Modular Framework for Autonomous Waypoint Following and Landing based on Behavior Trees

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

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

Ivan Maza
GRVC Robotics Lab
Universidad de Sevilla
Seville, Spain
imaza@us.es

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

***Abstract*—Autonomous navigation and landing are critical capabilities for modern unmanned aerial vehicles (UAVs), particularly in complex and dynamic environments. In this paper, we propose a modular framework that employs behavior trees to achieve reliable waypoint following and landing. Behavior trees offer a flexible and easily extensible method of managing autonomous behaviors, enabling the integration of various algorithms and sensors. Our framework is designed to enhance the robustness and flexibility of UAV navigation and landing procedures. Experimental results from Software In The Loop (SITL) simulation tests demonstrate the system's robustness and adaptability, showcasing its potential for a wide range of UAV applications. This work contributes to the advancement of autonomous UAV technology by providing a scalable and efficient solution for mission-critical operations.**

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

I. Introduction

The current environment demonstrates the broad applicability of Unmanned Aerial Vehicles (UAVs) in various civil missions, including cinema filming, photography, firefighting, package delivery, inspection [1], [2], and surveillance (e.g., [3], [4]). UAVs increasingly prove their reliability and value, especially in monitoring and diagnostic services for inspection operations [5]. Although skills are generally programmed at a lower level of abstraction (e.g, controllers for sensors and actuators), their coordination into higher-level representations to form missions is gaining increasing importance.

The traditional way of coordinating this autonomous decision-making has been finite-state machines. However, representing complex and dynamic environments quickly makes state machines difficult to manage. As references [6], [7] show, Behavior Trees (BTs) are gaining attention among roboticists as an effective language for high-level coordination, particularly for autonomous UAV missions, including landing. Originally developed for computer games to manage the behavior of autonomous non-player characters in complex and unpredictable environments, BTs offer significant benefits for robotics. They are favored for their modularity and ease of debugging, allowing for straightforward expansion and adjustment of missions [8]. The hierarchical structure of BTs organizes state transition logic in a tree format, with states as leaves, which enhances modularity and simplifies both synthesis and analysis. The nodes in a behavior tree are categorized into two types: 1) *control flow* nodes, which determine the execution path (*Sequence*, *Fallback*, *Parallel*, or *Decorator*); and 2) *execution* nodes, which carry out actions or check conditions. *Sequence* nodes run child nodes sequentially until one fails, *Fallback* nodes run child nodes sequentially until one succeeds, *Parallel* nodes allow multiple nodes to execute simultaneously, and *Decorator* nodes alter a child's behavior, such as by repeating it or adding conditions. *Execution* nodes provide feedback on the behavior tree's operation by returning SUCCESS, RUNNING, or FAILURE after execution. This structure enables a clear representation of layers of behavior, making it easier for both humans and algorithms to manage. These characteristics make BTs ideal for autonomous UAV applications, where managing complex and dynamic environments is crucial for successful mission execution. In this paper, we propose a proof-of-concept BT-based framework for aerial robot operations which provides a firm foundation for complex systems, missions, and scenarios.

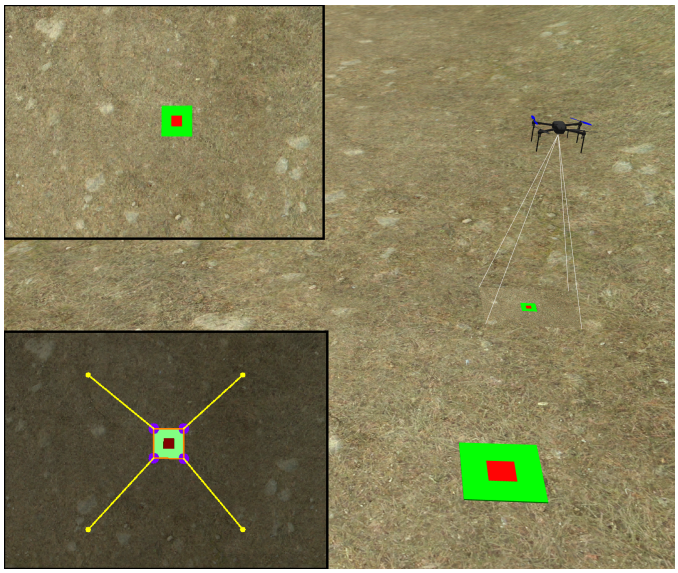

Fig. 1. Simulation setup for evaluation during autonomous landing. Top left: image obtained by the camera of the UAV. Bottom-left: result of the visual servoing used for landing (yellow lines represent the feature references).

The main contributions of this work are: 1) an extendable and modular framework to integrate behavior tress and UAVs to manage complex autonomous missions which only relies on open source tools, 2) the particularization of this framework to perform a complete waypoint following mission, 3) a new approach for autonomous landing governed by a behavior tree, and 4) the evaluation of our method using realistic simulations (see Figure 1). The rest of the paper is structured as follows. Section II presents the main work related to the topics addressed in the paper. Section III describes the proposed framework based on behavior trees. The simulation implemented for the experimental validation is presented in Section IV. Section V closes the paper with the conclusions and future work.

II. Related Work

To contextualize the development of our modular framework for autonomous waypoint following and landing based on BTs, it is essential to review existing research and advancements in the field. At first, in [9], the authors formalized and analyzed the reasons behind the success of the BTs using standard tools of robot control theory. The paper [10] introduces a comprehensive framework for BTs, serving as a tool for the representation and execution of the plan. This study [6] examines key language concepts in BTs and their application in real-world robotics, comparing them with state and activity diagrams, analyzing usage in open source repositories, and contributing a dataset of behavior models to inspire further development and use within the community. Then, reference [11] presents a comprehensive survey of the topic of BTs in Artificial Intelligence and Robotics.

Recently, the Robotic Operating System (ROS), as the main open source platform for robotic software, has adopted BTs as the main customization mechanism for its navigation stack Navigation2[1]. Navigation2 utilizes a BT to coordinate navigation tasks and incorporates innovative methods tailored for dynamic environments, making it compatible with a wider range of modern sensors. The paper [12] performs experiments in a campus setting, using Navigation2 to operate safely among students during a marathon. In [13], PlanSys2 is presented and it aims to be the reference task planning framework in ROS2 based on its optimized execution on BTs.

In the field of swarm robotics, an automatic design method that combines preexisting modules into BTs, demonstrating its effectiveness in missions such as foraging and aggregation when compared to other methods like Chocolate and EvoStick is presented in [14]. Similarly, [15] proposed a framework utilizing BTs for self-reactive planning in multi-robot systems with dynamic task assignments. This framework incorporates distributed algorithms and a novel priority mechanism for communication and negotiation among robots, showing promising results in simulation experiments.

Closer to this topic, UAV applications, the authors in [16] tackles the process of replacing the battery and manual restarting an inspection mission is automated using behavior trees, providing an effective means of autonomous mission control and supervision, eliminating the need for human intervention. In work [17], the flight control system uses a behavior tree as a decision control mechanism.

Finally, numerous studies have demonstrated the effectiveness of visual servoing techniques for onboard aerial robots across various applications. The work [18] presents an onboard vision-based control scheme to track an intruder flying arbitrarily. A visual method for autonomous landing based on vector fields is presented in [19]. The authors in [20] propose a method to autonomously land a UAV on a moving vehicle. The work [21] proposes a visual control approach for air-to-air autonomous landing.

III. Framework Description

Our framework aims to provide an extendable and easy-to-use solution for integrating UAVs with ROS Noetic and behavior trees. Designed to be comprehensive yet simple, this framework facilitates the deployment of UAVs, specifically those using MAVROS and ArduPilot, for various applications. Using open-source tools ensures flexibility and adaptability, making it an ideal first approach for developers and researchers working on advanced UAV systems. The modular design of the framework allows for the seamless integration of additional features and improvements, paving the way for sophisticated, reliable, and scalable multi-UAV operations.

A. Behavior Tree

The behavior tree begins by setting several blackboard variables to define the initial pose and four target goals (Goal_a to Goal_d). These targets outline a square path that the UAV will follow. In this case, the purpose is to simplify the whole scene and simulate a waypoint following route with the following actions:

- **TakeOff**: Before starting the waypoint mission the UAV takes off if it is not in the air.
- **GoToGoal**: This action will guide the UAV to a specified waypoint with the simple position control the mavros package offers.
- Landing: This last stage of the mission will be divided in three: **Approach**, **Fine** and **Land**.

The conditions considered to enable the actions are:

- ***CheckOnAir***: Check the landed state of the UAV, if it is in the air it will return SUCCESS so the waypoint following can be done.
- ***PlatformVisible***: If the landing platform is a given percentage of the camera image, the landing procedure can start.
- ***CheckError***: The condition will look for an specific threshold acceptable to change the landing phases.

B. Waypoint Following

As the main focus of the behavior trees is the management of high-level tasks, high-level functions are used thanks to the iq_gnc ROS package. We use the set_destination function to

[1]https://github.com/ros-navigation/navigation2

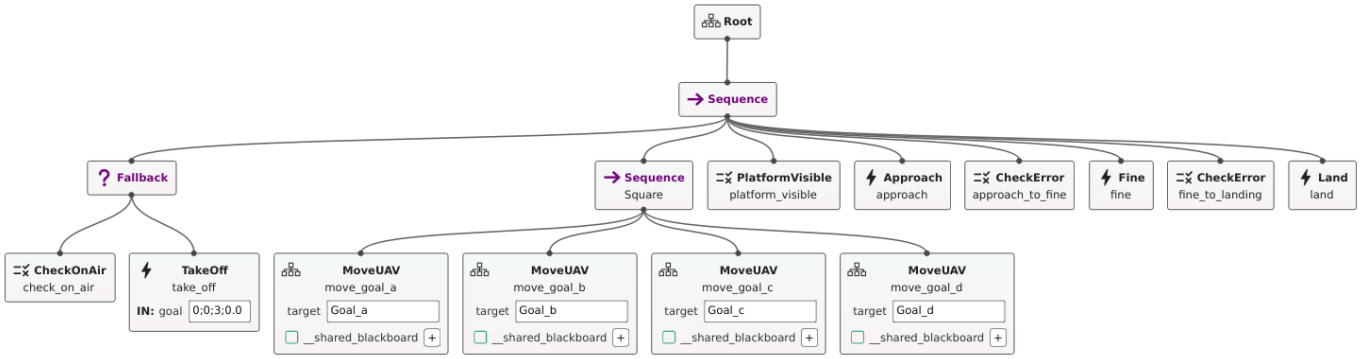

Fig. 2. Behavior tree used for the evaluation of our framework. Blackboard variables are hidden for a clearer visualization.

send waypoints that direct the UAV to fly to a specified location. These waypoints must be defined in the local reference frame, typically set from the UAV's launch position. The UAV is commanded in terms of position and heading.

C. Autonomous Landing

For autonomous landing we use RGB images and measurements from an altimeter. Camera calibration is known and images are undistorted prior to processing. The landing platform is an $N_A \times N_A$ square of color $\mathcal{C}_A$ with another concentric $N_F \times N_F$ square of color $\mathcal{C}_F$ ($N_F < N_A$) as the one shown in Fig. 4. We sequentially use the external square to center the UAV on the platform (**Approach** stage) and the internal for a finer positioning of the robot (**Fine** stage). This section describes the visual-based method used for positioning the UAV, which is the same for the **Approach** and **Fine** stages. Once the vehicle is at the final position, **Land** (i.e., vertical descent without visual guidance). The reference frames used in this section are presented in Figure 3.

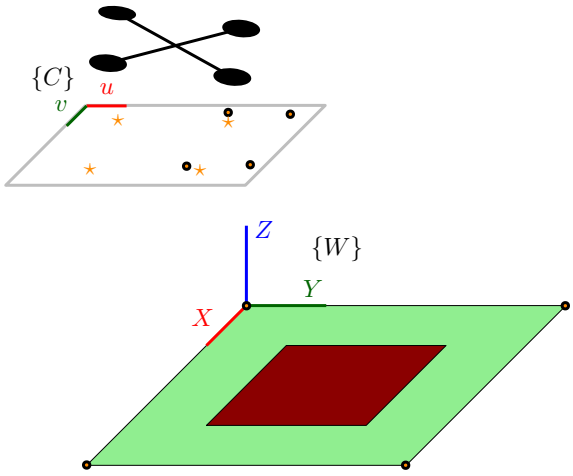

Fig. 3. Frame references: image plane $\{C\}$, world $\{W\}$.

First, the landing platform is detected by using color thresholding in the RGB images. Both $\mathcal{C}_A$ and $\mathcal{C}_F$ are known. The HSV ranges for thresholding are selected so that the method is robust against small variations in the color of the platform (e.g. changes in illumination). Then, the four corners of the square are detected by approximating the contour of the selected region by a 4-side polygon. Once the 4 corners $(u_1, v_1)^T$, $(u_2, v_2)^T$, $(u_3, v_3)^T$, and $(u_4, v_4)^T$ are detected, we use image-based visual servoing [22] to control the UAV towards the landing position. Given a point $(X, Y, Z) \in \mathbb{R}^3$ which is seen by the camera with linear velocity $(v_x, v_y, v_z) \in \mathbb{R}^3$ and angular velocity $(\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ expressed w.r.t. the camera frame, we have

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = - \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}. \tag{1}$$

The point is projected in the image as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{Z} \begin{bmatrix} f_x X + c_x Z \\ f_y Y + c_y Z \end{bmatrix}. \tag{2}$$

Deriving Equation 2:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \frac{1}{Z^2} \begin{bmatrix} f_x(\dot{X}Z - X\dot{Z}) \\ f_y(\dot{Y}Z - Y\dot{Z}) \end{bmatrix}. \tag{3}$$

Assuming $\bar{u} = u - c_x$ and $\bar{v} = v - c_y$:

$$\begin{aligned} \begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = & \frac{1}{Z} \begin{bmatrix} (\bar{u} v_z - f_x v_x) \\ (\bar{v} v_z - f_y v_y) \end{bmatrix} + \\ & + \begin{bmatrix} \frac{1}{f_y} \bar{u}\bar{v}\omega_x - \left(f_x + \frac{1}{f_x}\bar{u}^2\right)\omega_y + \frac{f_x}{f_y}\bar{v}\omega_z \\ -\frac{1}{f_x}\bar{u}\bar{v}\omega_y + \left(f_y + \frac{1}{f_y}\bar{v}^2\right)\omega_x - \frac{f_y}{f_x}\bar{u}\omega_z \end{bmatrix}. \end{aligned} \tag{4}$$

This leads to:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \mathbf{J}(u, v, Z) \begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}, \tag{5}$$

where

$$\mathbf{J}(u, v, Z) = \begin{bmatrix} -\frac{f_x}{Z} & 0 & \frac{\bar{u}}{Z} & \frac{\bar{u}\bar{v}}{f_y} & -\frac{f_x^2 + \bar{u}^2}{f_x} & \bar{v} \\ 0 & -\frac{f_y}{Z} & \frac{\bar{v}}{Z} & \frac{f_y^2 + \bar{v}^2}{f_y} & -\frac{\bar{u}\bar{v}}{f_x} & -\bar{u} \end{bmatrix}. \tag{6}$$

For our four detected points and assuming $Z_1 = Z_2 = Z_3 = Z_4 = h$, being h the measured altitude:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \mathbf{J}(u_1, v_1, h) \\ \mathbf{J}(u_2, v_2, h) \\ \mathbf{J}(u_3, v_3, h) \\ \mathbf{J}(u_4, v_4, h) \end{bmatrix}^{\dagger} \begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \dot{u}_2 \\ \dot{v}_2 \\ \dot{u}_3 \\ \dot{v}_3 \\ \dot{u}_4 \\ \dot{v}_4 \end{bmatrix}. \tag{7}$$

Adding a linear velocity controller in the form $\dot{\xi} = K(\xi^{\star} - \xi)$ and using a second-order approximation for faster convergence, we have

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = K \left(\begin{bmatrix} \mathbf{J}(u_1, v_1, h) \\ \mathbf{J}(u_2, v_2, h) \\ \mathbf{J}(u_3, v_3, h) \\ \mathbf{J}(u_4, v_4, h) \end{bmatrix}^{\dagger} + \begin{bmatrix} \mathbf{J}(u_1^{\star}, v_1^{\star}, h^{\star}) \\ \mathbf{J}(u_2^{\star}, v_2^{\star}, h^{\star}) \\ \mathbf{J}(u_3^{\star}, v_3^{\star}, h^{\star}) \\ \mathbf{J}(u_4^{\star}, v_4^{\star}, h^{\star}) \end{bmatrix}^{\dagger} \right) \begin{bmatrix} u_1^{\star} - u_1 \\ \dot{v}_1^{\star} - v_1 \\ u_2^{\star} - u_2 \\ \dot{v}_2^{\star} - v_2 \\ u_3^{\star} - u_3 \\ \dot{v}_3^{\star} - v_3 \\ u_4^{\star} - u_4 \\ \dot{v}_4^{\star} - v_4 \end{bmatrix}. \tag{8}$$

Finally, to avoid abrupt movements that cause the platform to be out of field of view (FoV), v_x, v_y and v_z are saturated. Since velocity control references are expressed w.r.t. the robot body frame R (and not w.r.t. the camera frame C), a transformation ${}^{R}\mathbf{T}_{C}$ must be applied to the output of Equation 8.

IV. Simulation Results

The complex and dynamic nature of UAV operating environments poses significant challenges in system design and implementation. Simulations provide a controlled and reproducible setting, allowing safe testing and evaluation of various algorithms, control strategies, and UAV behaviors before their deployment in real-world scenarios. A Software In The Loop (SITL) approach was used to simulate MAVROS compatible UAVs thanks to the Ardupilot firmware and being compatible with the Gazebo simulator. In this approach, the UAV control software, including the autopilot firmware and navigation algorithms, is executed in a simulated environment on a computer with a graphics engine, without the need for a physical UAV. Ardupilot establishes communication with the simulator to obtain sensor data from the simulated environment and transmits actuator values through the UDP protocol. The simulation environment for our experiments is shown in Figure 4. We utilized the winding valley heightmap Gazebo world, which provides a realistic and challenging terrain.

We use the Intelligent Quads repositories for the integration of the behavior tree and the simulation of the UAV. The repository iq_sim[2], which hosts Gazebo worlds for various UAV scenarios and configurations, is specifically designed to work with the Ardupilot control system. Using the Ardupilot Gazebo plugin, it enables seamless interfacing and control of model UAVs within Gazebo. In addition, the package iq_gnc[3] gives us a collection of high level functions to work with behavior trees easily.

We used the BehaviorTree.CPP library, which has been effectively adapted for use with ROS through the behaviortree_cpp_v3[4] package. For the vision-based algorithm used for landing, OpenCV was used. This combination allows us to design and execute the behavior trees needed for autonomous UAV navigation and control within a simulated environment. The implementation of our framework[5][6] can be found under the GNU license. A video demonstration[7] is also provided.

Our approach provided several benefits:

- **Modularity**: Each behavior in the tree is encapsulated in a node, enabling easy modifications and extensions.
- **Flexibility**: The use of blackboard variables and the hierarchical structure of behavior trees facilitated dynamic adjustment of goals and behaviors based on real-time sensor inputs.
- **Robustness**: The combination of Fallback and Sequence nodes ensured robust handling of various scenarios, such as checking if the UAV is airborne before initiating a takeoff.

The behavior tree workflow begins with the initialization stage, where several blackboard variables are set to define the initial pose and four target goals (Goal_a, Goal_b, Goal_c, and Goal_d). These targets create a square path for the UAV to follow. Next, in the takeoff sequence, a fallback node first checks if the UAV is already airborne using the ***CheckOnAir*** node. If the UAV is not airborne, it proceeds with the **TakeOff** node to reach the initial altitude.

Once in the air, the UAV enters the navigation sequence, where it follows the Square sequence to move to each predefined goal (Goal_a to Goal_d) using the MoveUAV subtree. Each move action is executed by the **GoToGoal** node, which navigates the UAV to the specified targets.

After completion of the square path, the UAV initiates the landing procedure. This involves a series of checks and maneuvers to locate and approach the landing platform using the ***PlatformVisible*** and **Approach** nodes. To ensure precision, additional error checks are performed with the ***CheckError*** nodes during the fine adjustment (**Fine**) and final landing (**Land**) phases. Although the sequence is valid for a simple inspection and landing mission, our aim is to provide a proof-of-concept BT-based framework for aerial robot operations. More complex missions and scenarios (including multi-UAV) can also be validated by extending the BT in a modular fashion, where finite state machines become harder to scale.

The UAV trajectory in the entire mission is presented in Figure 5, where it is shown the waypoint following stage and finally the approximation and landing phases.

Figure 6 shows the velocity variation during the landing phase. The difference in the Z axis (i.e., vertical axis) where

[2]https://github.com/Intelligent-Quads/iq_sim
[3]https://github.com/Intelligent-Quads/iq_gnc
[4]https://index.ros.org/r/behaviortree_cpp
[5]https://github.com/miggilcas/grvc_bt_ros
[6]https://github.com/raultapia/bt_ibvs
[7]https://youtu.be/UY87lFj3_M4

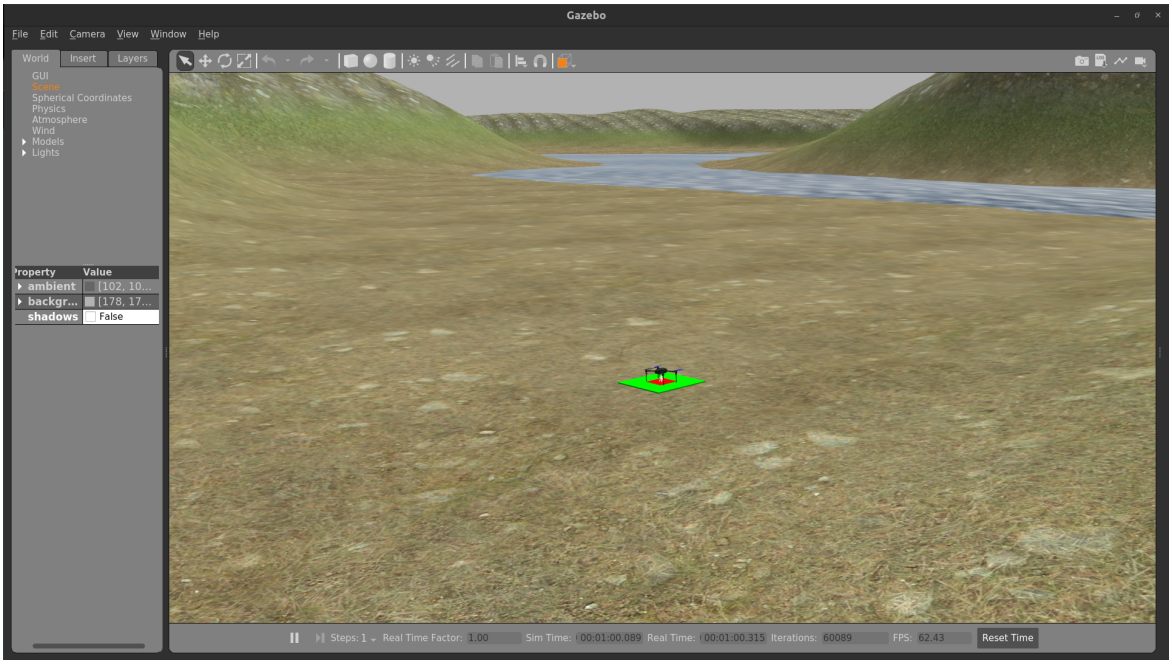

Fig. 4. Simulation framework used composed of winding valley heightmap Gazebo world, the iris quadrotor model with a zenital view camera and the landing platform.

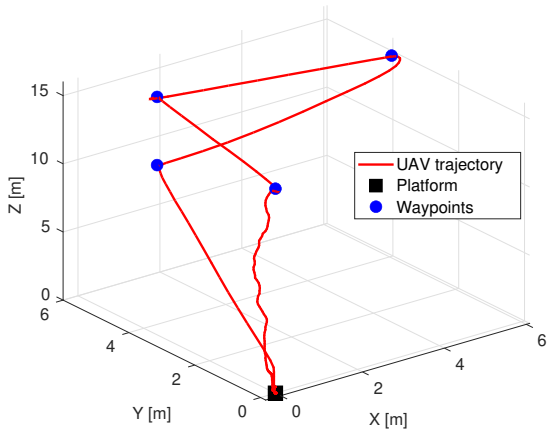

Fig. 5. Robot trajectory during take-off, waypoint following, and landing.

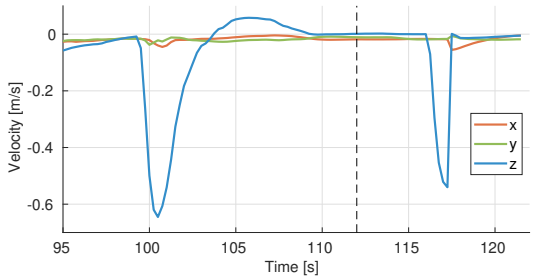

Fig. 6. UAV velocity during autonomous landing. Dashed black line indicates the time of the switch between **Approach** and **Fine**.

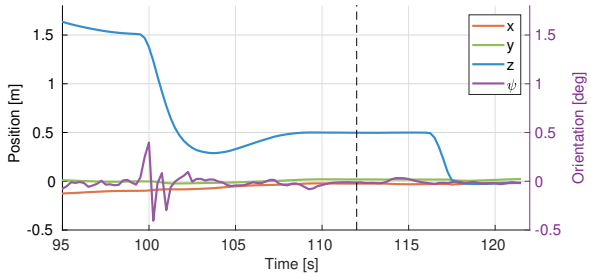

Fig. 7. UAV position and yaw during autonomous landing. Dashed black line indicates the time of the switch between **Approach** and **Fine**.

the change from **Approach** to **Fine** is clearly visible. In addition, Figure 7 shows the position and yaw angle during UAV landing. It should be noted that the position of the UAV tends to the location of the platform, i.e., $(0, 0, 0)$.

The trajectory of the features in the image plane during autonomous landing is shown in Figure 8. It is easy to see how the features converge to the desired final position as the UAV lands. Only the **Approach** phase is shown for brevity. The behavior of **Fine** provides similar results.

V. Conclusions and Future Work

The work proposed in this paper, validated in a simulated environment, demonstrated the UAV's capability to perform reliable autonomous waypoint following and precision landing, showcasing the effectiveness of behavior trees for autonomous UAV operations. In addition, a new approach to autonomous landing has been developed using RGB images and altimeter measurements to precisely guide the UAV to a landing plat-

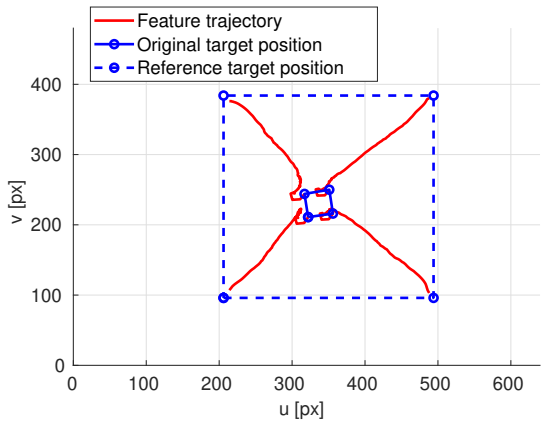

Fig. 8. Trajectory of the features during the **Approach** stage.

form through a sequence of visual-based positioning stages, leveraging color thresholding and image-based visual servoing. This method ensures robust and accurate landings even under varying illumination conditions.

Future work will focus on several key areas to enhance the framework. Firstly, our goal is to support a broader range of UAV types and autopilots, including those not using the MAVLink protocol, such as DJI models, by adapting communication and control interfaces for compatibility. Secondly, we plan to increase the complexity of the behavior trees to manage more sophisticated missions. This includes enabling UAVs to dynamically respond to emergency events, integrating advanced battery management strategies to prioritize tasks based on power levels, and enabling multi-UAV missions where drones can communicate and collaborate on large-scale operations. A more sophisticated study must be performed to analyze the response of the system in the event of emergency using BT conditions. In addition, we intend to transition from simulations to real-world experiments to validate the effectiveness and robustness of the framework in practical settings. Finally, we will port the framework to ROS2 to take advantage of its improved features for real-time performance, security, and communication, thus improving the scalability and suitability of the system for complex and distributed robotic applications.

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. 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.

[4] 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.

[5] M. Gil-Castilla, A. Caballero, I. Maza, and A. Ollero, "Integration of customized commercial UAVs with open-source tools in a heterogeneous multi-UAV system for power-lines inspection," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2024, pp. 1362–1369.

[6] R. Ghzouli, T. Berger, E. B. Johnsen, S. Dragule, and A. Wąsowski, "Behavior trees in action: a study of robotics applications," in *2020 International Conference on Software Language Engineering*, 2020.

[7] M. Colledanchise and L. Natale, "On the implementation of behavior trees in robotics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, p. 5929–5936, 2021.

[8] R. Tapia, M. Gil-Castilla, J. R. Martínez-de Dios, and A. Ollero, "Autonomous guidance of an aerial robot using 6-DoF visual control and behavior trees," *Iberian Robotics Conference*, 2024.

[9] M. Colledanchise and P. Ögren, "How behavior trees modularize robustness and safety in hybrid systems," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 1482–1488.

[10] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren, "Towards a unified behavior trees framework for robot control," in *2014 IEEE International Conference on Robotics and Automation*, 2014, pp. 5420–5427.

[11] M. Iovino, E. Scukins, J. Styrud, P. Ögren, and C. Smith, "A survey of behavior trees in robotics and AI," *Robotics and Autonomous Systems*, vol. 154, p. 104096, 2022.

[12] S. Macenski, F. Martin, R. White, and J. G. Clavero, "The Marathon 2: A navigation system," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[13] F. Martín, J. G. Clavero, V. Matellán, and F. J. Rodríguez, "PlanSys2: A planning system framework for ROS2," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 9742–9749.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] V. M. Gonçalves, R. McLaughlin, and G. A. S. Pereira, "Precise landing of autonomous aerial vehicles using vector fields," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4337–4344, 2020.

[20] A. Keipour, G. A. S. Pereira, R. Bonatti, R. Garg, P. Rastogi, G. Dubey, and S. Scherer, "Visual servoing approach to autonomous UAV landing on a moving vehicle," *Sensors*, vol. 22, no. 17, 2022.

[21] G. Roggi, G. Gozzini, D. Invernizzi, and M. Lovera, "Vision-based air-to-air autonomous landing of underactuated VTOL UAVs," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 3, pp. 2338–2349, 2024.

[22] 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.