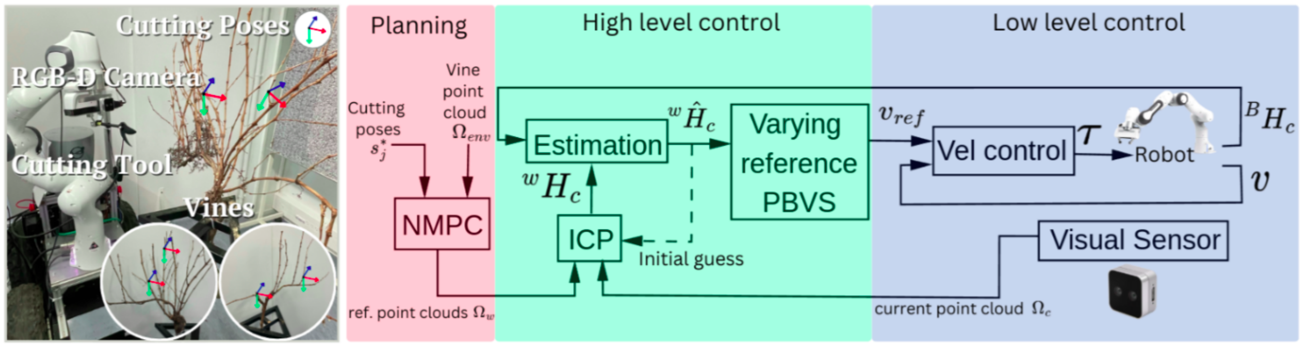


Fig. 1: (Left) Franka arm equipped with a cutting tool and RGB-D camera, along with cutting poses. (Right) Schematic diagram of the proposed method. Circled images show two other vine samples.

which connect the current EOT pose s_{init} to a given reference cutting pose s_j^*. Each w^{th} waypoint is parameterized as

$$s_w = \left[{}^Bt_w^\top, {}^B\Theta_w^\top\right]^\top, \tag{2}$$

with ${}^Bt_w \in \mathbb{R}^3$ and ${}^B\Theta_w \in \mathbb{R}^3$ is position and orientation. To enable reliable visual servoing, the sequence $\mathbb{W}$ must be collision-free and ensure visibility of the vine.

The NMPC waypoint planner is described in Algorithm 1 and illustrated on Figure 2. At each iteration, a "look-ahead" sequence of waypoints is computed over a receding prediction horizon N through dynamic optimization. The first waypoint from the obtained optimal sequence is appended to $\mathbb{W}$. Then, it is selected as the new initial state, and the process is repeated until the translational and rotational errors fall below some preset tolerances ε_c.

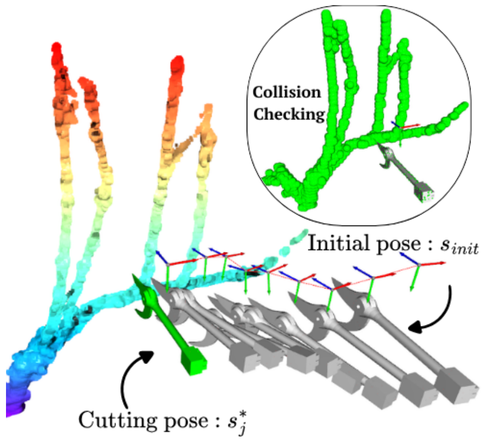


Fig. 2: Example sequence of waypoints. Collision margins are displayed as green spheres.

We define the prediction model as

$$s_{k+1} = f(s_k, u_k) = s_k + u_k \tag{3}$$

where $s_k = \left[{}^Bt_k^\top, {}^B\Theta_k^\top\right]^\top \in \mathbb{R}^6$ terms the predicted waypoint at step k in the optimization problem, with ${}^Bt_k \in \mathbb{R}^3$ and ${}^B\Theta_k \in \mathbb{R}^3$ the EOT cutting position and orientation, and the control input $u_k = \left[\Delta t_k^\top, \Delta\Theta_k^\top\right]^\top$ consists of the respective position and

Algorithm 1 NMPC Waypoint Planner

```
Require: $s_j^*$: reference cutting pose; $s_{\text{init}}$: current EOT pose; $N$: horizon length; $\varepsilon_c$: cutting pose-error tolerance
1: $\mathbb{W} \leftarrow \emptyset$                          ▷ Initialize waypoint sequence
2: while $\|e(s_{\text{init}}, s_j^*)\| > \varepsilon_c$ do
3:     Solve NMPC:
           $\{\hat{u}_k\}, \{\hat{s}_k\} \leftarrow \arg\min_{\{u_k\}} \sum_{k=0}^{N-1} \ell(s_k) + \ell_f(s_N)$
4:     $\mathbb{W}$.append($\hat{s}_1$)                   ▷ Append first waypoint
5:     $s_{\text{init}} \leftarrow \hat{s}_1$                       ▷ Shift horizon to $\hat{s}_1$
6: end while
```

orientation increments $\Delta t_k \in \mathbb{R}^3$ and $\Delta\Theta_k \in \mathbb{R}^3$. Over the N-length receding horizon, NMPC seeks the optimum control sequence:

$$\hat{u}_0, \ldots, \hat{u}_{N-1} = \arg\min_{\{u_k\}} \sum_{k=0}^{N-1} \ell(s_k) + \ell_f(s_N) \tag{4}$$

$$\text{subject to} \quad s_{k+1} = f(s_k, u_k), \quad k = 0, \ldots, N-1, \tag{5}$$

$$s_0 = s_{\text{init}}, \tag{6}$$

$$u_{\min} \le u_k \le u_{\max}, \quad k = 0, \ldots, N-1. \tag{7}$$

At each prediction step k, the stage cost

$$\ell(s_k) = J_{\text{pose}}(s_k) + J_{\text{coll}}(s_k) + J_{\min}(s_k) + J_{\text{los}}(s_k) \tag{8}$$

and terminal cost $\ell_f(s_N)$ are as follows.

a) Pose reaching: The quadratic cost defined as

$$J_{\text{pose}}(s_k) = \frac{1}{2}\|e(s_k)\|^2_{\Lambda_{pose}} \tag{9}$$

encourages the current state s_k to reach the next reference cutting pose s_j^*, by defining

$$e(s_k) = \begin{pmatrix} {}^Bt_k - {}^Bt_j^* \\ \alpha({}^BR_j^{*\top}\,{}^BR_k) \end{pmatrix} \in \mathbb{R}^6, \tag{10}$$

where the rotation matrices ${}^BR_k, {}^BR_j^* \in SO(3)$ are respectively associated to ${}^B\Theta_k, {}^B\Theta_j^*$, the function $\alpha(\cdot): SO(3) \to \mathbb{R}^3$ stands for the numerically robust axis-angle representation and $\Lambda_{pose} \in \mathbb{R}^{6\times6}$ is a diagonal matrix weight.

b) Collision avoidance: In view of the vine complexity, irregularity and proneness to clutter, the planning of the sequence of waypoints must incorporate collision avoidance. Depending on the setup and how close the robot is to the vines, the collision checking should be done with respect to the end effector, the last links or the full robot. In our case, we consider collisions with the cutting tool positioned as the robot's end effector, as these are the most likely. Collisions with the entire robot body could also be handled, but at the expense of an increased computation time. Let $\Omega_{\text{eot}}(s_k) \subset \mathbb{R}^3$ be the set of vertices (points) defining the EOT geometry positioned at waypoint s_k. The minimum distance between the EOT and the environment is given by

$$d_{\text{coll}}(s_k) = \min_{\substack{p_{\text{eot}} \in \Omega_{\text{eot}}(s_k) \\ p_{\text{env}} \in \Omega_{\text{env}}}} \| p_{\text{eot}} - p_{\text{env}} \|, \tag{11}$$

where $\Omega_{\text{env}} \subset \mathbb{R}^3$ terms the aforementioned point cloud obtained from a prior 3D scan of the vine. In practice, Eq. 11 is computed efficiently via a kd-tree spatial query. The collision-avoidance cost is then defined as

$$J_{\text{coll}}(s_k) = \begin{cases} \frac{\lambda_{\text{coll}}}{2\delta} \left(d_{\text{coll}}(s_k) - \delta \right)^2, & \text{if } 0 \leq d_{\text{coll}}(s_k) \leq \delta, \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

where $\delta > 0$ is a user-defined safety margin and $\lambda_{\text{coll}} > 0$ is the collision cost weight. This is illustrated on Figure 2. As usual in such methods, a tradeoff exists between the sharpness of collision checking and the computational cost.

c) Visibility: To ensure proper functioning of ICP-PBVS (detailed in Section IV), several visibility components are incorporated into the criterion of the NMPC planner. The continuous presence of a minimum number of points within the camera's field of view is enforced thanks to the soft constraint term

$$J_{\min}(s_k) = \begin{cases} e^{\alpha_{\text{vis}} \left(n_{\min} - n_{\text{vis}}(s_k) \right)} - 1, & \text{if } n_{\text{vis}}(s_k) < n_{\min}, \\ 0, & \text{otherwise}, \end{cases} \tag{13}$$

where $\alpha_{\text{vis}} > 0$ determines the penalty slope for falling below the visibility threshold, $n_{\min}$ is the desired minimum amount of visible points, and

$$n_{\text{vis}}(s_k) = |\Omega_{\text{vis}}(s_k)| > 0 \tag{14}$$

terms the number of points composing the point cloud $\Omega_{\text{vis}}(s_k)$ retrieved within the frustum of the camera at pose s_k, see Figure 3.

In visual servoing, it is desirable that the camera's optical axis points toward a desired target. Define this target as

$$^{B}t_d = {}^{B}t_j^* + {}^{B}R_j^* \left[0, 0, d_{\text{tool}} \right]^\top, \tag{15}$$

with d_{tool} the distance along the optical axis to the end of the cutting tool. A line-of-sight cost can then be defined as

$$J_{\text{los}}(s_k) = \lambda_{\text{los}} \left(\arccos(\hat{z}_k \cdot \hat{r}_k) \right)^2, \tag{16}$$

with $\lambda_{\text{los}} > 0$ the line of sight cost weight and

$$\hat{r}_k = \frac{^{B}t_d - {}^{B}t_k}{\| ^{B}t_d - {}^{B}t_k \|} \tag{17}$$

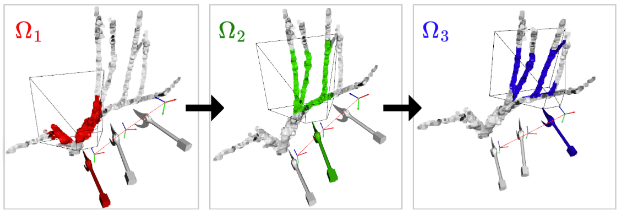


Fig. 3: Waypoints with associated point clouds Ω_w

the unit vector from the current waypoint origin to the desired target. Additionally, hard constraints (7) on the control inputs limit position and orientation changes between waypoints, preventing abrupt shifts in the environment's view.

d) Terminal cost : To emphasize final pose accuracy, the terminal cost is selected as

$$\ell_f(s_N) = \frac{1}{2} \| e(s_N) \|^2_{\Lambda_{term}} \tag{18}$$

where $e(s_N)$ is the pose error of the final waypoint in the horizon and $\Lambda_{term} \in \mathbb{R}^{6\times6}$ terms a diagonal weighting matrix.

As usual in multi-objective control, the above scalar (λ's) and matrix (Λ's) weights are first set so as to balance the maximum or average values or the components of (8). Then, if necessary, they are fine-tuned.

IV. Visual Servoing via Point Cloud Based Pathways

Recall that the planner generates a series of waypoints $\{s_w\}_{w=1}^{W}$, each paired with a corresponding reference frame F_w and a point cloud Ω_w (see Figure 3). In this section, we describe an ICP based position based visual servoing (PBVS) method that uses these point clouds $\{\Omega_w\}$ to navigate sequentially from one waypoint to the next, ultimately converging at the last waypoint, which corresponds to the reference cutting pose $s_W = s_j^*$.

A. ICP Pose Estimation

As aforementioned, the ICP algorithm is used for vision based pose estimation. At each update cycle z of the ICP, the sensor-acquired point cloud Ω_{c_z} is aligned with the reference point cloud of the corresponding waypoint Ω_w to provide

$$^{w}H_{c_z} = \text{ICP}(\Omega_{c_z}, \Omega_w) \tag{19}$$

Since the ICP frequency is slower than the robot inner control frequency due to the computational load of processing large point clouds and complex alignments, a parallel process at the high inner control frequency is implemented to update an estimate $^{w}\hat{H}_c$ of the camera pose. This estimate is initialized with (19) at ICP cycle z, then updated at each inner sample time i between ICP cycles z and $z+1$ by the recurrence

$$^{w}\hat{H}_{c_i} \leftarrow {}^{w}\hat{H}_{c_{i-1}} ({}^{B}H_{c_{i-1}})^{-1} \, {}^{B}H_{c_i}. \tag{20}$$

At the next ICP cycle $z+1$, the registration is warm-started with the latest high-frequency estimate $^{w}\hat{H}_c$ given by (20) as its initial guess so as to improve convergence.

The pose vector ${}^{w}s_c$ of the camera relative to the next waypoint is obtained from ${}^{w}\hat{H}_c$, i.e., is set to

$${}^{w}s_c = \left[{}^{w}t_c^{\top}, {}^{w}\Theta_c^{\top}\right]^{\top}, \text{ with } {}^{w}t_c = {}^{w}\hat{t}_c \text{ and } {}^{w}\Theta_c = {}^{w}\hat{\Theta}_c. \quad (21)$$

Thanks to the incorporation of ICP, the method is expected to be robust to errors in the vine model, as the associated 3D model may contain imperfections such as holes or partially represented small branches.

B. Waypoint-to-camera setpoint definition

A smooth passage through a waypoint s_w at non-zero velocity can be ensured by redefining on-the-fly an appropriate visual servoing setpoint s^* for ${}^{w}s_c$. Inspired by [17], we propose

$$s^* = \left[{}^{w}t_c^{*\top}, 0_{3\times1}^{\top}\right]^{\top}. \quad (22)$$

The vector of desired orientation angles is set to zero in order to align the camera with s_w. The translation component ${}^{w}t_c^*$ is defined as

$${}^{w}t_c{}^{*} = {}^{w}t_c + \|\phi_t(s)\| \frac{\phi(s)}{\|\phi(s)\|} \quad (23)$$

where the current camera position ${}^{w}t_c$ relative to F_w is estimated by (20)–(21) and

$$\phi(s) = \beta_t \phi_t(s) + \beta_n \phi_n(s), \quad (24)$$

terms a vector field whose definition entails the waypoint s_w as well as its predecessor s_{w-1} and successor s_{w+1} (Figure 4). In fact, the tangential term

$$\phi_t(s) = \begin{cases} \text{Proj}_{{}^{w}t_c}(s_{w-1}s_w), & \text{if } \|\text{Proj}_{{}^{w}t_c}(s_{w-1}s_w)\| \geq \varepsilon_t, \\ \text{Proj}_{{}^{w}t_c}(s_{w-1}s_w) + \frac{1}{3}s_w s_{w+1}, & \text{otherwise.} \end{cases} \quad (25)$$

is first set to the projection of ${}^{w}t_c$ onto the line segment $s_{w-1}s_w$, and is then modified so as to drive the camera toward the next waypoint s_{w+1} if its norm falls below the specified threshold ε_t. The normal term $\phi_n(s)$ of the vector field $\phi(s)$ is set to the perpendicular vector from the current position ${}^{w}t_c$ to the line segment $s_{w-1}s_w$. $\beta_t > 0$ and $\beta_n > 0$ are scalar tuning parameters: augmenting β_n enhances transversal convergence towards $s_{w-1}s_w$, while β_t regulates the tangential velocity.

Upon reaching s_w within a pose error tolerance of ε_w ($\varepsilon_w < \varepsilon_t$), the waypoint index w is incremented to shift the reference forward along the path, and the associated point cloud Ω_w for ICP is updated accordingly. Once the robot reaches the final waypoint (cutting pose), (23) is switched to

$${}^{w}t_c^* = 0_{3\times1}. \quad (26)$$

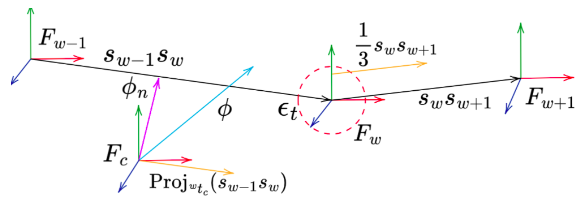


Fig. 4: Waypoint-to-camera setpoint definition: ϕ_t is defined along (25) (orange arrows); ϕ_t, ϕ_n are combined into ϕ (cyan) along (24); F_c is driven towards ϕ's direction, along (23).

By virtue of the PBVS controller (described in the next subsection), setting the setpoint to zero implies that the EOT will stop at the cutting pose.

C. PBVS Control

Position-based visual servoing (PBVS) is a widely used approach [18]. We define the 3D error function to be driven to zero as

$$e = \begin{pmatrix} {}^{w}t_c - {}^{w}t_c^* \\ \alpha({}^{w}R_c^{*\top}\,{}^{w}R_c) \end{pmatrix} \in \mathbb{R}^6, \quad (27)$$

where ${}^{w}s_c = \left[{}^{w}t_c^{\top}, {}^{w}\Theta_c^{\top}\right]^{\top}$ is obtained through (20)–(21); $s^* = \left[{}^{w}t_c^{*\top}, {}^{w}\Theta_c^{*\top}\right]^{\top}$ stands for its reference (22)–(23); and $\alpha(.)$ is defined in (10). To ensure an exponential decoupled decrease of the error, we select its closed-loop dynamics as

$$\dot{e} = -\kappa e \quad (28)$$

with $\kappa \in \mathbb{R}^{6\times6}$ a positive diagonal matrix gain. The control input vector (i.e., the camera velocity screw setpoint to the low-level robot controller) is then set to

$$v = -\kappa L^{-1} e \quad (29)$$

where $L \in \mathbb{R}^{6\times6}$ is the PBVS 3D interaction matrix [18].

TABLE I: Tuning parameters used in the experiments

Param	**Description**	**Value**
N	Prediction horizon length	2
ε	Tolerance for pose-error stopping	10^{-3}
Λ_{pose}	Pose weights $\|e(s_k)\|_{W_{\text{pose}}}$	$\text{diag}(10^6 I_3, 10^4 I_3)$
Λ_{term}	Terminal cost weights $\|e(s_N)\|_{W_{\text{term}}}$	$\text{diag}(10^8 I_3, 10^9 I_3)$
δ	Obstacle collision margin [m]	0.05
λ_{coll}	Collision-avoidance cost weight	10^8
α_{vis}	Visibility penalty	0.2
$n_{\min}$	Minimum visible points threshold	$0.15\lvert\Omega_{env}\rvert$
λ_{los}	Line-of-sight cost weight	$1e3$
$u_{\min/\max}$	Input constraints	$\pm(0.035\,\text{m}, 0.05\text{ rad})$
κ	Control gain in $\dot{e} = -\kappa e$	$(0.2, \ldots, 3.0)I_6$
β_n	Transversal convergence gain	2.5
β_t	Tangential convergence gain	0.5
d_{tool}	Tool offset along optical axis [m]	0.11
ε_t	Tangential update tolerance	0.04
ε_w	Waypoint switching tolerance	0.015

V. EXPERIMENTS ON REAL VINES

This section evaluates the proposed approach using an eye-in-hand RealSense D405 stereo camera mounted on a Franka Emika Panda robot (Figure 1, left), supported by ROS. The robot's controllers run at 1 kHz, while the camera operates in free-running mode. To build the global vine model, we applied a multiway registration method [19] to iteratively align multiple point clouds of the vine's geometry in the robot base frame. During online operation, the point cloud is processed in C++ with PCL [20], applying filtering and downsampling for efficiency. A textured background minimizes noise, as point cloud processing is not the focus of this paper. The PBVS and NMPC are respectively implemented with VISP [21], and NLopt [22]. Experiments assess performance on simple and complex vines, moving vines, and waypoint navigation under low visibility. The tuning parameters in Table I have been utilized for all experiments.

A. *General cases*

This section evaluates the method's ability to sequentially reach multiple cutting poses. Fig. 5 illustrates the obtained collision-free 3D trajectory of a cutting tool in charge of two pruning operations. It confirms the accurate alignment of the point clouds at both cutting poses. The planned waypoints are sequentially crossed until the respective cutting poses are reached, ensuring convergence despite significant differences between the initial and target point clouds.

Further experiments on a denser vine (Fig. 1, left white circle) are detailed in Fig. 6. The results validate the method's efficiency across varying initial errors, as the robot continuously navigates without stopping (unlike classical PBVS) through waypoints to successfully reach all four cutting poses. However, a slight final static error (especially in rotation) results from limited ICP precision, sensitivity to noise, point-cloud fluctuations, and mismatch between the model and real point cloud.

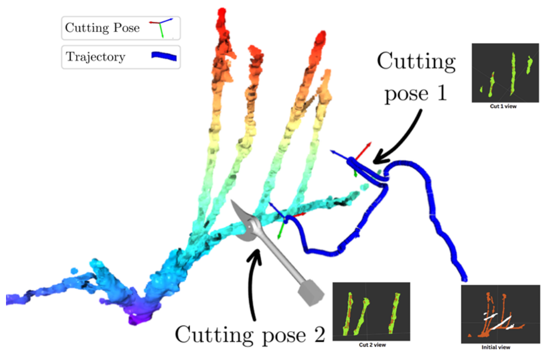


Fig. 5: The 3D robot's camera trajectory. The images show the initial (red) and target (white) point clouds; the green-red fusion indicates the final alignment at the goal pose.

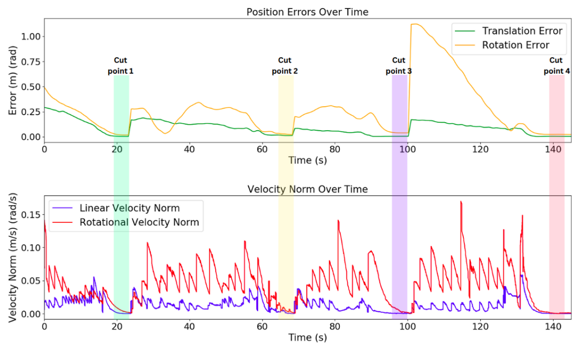


Fig. 6: (Top) Translation+Rotation errors & (Bottom) Linear+angular velocity control signals *vs* time for a dense vine.

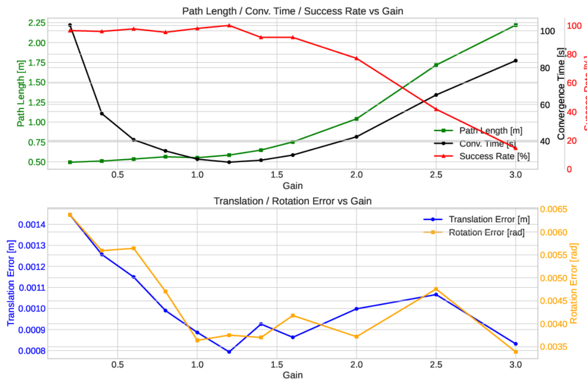
B. *Quantitative tests*

Quantitative evaluations were conducted on seven dense vines (Fig. 7a). These presented various shapes: asymmetric, symmetric, with some isolated branches, etc. Two vines served for parameter tuning (4 cutting poses $\times$ 3 initial poses $\times$ 2 runs). The remaining vines were tested under the same protocol, utilizing for the control gain κ either the tuned values or larger gains. Table II summarizes the results on the total of 1008 cuts.

The method demonstrates effective generalization with low tuning gains, achieving $\approx 95\%$ success rate. Average Final Static Errors (FSEs) are low (translation < 1.5 mm, rotation $< 0.4°$) and tightly distributed. Convergence times and path lengths remain consistent with those of the tuning dataset. Fig. 7b shows that a moderate increase in gain κ reduces convergence time (homographically, as predicted by theory) and minimizes steady-state errors (likely due to sharper error decay).

However, increasing κ improves convergence and error performance but reduces robustness to model mismatch and limited ICP update rate. This is confirmed by the second test set. Success rates decrease with κ (73% on average), and oscillations increase path length and convergence time. Given the large variability in FSEs, their reported mean values should be interpreted cautiously. Average planning time is 2.4 seconds. The planner is sometimes trapped in local minima, leading to failures.

(a)



(b)

Fig. 7: (a) Three of the seven vines used of the vines used for the quantitative analysis. (b) Results of the system performance for different control gains κ.

1008 cuts in 7 different dense branches						
Dataset	**Gain κ**	**Trans FSE(m)**	**Rot FSE(rad)**	**Path Len(m)**	**Conv Time(s)**	**Successful Cuts**
Tuning	0.2–0.8	0.00113	0.00382	0.51769	53.286	192/192
Test 1	0.2–0.8	0.00124	0.00632	0.52960	60.594	456/480
Test 2	1.0–3.0	0.00091	0.00386	1.07362	44.662	245/336
Average&Total		0.00109	0.00467	0.70697	52.847	893/1008

TABLE II: Quantitative test results for varying control gains.

C. Unexpected movements between the robot and the vine

The primary advantage of our approach lies in its ability to handle unexpected variations that are likely to arise in real-world scenarios. For example, movements of the vine or the robot between model acquisition and the cutting task can introduce discrepancies. To assess the robustness of the method against such model inaccuracies, we applied a 4 cm translation between the vine's modeled position and its actual location. This perturbed model was then tested using both an open-loop controller and the proposed method. Figure 8 illustrates the resulting trajectories: the red trajectory, generated by the open-loop strategy, fails to fulfill the task due to the model mismatch. On the other hand, the green trajectory, produced by our ICP-PBVS approach (with visual feedback), dynamically compensates for the motion and successfully reaches the target. This capability is enabled by our method's reliance on local rigidities within the vine structure, despite relative displacements between the robot and the vine. Further experiments concerning movements of the vines and the robot base are provided in the accompanying video.

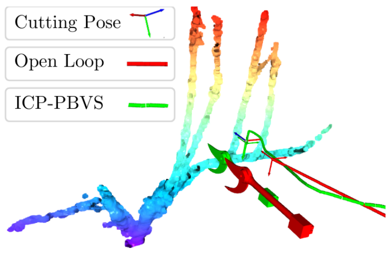
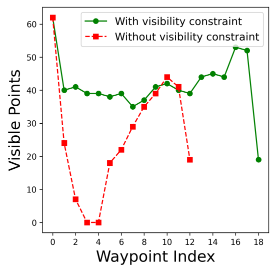


Fig. 8: Comparison of the trajectories reaching the goal.

D. Poor visibility case

A critical challenge is maintaining the vine within the camera's field of view when transitioning between cutting poses, particularly in low-visibility areas. As visual feedback is essential, the visibility constraint is crucial for effective visual servoing, ensuring a continuous, relevant visual path.

Figure 9b highlights a low-visibility region of the vine, where no branches are present, while the robot is required to navigate from right to left. Figure 9a quantifies the impact of visibility constraints by showing the number of visible points (i.e., the size of the visible point cloud) at each waypoint during the planning phase. Without the visibility constraint, there is a marked reduction in visible points within the low-visibility area, compromising the convergence of the ICP-PBVS algorithm. In contrast, enforcing the visibility constraint ensures a stable number of visible points throughout the trajectory, albeit at the cost of an increased number of waypoints as the planner adjusts the path to maintain continuous visibility. This experiment underscores the critical role of visibility constraints in ensuring the reliability of the approach, particularly in challenging environments where visual feedback is intermittent. By addressing this limitation, our method demonstrates robust performance, enabling successful task execution even under complex and adverse conditions.



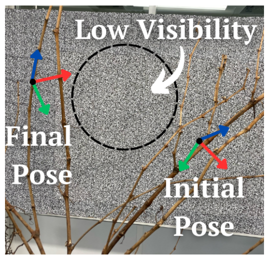
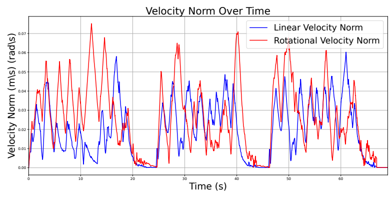


Fig. 9: (a) Visibility comparison with and without visibility constraint. (b) The low visibility area to traverse.

E. Outdoor experiments under strong wind

To evaluate performance under strong wind, experiments were conducted outdoors on a day with natural wind (38 km/h), supplemented by an auxiliary blower to increase localized wind intensity. The setup (Figure 10*a*) used natural vegetation as the background, reflecting real-world conditions. Figure 10*b* confirms that the system reached all three target positions (Avg. Trans FSE: 0.003 m, Avg. Rot FSE: 0.011 rad). It is interesting to notice at the cutting times (around $t = [22, 45, 65]$ seconds) how the velocity undergoes small variations in order to couple with the branch movements induced by strong wind. These results demonstrate the system's ability to operate effectively in challenging, high-wind outdoor environments.

(a) Setup with strong wind.



(b) Velocity profiles.

Fig. 10: Outdoor experiments: (a) setup (with another orientation of the cutting tool); (b) velocity profiles.

VI. CONCLUSION & DISCUSSION

Vision based control for sequentially reaching multiple target poses using point clouds is addressed, with a specific application to vine pruning. An NMPC based planner is proposed to generate a sequence of waypoints guiding the robot toward the desired cutting poses. Subsequently, a time-varying reference ICP-PBVS computes the robot's velocity control input, ensuring smooth transitions through the waypoints without stopping. The proposed approach enables reactive waypoint navigation through the complex vine structure while maintaining robustness against model inaccuracies, low visibility conditions, and potential collisions. Extensive experiments conducted with a Franka robot across various vine structures demonstrate the method's effectiveness. The results highlight its advantages over open-loop controllers (with no exteroceptive feedback), particularly in real-world scenarios where unexpected relative movements occur between the vine and the robotic system. In addition, conclusive tests were conducted outdoor, under strong wind conditions.

As the method utilizes point cloud data, it is compatible with other real-time sensors such as LiDARs. Yet, some limitations remain. As waypoint planning and visual based navigation are executed in sequence, the planner can only cope with obstacles detected prior to robot motion. If an unexpected obstacle is detected while the robot is moving, then either replanning would be required—which, while fast, may be inconvenient and disrupts the task flow—or specific visual servoing techniques should be implemented. Besides, the controller's dependency on ICP-like algorithms makes it sensitive to initialization. Furthermore, the high computational cost limits the controller's frequency and achievable gains κ, and thus impacts the overall task time. GPU implementations of learning-based point cloud registration and/or servoing techniques may be of interest.

Last, a solution to reduce the overall task duration is envisaged, which consists in the sequence of two stages: the open-loop execution of preplanned trajectories, i.e., with no exteroceptive feedback, followed by the proposed visual based method in the vicinity of the vine to make the best of its performance and robustness. The transition between the two stages must be carefully studied: automatic switching time, control signal smoothness, etc. Future work will further investigate waypoint planning. First, NMPC will be benchmarked against conventional strategies (A^*, RRT, etc.). Then, as an alternative to the proposed offline synthesis of a full waypoint sequence after each cut, a "minimal" NMPC planner will be implemented so as to generate online two waypoints ahead every time the robot reaches a waypoint. Extensive tests will be conducted in commercial vineyards.

References

[1] Spyros Fountas, Nikos Mylonas, Ioannis Malounas, Efthymios Rodias, Christoph Hellmann Santos, and Erik Pekkeriet, "Agricultural robotics for field operations," *Sensors*, vol. 20, no. 9, pp. 2672, 2020.

[2] Luiz FP Oliveira, António P Moreira, and Manuel F Silva, "Advances in agriculture robotics: A state-of-the-art review and challenges ahead," *Robotics*, vol. 10, no. 2, pp. 52, 2021.

[3] Chung Hee Kim, Abhisesh Silwal, and George Kantor, "Autonomous robotic pepper harvesting: Imitation learning in unstructured agricultural environments," *arXiv preprint arXiv:2411.09929*, 2024.

[4] Qinghui Pan, Dong Wang, Jie Lian, Yongxiang Dong, and Chaochao Qiu, "Development of an automatic sweet pepper harvesting robot and experimental evaluation," in *IEEE Int. Conf. on Robotics and Automation (ICRA'2024)*.

[5] Ege Gursoy, Benjamin Navarro, Akansel Cosgun, Dana Kulić, and Andrea Cherubini, "Towards vision-based dual arm robotic fruit harvesting," in *IEEE Int. Conf. on Automation Science and Engineering (CASE'2023)*.

[6] Sotiris Stavridis, Leonidas Droukas, and Zoe Doulgeri, "Bimanual grape manipulation for human-inspired robotic harvesting," *IEEE/ASME Transactions on Mechatronics*, 2024.

[7] Christian Lenz, Rohit Menon, Michael Schreiber, Melvin Paul Jacob, Sven Behnke, and Maren Bennewitz, "Hortibot: An adaptive multi-arm system for robotic horticulture of sweet peppers," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'2024)*.

[8] T. Botterill, S. Paulin, R. Green, S. Williams, J. Lin, V. Saxton, S. Mills, X. Chen, and S. Corbett-Davies, "A robot system for pruning grape vines," *Journal of Field Robotics*, vol. 34, no. 6, 2017.

[9] A. Silwal, F. Yandun, A. Nellithimaru, T. Bates, and G. Kantor, "Bumblebee: A path towards fully autonomous robotic vine pruning," *Field Robotics*, , no. 2, 2022.

[10] Miguel Fernandes, Juan D Gamba, Francesco Pelusi, Angelo Bratta, Darwin Caldwell, Stefano Poni, Matteo Gatti, and Claudio Semini, "Grapevine winter pruning: Merging 2d segmentation and 3d point clouds for pruning point generation," *Computers and Electronics in Agriculture*, vol. 237, pp. 110589, 2025.

[11] Tao Teng, Miguel Fernandes, Matteo Gatti, Stefano Poni, Claudio Semini, Darwin Caldwell, and Fei Chen, "Whole-body control on non-holonomic mobile manipulation for grapevine winter pruning automation," in *IEEE Int. Conf. on Advanced Robotics and Mechatronics (ICARM'2011)*.

[12] F. Yandun, T. Parhar, A. Silwal, D. Clifford, Z. Yuan, G. Levine, S. Yaroshenko, and G. Kantor, "Reaching pruning locations in a vine using a deep reinforcement learning policy," in *IEEE Int. Conf. on Robotics and Automation (ICRA'2021)*, Xi'an, China.

[13] S. Katyara, F. Ficuciello, D.G. Caldwell, F. Chen, and B. Siciliano, "Reproducible pruning system on dynamic natural plants for field agricultural robots," in *Int. Work. on Human-Friendly Robotics 2020*.

[14] Henry Williams, David Smith, Jalil Shahabi, Trevor Gee, Ans Qureshi, Ben McGuinness, Scott Harvey, Catherine Downes, Rahul Jangali, Kale Black, et al., "Archie jnr: A robotic platform for autonomous cane pruning of grapevines," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'2024)*.

[15] Gaoyuan Liu, Bas Boom, Naftali Slob, Yuri Durodié, Ann Nowé, and Bram Vanderborght, "Automated behavior planning for fruit tree pruning via redundant robot manipulators: Addressing the behavior planning challenge," *IEEE Robotics & Automation Magazine*, 2025.

[16] Fadi Gebrayel, Martin Mujica, and Patrick Danès, "Visual servoing for vine pruning based on point cloud alignment," in *Int. Conf. on Informatics and Control, Automation and Robotics (ICINCO'2024)*.

[17] Andrea Cherubini and François Chaumette, "Visual navigation with a time-independent varying reference," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'2009)*.

[18] François Chaumette and Seth Hutchinson, "Visual servo control. I. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[19] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun, "Robust reconstruction of indoor scenes," in *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR'2015)*.

[20] Radu Bogdan Rusu and Steve Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE Int. Conf. on Robotics and Automation (ICRA'2011)*, Shanghai, China.

[21] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for visual servoing: A generic software platform with a wide class of robot control skills," *IEEE Robotics & Automation Magazine*, vol. 12, no. 4, 2005.

[22] Steven G. Johnson, "The NLopt nonlinear-optimization package," https://github.com/stevengj/nlopt, 2007.