Autonomous Guidance of an Aerial Robot using 6-DoF Visual Control and Behavior Trees

Raul Tapia
GRVC Robotics Lab
Universidad de Sevilla
Seville, Spain
raultapia@us.es

Miguel Gil-Castilla
GRVC Robotics Lab
Universidad de Sevilla
Seville, Spain
mgil8@us.es

José Ramiro Martínez-de Dios
GRVC Robotics Lab
Universidad de Sevilla
Seville, Spain
jdedios@us.es

Anibal Ollero
GRVC Robotics Lab
Universidad de Sevilla
Seville, Spain
aollero@us.es

***Abstract*—The use of unmanned aerial vehicles in inspection and maintenance tasks reduces risks to human operators, increases the efficiency and quality of operations, and reduces the associated economic costs. Accurate navigation is crucial for automating these missions. This paper introduces an autonomous guidance scheme for aerial robots based on behavior trees that combine Position-Based Visual Servoing (PBVS) and waypoint following. The behavior tree dynamically adapts the robot navigation strategy depending on the required level of accuracy, accommodating to different types of missions and the presence of obstacles. As a result, the proposed navigation scheme enables complex missions to be performed in tight environments with significant levels of safety and robustness. All functional modules have been integrated into a behavior tree architecture, which provides robust, reactive, modular, and flexible frameworks to handle high-level tasks in complex missions. Validation using extensive software-in-the-loop simulations has demonstrated the robustness and accurate performance of the method, confirming its potential for real-world applications. An open-source implementation of our method is released under the GNU GPL3 license.**

***Index Terms*—autonomous vehicles, aerial robots, visual servoing, behavior trees**

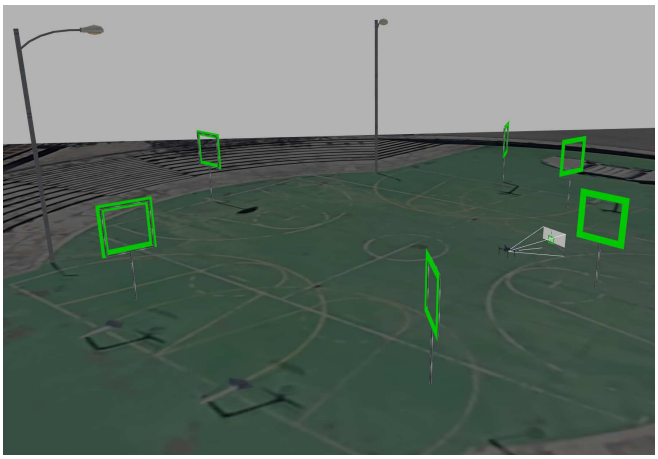

Fig. 1. Simulation setup for the evaluation of the proposed aerial robot navigation method.

I. Introduction

Aerial robots have shown to be extremely useful in a wide variety of applications. The use of Unmanned Aerial Vehicles (UAVs) is particularly interesting in inspection & maintenance [1], [2], manipulation [3], surveillance [4], [5], or packet delivery [6], among many others, due to their capability to access hard-to-reach areas without compromising human safety. For instance, the use of aerial robots for the inspection of infrastructures (such as bridges [7], powerlines [8] [9], or industrial settings [10]) strongly reduces the risk level of inspection operators and also inspection times, involving cost and operational advantages.

To perform inspection missions robustly and safely in complex and dynamic environments, many UAV navigation schemes rely on autonomous guidance methods based on onboard perception. Cameras are widely used to perceive the environment in aerial robots due to their ease of integration and low cost compared to other sensors such as LIDARs or SONARs, among others. Visual servoing, or vision-based control, leverages visual information to control the motion of the UAV in real-time. Many works have demonstrated the suitability of visual servoing methods on board aerial robots in a wide variety of applications. The work [11] presents an onboard vision-based control scheme to track an intruder (i.e., another UAV) flying arbitrarily. The authors in [12] propose a robust Image-Based Visual Servoing (IBVS) scheme for quadrotors that perform aggressive maneuvers while ensuring target visibility. A visual tracking method is presented in [13] for the inspection of linear infrastructures using an autonomous quadrotor. The work [14] proposes a method based on deep neural networks to tune visual servoing systems on UAVs. Visual guidance is used not only onboard multirotors. The work [15] presents a vision-based control framework to track a moving target using a fixed-wing UAV with a monocular pan-tilt camera. The authors of [16] propose a visual servoing approach to track windows in an urban environment from a helicopter. The work [17] uses a visual servoing method based on features extracted with an event camera on board a flapping-wing aerial robot [18]–[20]. Although the previous works present an adequate performance for the mission they propose, none of them focuses on the use of high-level decision-making schemes that enable the integration of different functional modules and security checks.

Visual servoing can be combined with other navigation schemes such as waypoint following. These very different

navigation strategies should be seamlessly integrated through suitable mission control schemes that enable the safety, robustness, and efficiency in mission execution. Behavior Trees (BTs) are flexible software design architectures for decision-making processes in autonomous systems [21], [22]. They are widely used in robot navigation [23], manipulation [24], and robot swarm coordination [25], [26], due to their ease of integration, modularity (complex tasks are composed from simple reusable tasks), and robustness (via condition checks). Robots controlled using behavior trees can perform complex tasks using straightforward and well-defined rules and retain the ability to respond to environmental changes and unforeseen events. The work [27] shows that employing BT architectures can enhance the modularity, reusability, and complexity of UAV guidance and control systems. The work [28] addresses the process of replacing the battery and manually restarting an inspection mission is automated using behavior trees, providing an effective means for autonomous mission control and supervision, eliminating the need for human intervention. The authors in [29] present a behavior tree as a decision control mechanism included in the flight control system.

This paper presents an autonomous guidance method for aerial robots based on behavior trees that leverages Position-Based Visual Servoing (PBVS) to accurately guide the robot towards specific navigation targets. The behavior tree scheme dynamically changes the aerial robot control strategy depending on the required level of navigation accuracy: 1) when a target or obstacle is not visible, the navigation is based on waypoint following (e.g. using GNSS), and 2) when a target or obstacle is visible, it uses PBVS for precise navigation enabling obstacle avoidance or performing complex maneuvers accurately. As a result, the proposed scheme enables high levels of robustness and safety, using visual servoing when necessary to increase navigation accuracy. Behavior trees provide modularity, ease of integration, and condition checks that make sure that the status of the UAV or its battery level are enough to perform the task. As a result, they provide high levels of safety and robustness that enable UAVs to perform complex missions. Our work uses PBVS, which, in contrast to Image-Based Visual Servoing (IBVS) can operate robustly even if the target comes out of the camera field of view during the execution of the maneuver, hence providing high robustness. The proposed guidance scheme has been extensively validated in realistic simulations under different conditions, evidencing its suitability for complex missions.

The main contributions of this work are: 1) a PBVS-based method for autonomous guide of an aerial vehicle and 2) its integration together with waypoint following in a behavior tree architecture for safe operations. Our approach aims to provide a novel flexible framework in which visual-based control is integrated in a modular fashion, increasing the safety and robustness of the system while enabling its scalability to different missions and scenarios. The method has been extensively validated on realistic simulations. In addition, an open source implementation[1][2] of our method has been released under the GNU General Public License v3.0.

The rest of the paper is organized as follows. Section II describes the proposed method. The experimental evaluation is presented in Section III. Finally, Section IV closes the paper and highlights the main future work.

II. Method Description

Our objective is to automatically guide an aerial robot toward an indeterminate number of sequentially arranged targets by combining two navigation strategies: waypoint following and PBVS guidance. To illustrate the capabilities of the proposed method, we address the challenging problem of navigating through a set of narrow windows located at not fully known positions. Each of the windows acts as a target. The proposed method does not require the exact location of each target, it requires only an approximate location from which each target (window) can be detected in the aerial robot images. These locations are taken as waypoints in waypoint following navigation. The proposed scheme relies on behavior trees to manage the mission at a high level, controlling various condition checks (e.g., UAV status or battery level) and dynamically deciding the navigation strategy depending on the case. When no windows are visible in the aerial robot image, the behavior tree selects a navigation strategy based on waypoint following. When a window is visible, to have an accurate navigation strategy that enables crossing through the narrow window, the behavior tree selects PBVS guidance using the window as target. The highest navigation accuracy is required during the maneuvers of crossing through the windows. During these maneuvers, the UAV is controlled with PBVS using 6 DoF (Degree of Freedom) pose estimations of the UAV computed only with the onboard camera, being insensitive to the noise level in the GNSS measurements and even being capable of adapting to moving windows.

The developed PBVS method is described in Section II-A. Section II-B summarizes the proposed behavior tree scheme that combines PBVS and waypoint following.

A. Vision-based Control

PBVS relies on computing the relative pose that guides the robot toward the desired final pose using the projection of the target points onto the image plane. It is critical to accurately calculate the pose of the target with respect to the robot to ensure an accurate and efficient trajectory toward the target. The used reference frames are presented in Figure 2. ${}^{A}\mathbf{T}_{B}$ denotes the transformation from frame $\{A\}$ to frame $\{B\}$. Our PBVS method operates under three hypotheses: 1) the target geometry (a window of height H, width W and thickness Δ) and color are known –although they may vary from one target to another; 2) camera intrinsic parameters and distortion coefficients are known; and 3) camera-robot extrinsic calibration (${}^{C}\mathbf{T}_{R} = {}^{C^\star}\mathbf{T}_{R^\star}$) is known.

[1] https://github.com/raultapia/bt_pbvs
[2] https://github.com/miggilcas/grvc_bt_ros

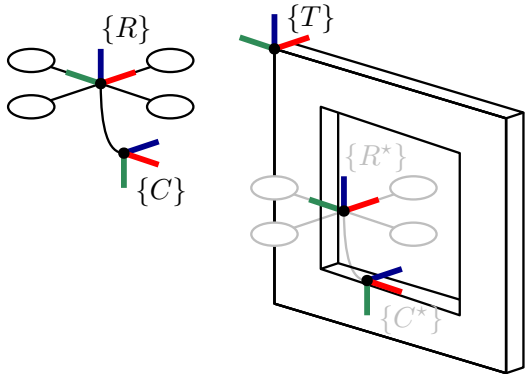

Fig. 2. Reference frames used in PBVS: R - robot, C - camera, T - target. Axis: red - x, green - y, blue - z.

Target pose estimation

Target pose with respect to the camera frame (i.e. ${}^{C}\mathbf{T}_{T}$) is estimated from the target geometry, observed features, and the camera intrinsic parameters. For simplicity, we use the four external corners of the window in the image plane as features. The target window is detected using color thresholding in the acquired images. The HSV ranges for thresholding are selected to achieve robustness against small variations in the target color (caused by, e.g., illumination changes). The four corners are detected by approximating the contour of the selected region by a 4-side polygon. We only consider regions with suitable geometric properties. Once the 4 corners (u_1, v_1), (u_2, v_2), (u_3, v_3), and (u_4, v_4) are detected, ${}^{C}\mathbf{T}_{T}$ is estimated by solving a Perspective-n-Point (PnP) problem. A point $({}^{T}X, {}^{T}Y, {}^{T}Z) \in \mathbb{R}^3$ expressed w.r.t. frame $\{T\}$ is projected in the image as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \cong \mathbf{K}[{}^{C}\mathbf{R}_{T} | {}^{C}\mathbf{t}_{T}] \begin{bmatrix} X_T \\ Y_T \\ Z_T \\ 1 \end{bmatrix} \text{ with } {}^{C}\mathbf{T}_{T} = \begin{bmatrix} {}^{C}\mathbf{R}_{T} & {}^{C}\mathbf{t}_{T} \\ \mathbf{0}^{T} & 1 \end{bmatrix}. \quad (1)$$

We solve Equation 1 using non-linear Levenberg-Marquardt optimization to minimize the reprojection error. Since our target features form a plane, the initial solution is obtained by homography decomposition.

Position-Based Visual Servoing

A PBVS system computes the required camera movement from its initial unknown pose to a goal pose. Assuming that ${}^{C}\mathbf{T}_{T}$ has been precisely estimated, the camera should travel from its current pose to achieve a desired relative robot-target pose ${}^{C^\star}\mathbf{T}_{T}$:

$${}^{C^\star}\mathbf{T}_{T} = {}^{C^\star}\mathbf{T}_{C} \cdot {}^{C}\mathbf{T}_{T}. \quad (2)$$

Hence, the change in the camera pose should be:

$${}^{C}\mathbf{T}_{C^\star} = {}^{C}\mathbf{T}_{T} \cdot {}^{T}\mathbf{T}_{C^\star}. \quad (3)$$

For convenience, as we command the robot w.r.t. its body frame $\{R\}$, the previous expression can be rearranged as follows:

$$\begin{aligned} {}^{R}\mathbf{T}_{R^\star} &= {}^{R}\mathbf{T}_{C} \cdot {}^{C}\mathbf{T}_{C^\star} \cdot {}^{C^\star}\mathbf{T}_{R^\star} = \\ &= {}^{R}\mathbf{T}_{C} \cdot {}^{C}\mathbf{T}_{T} \cdot {}^{T}\mathbf{T}_{C^\star} \cdot {}^{C^\star}\mathbf{T}_{R^\star} = \\ &= {}^{R}\mathbf{T}_{C} \cdot {}^{C}\mathbf{T}_{T} \cdot {}^{T}\mathbf{T}_{R^\star}. \end{aligned} \quad (4)$$

Therefore, the pose command can be computed from the camera-robot extrinsic calibration ${}^{R}\mathbf{T}_{C}$ (known), ${}^{C}\mathbf{T}_{T}$ (output of PnP), and the desired target pose ${}^{T}\mathbf{T}_{R^\star}$ (goal).

Since in our mission the aerial robot should traverse through the window, we select the transformation ${}^{T}\mathbf{T}_{R^\star}$ such that the final robot location in the Y and Z coordinates of frame $\{T\}$ corresponds to the center of the window and the X coordinate is $\Delta + \delta$, being Δ the window thickness and $\delta \in \mathbb{R}$ is the final robot displacement after crossing the window:

$${}^{T}\mathbf{T}_{R^\star} = \begin{bmatrix} 1 & 0 & 0 & \Delta + \delta \\ 0 & 1 & 0 & -0.5W \\ 0 & 0 & 1 & -0.5H \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

To increase accuracy, the target is detected multiple times while the UAV is executing the trajectory. The same procedure is executed to modify the pose reference.

One of the main advantages of our approach is that the last command ${}^{R}\mathbf{T}_{R^\star}$ given to the robot remains as a valid reference for the low-level control of the UAV after the target is no longer visible. One suitable strategy is to stop running PBVS when the extracted features are near the edges of the image (i.e., close to moving out of the frame). Further, smoother trajectories can be achieved by commanding intermediate reference poses. For that, an interpolation between $\mathbf{I}$ and ${}^{R}\mathbf{T}_{R^\star}$ is computed. Linear interpolation is used for translation $\mathbf{t} \in \mathbb{R}^3$ and SLERP for rotation $\mathbf{q} \in \mathbb{H}$ (expressed as a quaternion for convenience):

$$\mathbf{t}' = \lambda \mathbf{t}, \qquad \mathbf{q}' = \mathbf{q}^{\lambda}, \quad (6)$$

with $\lambda \in [0, 1]$.

B. Mission Control based on Behavior Trees

A behavior tree is a hierarchical structure of nodes, each representing a specific task. Decision-making processes and complex tasks can be decomposed into simple reusable components. The nodes in a behavior tree are divided into: 1) *control flow* nodes to select the execution flow (*Sequence*, *Fallback*, *Parallel*, or *Decorator*) and; 2) *execution* nodes to execute actions or verify conditions. *Sequence* nodes execute child nodes in order until one fails, *Fallback* nodes execute child nodes in order until one succeeds, *Paralell* make possible the concurrent execution of several nodes, and *Decorator* nodes modify the behavior of a child node, such as repeating it or adding conditions. *Execution* nodes return `SUCCESS`, `RUNNING`, or `FAILURE` after being executed, which is used as a feedback for the operation of the behavior tree. The behavior tree used for mission high-level control for a scenario with n

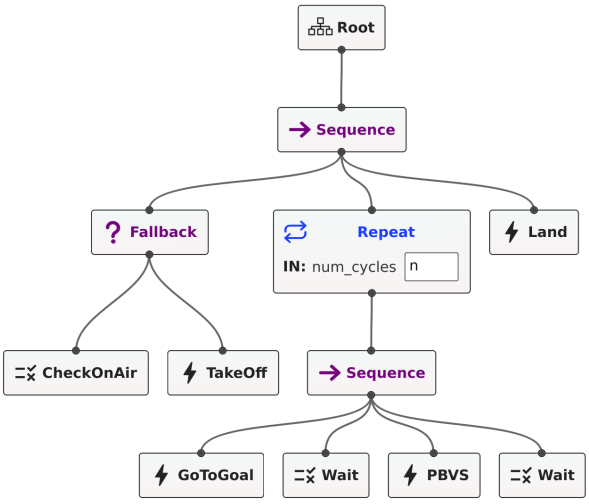

Fig. 3. Behavior tree used for mission high-level control for a scenario with n targets. *Repeat* is a type *Decorator* representing the loop execution of the *Sequence* downstream. Actions: **`TakeOff`**, **`GoToGoal`** (i.e., waypoint guidance), **`PBVS`** (i.e., visual guidance), and **`Land`**. Conditions: ***`CheckOnAir`*** (`SUCCESS` if UAV is ready to fly) and ***`Wait`*** (`SUCCESS` if UAV on hovering).

targets is shown in Figure 3. It establishes four actions that specify the operation that must be performed by the UAV:

- **`TakeOff`**: Before starting the mission the UAV takes off if it is not in the air.
- **`GoToGoal`**: This action guides the UAV to a specified waypoint using low-level pose control.
- **`PBVS`**: This action directs the UAV using the PBVS method described in Section II-A.
- **`Land`**: When the mission is completed, the UAV lands.

Behavior trees also have conditions. If the condition is not satisfied, the next action in the *Sequence* is not executed. The proposed behavior tree establishes two conditions:

- ***`CheckOnAir`***: If the robot has taken-off properly and is ready for flying, it returns `SUCCESS`.
- ***`Wait`***: If $\|\mathbf{v}\| \approx 0$ with $\mathbf{v} \in \mathbb{R}^3$ the linear velocity of the UAV, it returns `SUCCESS`.

In both conditions, the status of the UAV and of the battery is also checked. If one or more of these conditions are not fulfilled, it returns `FAILURE`.

III. Evaluation

The proposed navigation scheme has been extensively validated in realistic simulations under different conditions (set-ups, initial and final mission positions, IMU and GNSS noise level). The simulation was designed to facilitate migration to real robots using Software In The Loop (SITL) to simulate MAVROS-compatible UAVs. The aerial platform was simulated using the well-known Intelligent Quads[3], which is based on Ardupilot and hosts Gazebo worlds for various scenarios and UAV configurations. We used realistic Gazebo scenarios that replicate the GRVC Outdoor testbed *Plaza del Agua*[4], see Figure 1. The scenario includes obstacles such as lampposts and also 6 windows of sizes 1.8×1.8 m that act as targets in the proposed navigation method. The UAV control software, including the autopilot firmware and navigation and vision algorithms, is executed in a simulated environment. Ardupilot establishes communication with the simulator to obtain sensor data from the environment and transmits actuation signals. For the implementation of the method, we used the `BehaviorTree.CPP` library for the design of the behavior tree architecture and `OpenCV` and `Eigen3` for the PBVS.

The workflow of the behavior tree shown in Figure 3 is initialized with the initial UAV pose and the poses from which the six targets are visible in the UAV camera image. These poses (`Goal_1` - `Goal_6`) are used as reference for waypoint following by the **`GoToGoal`** node of the behavior tree. Next, in the **`TakeOff`** stage, a *Fallback* node first checks if the UAV has already taken off using the ***`CheckOnAir`*** condition. If the UAV is not in the air, it proceeds with the **`TakeOff`** action to reach the initial altitude. Once in the air, the UAV enters the navigation stage, in which the UAV repeats for each window `i` the following *Sequence*:

1) **`GoToGoal`**, using waypoint following the UAV reaches `Goal_i`,
2) **`Wait`**, the UAV waits until its velocity is approximately zero (i.e, $\|\mathbf{v}\| \approx 0$),
3) **`PBVS`**, after the UAV has completely stopped at `Goal_i` (from which window is visible in the UAV camera images), it starts navigation using visual servoing to guide the UAV to cross the window,
4) **`Wait`**, the UAV waits again until $\|\mathbf{v}\| \approx 0$.

We opt to wait for hovering for two reasons: 1) to ensure that control –either waypoint-based or vision-based– has converged, and 2) to add a further layer of safety by initiating trajectories from a very low kinetic energy state. Although beyond the scope of this work, the total operation time can be reduced by increasing the speed threshold. The sequence is repeated for every of the 6 windows considered in this set-up. During the mission, if some module returns `FAILURE`, the mission is aborted, and the UAV will hover to recover the control manually, as it would be in a real experiment.

The results obtained in the execution of a mission are presented. The mission consisted on completing three consecutive laps, in each of which the UAV crossed through the 6 windows. After completing the three laps, the UAV lands at the specified landing position. Figure 4 shows the UAV trajectory along one of the missions. The UAV position and yaw angle along the mission are shown in Figure 5. The transitions between waypoint following and PBVS guidance can be noticed since the UAV velocity in those transitions was zero. The transitions can also be easily noticed in Figure 6, which depicts the linear velocities of the UAV along that mission. A video[5] of the execution of the mission is also available.

[3] `https://github.com/Intelligent-Quads`

[4] `https://grvc.us.es/newweb/infraestructures/`

[5] `https://youtu.be/PSM_HQzml1g`

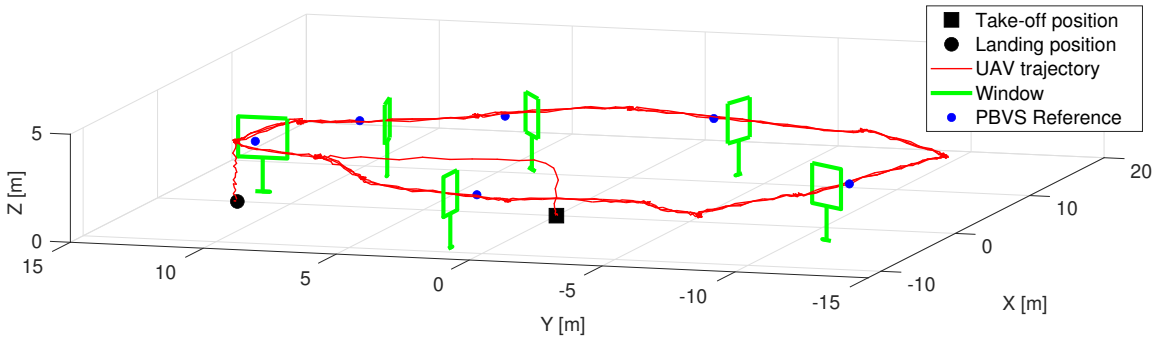

Fig. 4. UAV trajectory in a mission consisting in three consecutive laps, in each of which the UAV crossed through 6 windows. UAV 3D trajectory is depicted in red. Position references for PSBV are depicted as blue points. The trajectory includes take-off and landing.

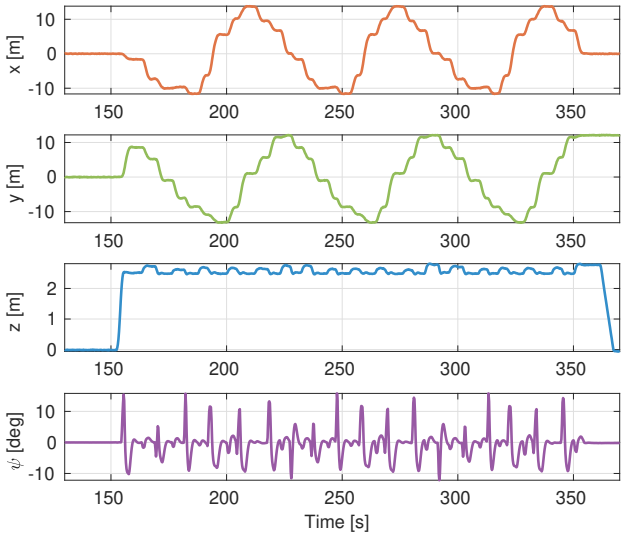

Fig. 5. UAV position and yaw angle along the mission shown in Figure 4 (three consecutive laps, each of them with 6 targets). Sequence: take-off, first lap (t=160 s to t=225 s), second lap (t=225 s to t=290 s), third lap (t=290 s to t=350 s), landing.

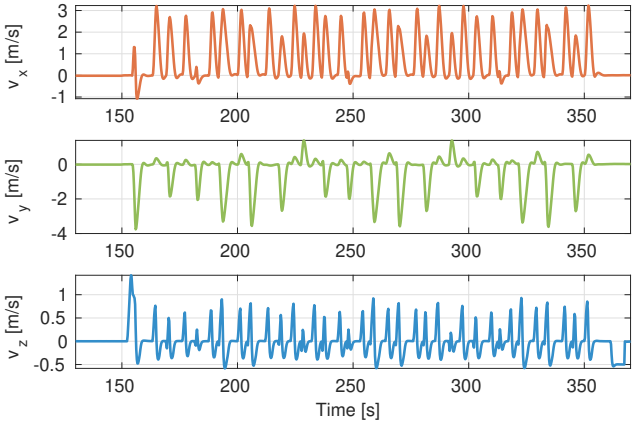

Fig. 6. Linear velocities of the UAV along the mission shown in Figure 4 (three consecutive laps, each of them with 6 targets). The transitions from waypoint following to PBVS guidance can be easily noticed.

The proposed navigation system has been extensively validated in multiple missions with different numbers of windows (from 4 to 20) at different locations and different take-off and landing positions. In all cases, the mission was satisfactorily accomplished, with all the windows passed through and all the missions completed without collisions. The IMU and GNSS noise levels in the simulation were also increased to assess the robustness of the approach. It should be noted that the highest navigation accuracy is required during the maneuvers to cross the windows. During these maneuvers, the UAV is controlled with PBVS using 6 DoF pose estimations of the UAV computed only with the onboard camera, and hence the method is insensitive to the noise level in the GNSS measurements. In addition, PBVS enables the UAV control to adapt to dynamic changes in the environment. For instance, it can cope with moving targets as far as they are in the camera field of view at the time when PBVS started. During the execution of the maneuver, PBVS estimates the relative 6 DoF pose of the UAV w.r.t. the window, hence being capable of intrinsically accommodating windows with slow motion.

IV. Conclusions and Future Work

This paper presented an autonomous guidance scheme for aerial robots based on behavior trees that combine Position-Based Visual Servoing (PBVS) and waypoint following. The behavior tree dynamically changes the aerial robot control strategy depending on the required level of navigation accuracy: 1) when the robot has no target or obstacle in its surroundings (e.g. the target is not visible), the navigation is based on waypoint following, and 2) when a target or obstacle is visible, it uses PBVS to provide accurate navigation. Behavior trees provide robust, reactive, flexible, and modular frameworks for handling high-level tasks in complex UAV missions. They naturally have safety-driven condition checks that provide additional robustness. On the other hand, PBVS enables robust UAV guidance, adaptation to moving targets,

and, unlike IBVS, can easily solve situations when the target features fall out of the camera field of view. As a result, the proposed navigation scheme enables performing complex maneuvers in tight environments with significant levels of robustness and safety.

The proposed guidance scheme has been extensively validated in realistic simulations using SITL. The missions consisted in crossing through a large number (up to 20) of tight windows. A large number of missions with different number of windows (from 4 to 20) at different locations, different take-off and landing positions, and different sensor noise levels were satisfactory accomplished. Future works include the extension of the method to different types of targets –with different shapes and sizes– and multiple aerial robots and the evaluation on real aerial platforms. In addition, the use of a more complex behavior tree to consider safety checks typical of real-world aerial vehicle operations (e.g., C2 link status, geofencing, geocaging, etc.) is also considered for future research.

ACKNOWLEDGMENT

This work was funded by the SARA project (*Sistema aéreo no tripulado seguro para la inspección de líneas eléctricas fuera de la línea de vista*, TED2021-131716B-C22) of the Ministerio de Ciencia e Innovación del Gobierno de España. Partial funding was obtained from the Plan Estatal de Investigación Científica y Técnica y de Innovación of the Ministerio de Universidades del Gobierno de España (FPU19/04692).

REFERENCES

[1] R. Tapia, J. R. Martínez-de Dios, and A. Ollero, "Efficient mosaicking for linear infrastructure inspection using aerial robots," in *2019 Jornadas de Automática*, 2019, pp. 802–809.

[2] A. Caballero, R. Tapia, and A. Ollero, "Combining route planning and visual servoing for offshore wind-turbine inspection using UAVs," *Iberian Robotics Conference*, 2024.

[3] J. Cacace, S. M. Orozco-Soto, A. Suarez, A. Caballero, M. Orsag, S. Bogdan, G. Vasiljevic, E. Ebeid, J. A. Acosta Rodriguez, and A. Ollero, "Safe local aerial manipulation for the installation of devices on power lines: Aerial-core first year results and designs," *Applied Sciences*, vol. 11, no. 13, 2021.

[4] J. R. Martínez-de Dios, A. Gómez Eguíluz, J. P. Rodríguez-Gómez, R. Tapia, and A. Ollero, "Towards UAS surveillance using event cameras," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2020, pp. 71–76.

[5] F. J. Gañán, J. A. Sanchaz-Diaz, R. Tapia, J. R. Martínez-de Dios, and A. Ollero, "Efficient event-based intrusion monitoring using probabilistic distributions," in *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2022, pp. 211–216.

[6] F. J. Gañán, A. Suarez, R. Tapia, J. R. Martínez-de Dios, and A. Ollero, "Aerial manipulation system for safe human-robot handover in power line maintenance," *2022 Robotics: Science and Systems. Workshop in Close Proximity Human-Robot Collaboration*, 2022.

[7] A. E. Jimenez-Cano, P. J. Sanchez-Cuevas, P. Grau, A. Ollero, and G. Heredia, "Contact-based bridge inspection multirotors: Design, modeling, and control considering the ceiling effect," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3561–3568, 2019.

[8] A. Suarez, R. Salmoral, A. Garofano-Soldado, G. Heredia, and A. Ollero, "Aerial device delivery for power line inspection and maintenance," in *2022 International Conference on Unmanned Aircraft Systems*, 2022, pp. 30–38.

[9] A. Caballero, F. J. Roman-Escorza, I. Maza, and A. Ollero, "A multi-UAV route planning method for fast inspection of electric power transmission lines," in *2024 International Conference on Unmanned Aircraft Systems*, 2024, pp. 835–842.

[10] A. Suarez, A. Caballero, A. Garofano, P. J. Sanchez-Cuevas, G. Heredia, and A. Ollero, "Aerial manipulator with rolling base for inspection of pipe arrays," *IEEE Access*, vol. 8, no. 1, pp. 162 516–162 532, 2020.

[11] G. Wang, J. Qin, Q. Liu, Q. Ma, and C. Zhang, "Image-based visual servoing of quadrotors to arbitrary flight targets," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2022–2029, 2023.

[12] C. Qin, Q. Yu, H. S. H. Go, and H. H. T. Liu, "Perception-aware image-based visual servoing of aggressive quadrotor UAVs," *IEEE/ASME Transactions on Mechatronics*, vol. 28, no. 4, pp. 2020–2028, 2023.

[13] O. Araar and N. Aouf, "Visual servoing of a quadrotor UAV for the tracking of linear structured infrastructures," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 3310–3315.

[14] O. A. Hay, M. Chehadeh, A. Ayyad, M. Wahbah, M. A. Humais, I. Boiko, L. Seneviratne, and Y. Zweiri, "Noise-tolerant identification and tuning approach using deep neural networks for visual servoing applications," *IEEE Transactions on Robotics*, vol. 39, no. 3, pp. 2276–2288, 2023.

[15] L. Yang, Z. Liu, G. Wang, and X. Wang, "Image-based visual servo control for ground target tracking using a fixed-wing UAV with pan-tilt camera," in *2020 International Conference on Unmanned Aircraft Systems*, 2020, pp. 354–361.

[16] L. Mejias, P. Campoy, S. Saripalli, and G. Sukhatme, "A visual servoing approach for tracking features in urban areas using an autonomous helicopter," in *2006 IEEE International Conference on Robotics and Automation*, 2006, pp. 2503–2508.

[17] A. Gómez Eguíluz, J. P. Rodríguez-Gómez, R. Tapia, F. J. Maldonado, J. A. Acosta, J. R. Martínez-de Dios, and A. Ollero, "Why fly blind? Event-based visual guidance for ornithopter robot flight," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 1958–1965.

[18] J. P. Rodríguez-Gómez, R. Tapia, J. Paneque, P. Grau, A. Gómez Eguíluz, J. R. Martínez-de Dios, and A. Ollero, "The GRIFFIN perception dataset: Bridging the gap between flapping-wing flight and robotic perception," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1066–1073, 2021.

[19] R. Tapia, J. Rodríguez-Gómez, J. Sanchez-Diaz, F. Gañán, I. Rodríguez, J. Luna-Santamaria, J. R. Martínez-de Dios, and A. Ollero, "A comparison between frame-based and event-based cameras for flapping-wing robot perception," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.

[20] R. Tapia, J. R. Martínez-de Dios, and A. Ollero, "eFFT: An event-based method for the efficient computation of exact Fourier transforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[21] M. Colledanchise and P. Ogren, *Behavior Trees in Robotics and AI: An Introduction*. CRC Press, 2018.

[22] M. Gil-Castilla, R. Tapia, I. Maza, and A. Ollero, "Modular framework for autonomous waypoint following and landing based on behavior trees," *Iberian Robotics Conference*, 2024.

[23] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 2718–2725.

[24] C. Paxton, N. Ratliff, C. Eppner, and D. Fox, "Representing robot task plans as robust logical-dynamical systems," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2019, pp. 5588–5595.

[25] J. Kuckling, A. Ligot, D. Bozhinoski, and M. Birattari, "Behavior trees as a control architecture in the automatic modular design of robot swarms," in *2018 International Conference on Swarm Intelligence*, 2018, pp. 30–43.

[26] Q. Yang, Z. Luo, W. Song, and R. Parasuraman, "Self-reactive planning of multi-robots with dynamic task assignments," in *2019 International Symposium on Multi-Robot and Multi-Agent Systems*, 2019, pp. 89–91.

[27] P. Ogren, "Increasing modularity of UAV control systems using computer game behavior trees," in *AIAA Guidance, Navigation, and Control Conference*, 2012, pp. 1–8.

[28] B. M. Rocamora, P. V. G. Simplicio, and G. A. S. Pereira, "A behavior tree approach for battery-aware inspection of large structures using drones," in *2024 International Conference on Unmanned Aircraft Systems*, 2024, pp. 234–240.

[29] G. Y. Li, R. T. Soong, J. S. Liu, and Y. T. Huang, "UAV system integration of real-time sensing and flight task control for autonomous building inspection task," in *2019 International Conference on Technologies and Applications of Artificial Intelligence*, 2019, pp. 1–6.