







Article

An Efficient Framework for Autonomous UAV Missions in Partially-Unknown GNSS-Denied Environments

Michael Mugnai ^{*,†} , Massimo Teppati Losé [†] , Edwin Paúl Herrera-Alarcón , Gabriele Baris ,
Massimo Satler  and Carlo Alberto Avizzano 

Institute of Mechanical Intelligence, Sant'Anna School of Advanced Studies, 56127 Pisa, Italy; massimo.teppatilose@santannapisa.it (M.T.L.); edwinpaul.herreraalarcon@santannapisa.it (E.P.H.-A.); gabriele.baris@santannapisa.it (G.B.); massimo.satler@santannapisa.it (M.S.); carloalberto.avizzano@santannapisa.it (C.A.A.)

* Correspondence: michael.mugnai@santannapisa.it

† These authors contributed equally to this work.

Abstract: Nowadays, multirotors are versatile systems that can be employed in several scenarios, where their increasing autonomy allows them to achieve complex missions without human intervention. This paper presents a framework for autonomous missions with low-cost Unmanned Aerial Vehicles (UAVs) in Global Navigation Satellite System-denied (GNSS-denied) environments. This paper presents hardware choices and software modules for localization, perception, global planning, local re-planning for obstacle avoidance, and a state machine to dictate the overall mission sequence. The entire software stack has been designed exploiting the Robot Operating System (ROS) middleware and has been extensively validated in both simulation and real environment tests. The proposed solution can run both in simulation and in real-world scenarios without modification thanks to a small sim-to-real gap with PX4 software-in-the-loop functionality. The overall system has competed successfully in the Leonardo Drone Contest, an annual competition between Italian Universities with a focus on low-level, resilient, and fully autonomous tasks for vision-based UAVs, proving the robustness of the entire system design.

Keywords: UAV; MAV; mission planning; collision avoidance; navigation in partially-known environments; visual-based navigation; GNSS-denied



Citation: Mugnai, M.; Teppati Losé, M.; Herrera-Alarcón, E.P.; Baris, G.; Satler, M.; Avizzano, C.A. An Efficient Framework for Autonomous UAV Missions in Partially-Unknown GNSS-Denied Environments. *Drones* **2023**, *7*, 471. <https://doi.org/10.3390/drones7070471>

Academic Editors: Samir Khan, Vaios Lappas and Nadjim Horri

Received: 23 May 2023
Revised: 9 July 2023
Accepted: 15 July 2023
Published: 18 July 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Recent technological advances have raised interest in autonomous Micro-Aerial Vehicles (MAVs) for different applications from agriculture to industry. MAVs assist human operators in repetitive or dangerous tasks, such as powerline maintenance [1] and Search and Rescue (SAR) activities. The fields where autonomous drones are reducing the complexity and risk of different human chores are increasing daily. Current challenges in MAV research involve using only local sensing and pushing the efficiency of the algorithms further in order to improve the agent's self-sufficiency. Anyway, the design of a fully autonomous framework for a small platform with limited payload and sensing capabilities poses significant challenges. The design of such systems is a process that requires the continuous improvement of both the hardware and the software components to find the optimal trade-off between performance and flight duration.

For example, from 2018 to 2021 international agencies such as DARPA funded the Subterranean Challenge (SubT) Search and Rescue competition. The challenge aimed to boost the research and the development of technologies that can support operations in complex and diverse underground settings which pose significant risks and challenges for military and civilian first responders. In this challenge, the teams had to address perception, networking, and navigation problems that arise when using autonomous robots

in unpredictable subsurface environment conditions in order to obtain a map of the scenario. The teams that participated in the challenge were Team CoStar [2], Team CSIRO Data61 [3], Team CTU-CRAS-NORLAB [4], Team Cerberus [5], Team Explorer (www.cmu.edu/news/stories/archives/2021/september/darpa-subt-finals.html (accessed on 17 July 2023)), Team Marble (www.colorado.edu/today/2021/09/24/engineers-take-home-5000-00-international-underground-robotics-competition (accessed on 17 July 2023)), Team Coordinated Robotics, and Team Robotika. These teams implemented different heterogeneous multi-robot systems for the SAR exploration task that can be split into two categories. On one side, there was a significant variety of ground robots, including legged robots, crawling hexapods, and both wheeled and tracked rovers. On the other side, the flying robots consisted of custom-built multirotors, which are optimal for autonomous operations relying only on onboard sensing due to their maneuverability, vertical take-off and landing capability, small factor size, and low cost.

The flying platforms of the DARPA Subterranean Challenge present different frameworks towards full aerial autonomy, for which environment perception sensors strongly rely on 3D LiDAR technology due to the lack of good visibility in certain areas. Nevertheless, most MAVs used cameras, as the subterranean exploration mission was oriented towards identifying and localizing specific objects within the environment, which was incrementally mapped as the robots navigate in it.

Depending on the scenario, lighting conditions, and tasks to be accomplished, autonomous platforms could generally operate based on camera systems only regardless of whether they used a 3D LiDAR. LiDAR technology is fast and accurate, and the amount of information obtained at each sample time is huge. Therefore, its usage in an autonomous system requires an onboard computer with the computational capacity to handle a large amount of information in real-time. In addition, its price with respect to other onboard sensors is significantly higher. Vision-based navigation relying only on onboard sensors such as cameras and inertial measurement units (IMU) is a plausible option. Cameras are promising sensors for small drones because they can gather rich information and span wide fields of view for comparable mass. An example of a vision-based MAV autonomous stack is Ingenuity (<https://mars.nasa.gov/technology/helicopter/> (accessed on 17 July 2023)) [6], a 1.8 kg multirotor designed to autonomously fly for up to 90 s in the atmosphere of Mars. Ingenuity is the first aircraft to accomplish autonomous exploration on a planet relying entirely on onboard sensors. Its sensors include inertial measurement units for accelerations and angular rates, a laser rangefinder to measure its distance to the ground, and a navigation camera to take pictures.

New challenges related to vision-based tasks in autonomous drones have appeared in recent years. For instance, the MBZIRC 2020 competition's first challenge was intended for an autonomous UAV to localize, track, and neutralize an intruder UAV in an outdoor open space. The International Conference of Unmanned Aircraft Systems (ICUAS) 2022 proposed a UAV competition in which the drone had to quickly explore an environment, detect a static target, and precisely deliver an object to the target.

1.1. Competition Details and Proposed System Overview

Leonardo S.p.A., an Italian aerospace and defense company, created the Drone Contest challenge (<https://www.leonardo.com/it/innovation-technology/open-innovation/drone-contest> (accessed on 17 July 2023)) in 2019, a yearly event among Italian universities to solve vision-based tasks in an indoor GNSS-denied environment using a UAV. In the 2022 competition, the environment was partially unknown: teams knew the position of many obstacles, while others were randomly placed before each round. Each round consisted of two phases: first, the drone had to explore the environment looking for a moving target, then it had to be tracked for at least 10 s. In the second phase, a sequence of actions was provided to the team, with each action having a score. The teams decided on the execution order of the actions. Actions could be of two types: precision landing, or taking a picture of a marker. The winner was the team with the highest score over the three rounds of

the competition. In detail, the autonomous platform had to navigate autonomously in an environment of dimensions 20 m × 10 m with a maximum height of 3 m while relying only on vision-based sensors. Within the environment, the drone had to search for a mobile ground robot and track it for a minimum amount of time. Then, a sequence of actions was provided to the UAV. These actions were divided between precision landing and taking pictures of fiducial markers positioned on the vertical faces of the obstacles.

This paper introduces a ROS framework capable of performing autonomous missions in a partially unknown GNSS-denied environment and presents a solution for the Drone Contest challenge with a vision-based autonomous MAV based on the open-source PX4 firmware. The proposed solution was tested in a real environment during competition. (a video of the competition can be found at <https://www.youtube.com/watch?v=HiNrlqCKWIQ&t=3133s> (accessed on 17 July 2023)). While within the Drone Contest challenge a partial map of the environment is available, this work considers the possibility of variations within the environment such as the presence of new unknown obstacles, extending the global planning strategy to work alongside a local re-planning module. In previous works, we proposed an object-oriented exploration and volumetric mapping of an unknown environment [7] and a 3D low-dimensional topological graph generated on previously explored environments, which is useful for global path planning [8]. In this work, the navigation graph is extended as the entry point of a trajectory planner able to guide UAVs in partially known environments through an obstacle avoidance module that overrides desired UAV velocities in order to evade unmapped obstacles. The overall system is handled by a mission planner that drives the UAV during the entire challenge using a finite-state machine.

1.2. Related Works

This section reviews works presented in the literature regarding fully autonomous frameworks, then works concerning their evolution towards vision-based platforms, and finally provides a comparison between the presented hardware and software modules with other state-of-the-art systems with similar characteristics.

Over the last decade, the research topic of aerial navigation has driven towards the challenging problem of employing solely onboard sensors. The earliest overall frameworks, such as Shen et al. [9], proposed a platform based on an IMU, monocular camera, and 3D LiDAR as unique sensors for the autonomous UAV. Tomic et al. [10] introduced a quadcopter with navigation based on a combination of Visual Odometry from a stereo camera and 3D LiDAR scanning. These works focused on providing greater robustness for both indoor and outdoor SAR scenarios. Both frameworks were validated through experiments that highlighted navigation performance through evaluations of the pose estimation and loop closure in different indoor and outdoor scenarios. On the other side, Fraundorfer et al. [11] presented a quadrotor with navigation that depended solely on visual information, using a front-facing stereo camera and a down-facing optical flow camera. Their work emphasized accurate perception of the environment by using two different sensors to analyze different planes. The navigation was validated by testing it with an onboard planning and mapping algorithms, while the loop closure was done offboard.

Nowadays, the most common architecture for autonomous custom-built drones uses two boards: a companion computer for high-level modules and a low-level attitude controller board. For the latter, Pixhawk (<https://pixhawk.org> (accessed on 17 July 2023)) hardware has become an open-source standard. Up to its original work on autonomous vision-based MAVs, Pixhawk [12] has become a flexible research module to standardize the low-level controller and the state estimation. An important advantage of Pixhawk boards is that they are supported by the most popular autopilots for autonomous vehicles: PX4 [13] and ArduPilot (<https://ardupilot.org/> (accessed on 17 July 2023)).

Another option for low-level control and estimation is the board used by Loianno et al. [14] with the Qualcomm® Snapdragon™ Flight™, which aims to achieve lightweight visual-inertial navigation. This board features a downward-facing VGA camera with 160° field

of view, a VGA stereo camera, and a 4 K camera, all packed into a board of dimensions 58 mm × 40 mm. Ge et al. [15] used this setup in a MAV swarm system for decentralized vision-based detection and tracking strategies. On the same topic, Thakur et al. [16] instead used the successor of the previously mentioned board. These works depict examples of lightweight vision-based platforms using smartphone processors with weights under 450 g. These platforms are lightweight enough to be both agile and resistant in case of collisions, and as such are a good prototype for testing new algorithms. However, they are currently not designed to fulfill long high-level missions due to their lack of high computational power and low level of flight autonomy.

The Size, weight, and power (SWaP) constraints perfectly describe the trade-offs for designing MAV platforms. For this reason, autonomous flying system developers often have to accurately choose the number of sensors to use, then estimate the computational cost needed for processing their information and the minimum accuracy they want to reach in terms of both the navigation error and the trajectory tracking performance. Liu et al. [17] highlighted the evolution of technologies for autonomous multirotors as well as the need for bigger sensors, more computational power, and better batteries for GNSS-denied environments. On one side, platforms under 450 g (wheelbase under 0.35 m from tip to tip with 3-inch propellers) continue to see reduced form factor and battery size while increasing their flight time and performance. On the other side, the evolution of larger prototypes (wheelbase over 0.75 m) has significantly improved in terms of endurance, sensor carrying technology, and computational power. However, their weight remains over 3.5 kg.

Moon et al. [18] described the challenges involved in Autonomous Drone Racing (ADR) and analyzed the technologies and results in the IROS competitions in 2016 and 2017. ADR is a vision-based competition in which the drone has to navigate in a cluttered indoor environment while detecting and crossing through gates and relying only on onboard resources. Several of the platforms used in this competition are based on commercial off-the-shelf drones [19,20] modified for the competition rules, while other teams build their own custom drones [21,22]. Other published platforms relate to specific competitions or programs, such as the drone from the AlphaPilot [23] ADR Competition (www.lockheedmartin.com/en-us/news/events/ai-innovation-challenge.html (accessed on 14 March 2023)), the team from Politecnico di Milano [24] in the Leonardo Drone Contest, and Mohta et al. [25,26] from the DARPA Fast Lightweight Autonomy program.

Several projects have been published with the aim of reproducibility and to encourage open-sourcing. For example, the Open Vision Computer (<https://osrf.github.io/ovc/> (accessed on 17 July 2023)) project (OVC) was tested on a drone [27]. Oleynikova et al. [28] and Kompis et al. [29] have published both the hardware and software setup of their fully autonomous projects. Finally, other projects have published their integrated frameworks to display their performance in accomplishing specific problems, such as SAR tasks [30], sensor placement [31], and using computational-constrained platforms [32].

Table 1 summarizes the overall specifications and hardware information of the previously mentioned frameworks. All of the mentioned platforms navigate autonomously while relying only on vision sensors. These platforms were presented to solve different problems, such as racing, high-speed navigation, exploration in unknown cluttered environments, dense mapping and reconstruction of real-world scenarios, and sensor allocation. Compared to the other solutions, our proposed platform shows a fair trade-off between the available computational power of the companion computer and its small size, making it particularly suitable for performing complex navigation and exploration tasks in cluttered environments while relying primarily on vision. The main disadvantages are the limited flight time and the limited angular range to detect obstacles due to the lack of LiDAR sensors.

For autonomous navigation and obstacle avoidance in GNSS-denied environments, the platform must continuously preserve information about its pose with different localization strategies. Different works in the literature have focused on the development of several

individual modules in simulation [33,34] without developing a comprehensive solution in real-world experiments. In this work, we present a complete framework that enables a drone to autonomously navigate in partially known environments even in the absence of a reliable position reference such as a GNSS.

Table 1. Summarized characteristics of autonomous vision-based middle-size platforms described in the literature.

| Reference | Wheelbase/ Weight | Flight Time | Companion Computer | FCU/ Autopilot | Sensors |
|------------------------------|----------------------|-------------|---|---------------------------------------|---|
| Rojas-Perez et al. [19] | 47 cm ≈560 g | <20 min | Odroid XU4 | Parrot Board Bebop Autopilot | Sensors from Parrot Bebop 2 |
| Kaufmann et al. [21] | 49 cm 0.950 g | N.S. | Aaeon UP Board | Qualcomm Snapdragon Flight N.S. | Intel RS R200 or Front-facing Camera |
| Li et al. [20] | 51 cm ≈400 g | ≈12 min | Parrot P7 dual-core CPU Cortex A9 | Parrot Board Paparazzi | Sensors from Parrot Bebop 1 |
| Jung et al. [22] | 73 cm 2.26 kg | 12 min | NVIDIA Jetson TK1 | In-house | ZED Stereo Camera LiDAR Lite V3 PX4Flow |
| Mohta et al. [25,26] | 75 cm ≈2.8 kg | ≈5 min | Intel NUC i7 | Pixhawk 4 PX4 | 2 × FLIR Chamaleon VN-100 IMU LiDAR Lite V3 Hokuyo UTM-30LX * |
| Quigley et al. [27] | 40 cm ≈1.3 kg | 15 min | NVIDIA Jetson TX2 | Pixhawk N.S. | FPGA with: 2 × 1300 Python Cameras 4 × Fish-eye Cameras |
| AlphaPilot [23] | 70 cm 3.4 kg | N.S. | NVIDIA Jetson Xavier AGX | N.S. N.S. | 2 × Leopard IMX 264 IMU Bosch BMI088 LiDAR Lite V3 |
| Sandino et al. [30] | 73 cm Under 2 kg | 8 min | Aaeon UP Squared Board | Pixhawk 4 PX4 | Intel RS D435 Intel RS T265 HBV-1615 RGB camera TFMini Plus |
| Oleynikova et al. [28] | 70 cm ≈2.5 kg | 15 min | Intel NUC i7 | mRo Pixhawk PX4 | VI Sensor ([35]) Intel RS D435 |
| Kompis et al. [29] | 75 cm ≈1.6 kg ** | 15 min | Intel NUC i5/i7 or NVIDIA Jetson TX2 | Pixhawk 4 PX4 | Stereo VI Units ***: Intel RS D455 Alphasense Core Skybotix VI-Sensor ZED 2 Structure Core Venus RGB BlueFox |
| Campos-Macias et al. [32] | 59 cm 1.3 kg | <20 min | Intel Atom x7-Z8750 | STM32F427V N.S. | Intel RS D435 Intel RS T265 |
| Roggi et al. [24] | 53 cm 2.85 kg | ≈20 min | NVIDIA Jetson Xavier NX | Pixhawk 4 PX4 | ZED Stereo Camera 2 × OpenMV H7 Plus LiDAR Lite V3 |
| Stephens et al. [31] | 60 cm 1.9 kg | ≈20 min | Intel NUC i7 | Pixhawk Pixracer PX4 | Intel RS D435i Intel RS T265 |
| our quadcopter | 54 cm 1.5 kg | ≈12 min | LattePanda Alpha 864s | Pixhawk Durandal PX4 | Intel RS D435i Intel RS T265 TFMini-S |

N.S.: Not Specified. * The LiDAR sensor was used only for dense mapping. ** Weight without the VI camera and its respective supports. *** The platform was tested with different VI cameras, not simultaneously.

The rest of this paper is organized as follows. Section 1.2 reviews fully autonomous UAV frameworks employed in applications similar to the proposed one. Section 2 presents the proposed custom UAV's architecture, including both the hardware and software mod-

ules. In Section 3, the navigation stack is exposed in depth. Simulations and real experiments are described in Section 4, and the conclusions are presented in Section 5.

2. System Overview

This section describes the UAV architecture, consisting of hardware components and software modules, that guarantees autonomous flight while relying only on onboard vision-based sensing. The competition rules forbid the use of LiDARs and GNSS-based positioning systems.

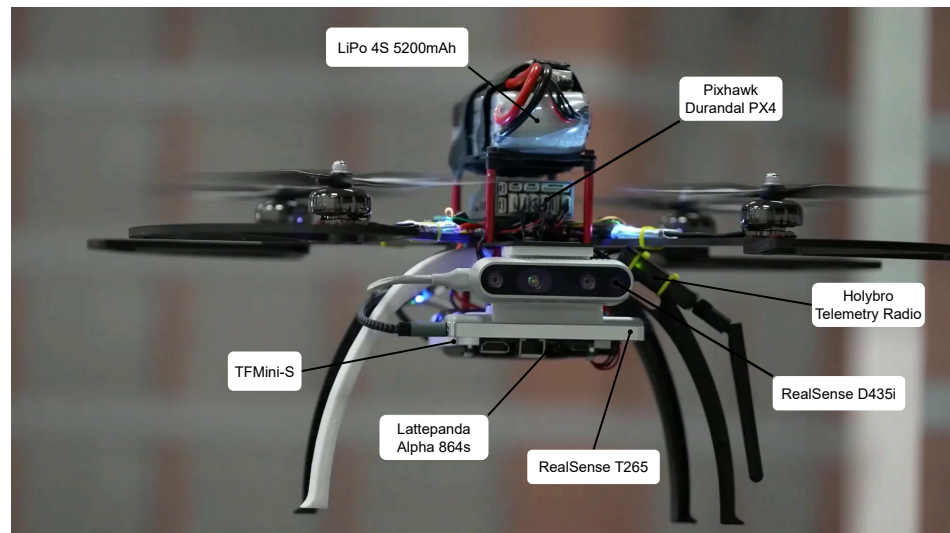


Figure 1. One of the prototypes used in the Leonardo Drone Contest 2022 competition, with 7-inch propellers (wheelbase 51.5 cm from tip to tip) and a LattePanda Alpha 864s for onboard processing. The low-level control board is a Pixhawk Durandal. The drone is equipped with a front-facing camera for stereo depth and an RGB channel and a down-facing fisheye camera for VIO and object tracking.

2.1. Hardware Overview

The MAV employed in this work in Figure 1 is an evolution of the experimental platform presented in our previous work [7], a custom quadrotor built with commercially available off-the-shelf products. The chosen frame is a carbon fiber quadcopter with a wheelbase without propellers of 36 cm, intended for propellers up to 8 inches (maximum wheelbase of up to 54 cm from tip to tip).

The perception task is accomplished by two stereo-cameras, one front-facing for obstacle avoidance and one down-facing for Visual-Inertial Odometry (VIO). In particular, the former is an Intel Realsense D435i camera able to provide depth-maps through stereo-vision information, while the latter is an Intel Realsense T265 camera able to provide MAV's pose in a local reference frame. Both the depth and VIO are computed onboard these devices through proprietary firmware.

Because the environment is only partially known and front pictures must be taken as a task during the mission, it is necessary to employ a front-facing camera for the local planner module. A down-facing camera can work instead as an optical flow module; thanks to its wide fisheye's Field Of View (FOV), the T265 can observe the UAV's surroundings (≈ 5 m wide at an altitude of 1.5 m) both for visual navigation and mission tasks such as ground patrolling. The chosen altimeter is a Benewake TFmini-S that allows distances up to 12 m to be measured in a small form-factor. The flight controller is a Pixhawk Durandal board with PX4 Autopilot firmware v1.12.3.

2.2. Software Overview

The UAV estimates its 3D pose and velocity relative to a local starting position through Visual Inertial Odometry (VIO). The resulting pose estimation is combined with the Pix-

hawk's IMUs and altimeter through an Extended Kalman Filter (EKF) implemented in PX4 and computed inside the flight controller. In particular, the EKF fuses the on-board IMU data (two accelerometers, two gyros, and one magnetometer) with external pose estimates, which in our case are computed by the VIO and the drone altitude sensor. The flight controller board runs multiple instances of the Extended Kalman Filter. By comparing estimate consistencies of the EKF instances, an EKF selector is able to determine which filter output is the more confident, thereby monitoring sensor faults. This enables detection and isolation of faults such as sudden changes in IMU bias, saturation, and stuck data.

As shown in Figure 2, the estimated pose is provided to both the mission planner and global planner. The mission planner, described in depth in Section 3.6, dictates the overall behavior of the UAV during the mission through a Finite State Machine that spans from the initial takeoff to each mission task needed to fulfill the competition or challenge. It requests goal poses in a map frame for the global planner, which is divided into two components. The global planner first performs a fast exploration of a precomputed navigation graph, generated as specified in Section 3.1. With the obtained feasible sequence of intermediate waypoints, the current pose is connected to the goal pose through rectilinear segments, which are proven to lie in the obstacle-free region of the known map. In Section 3.2, the trajectory reference is generated from the interpolation of the provided waypoints while obeying dynamic constraints. Collisions with unknown obstacles or due to trajectory tracking errors are handled by the local planner, which steers the MAV by altering the trajectory reference velocity vector, as explained in Section 3.3. The output trajectory is tracked with velocity commands, generated as in Section 3.4; these are forwarded to the Pixhawk's micro-controller, which converts the center-of-mass desired velocities as commands for the actuator drivers.

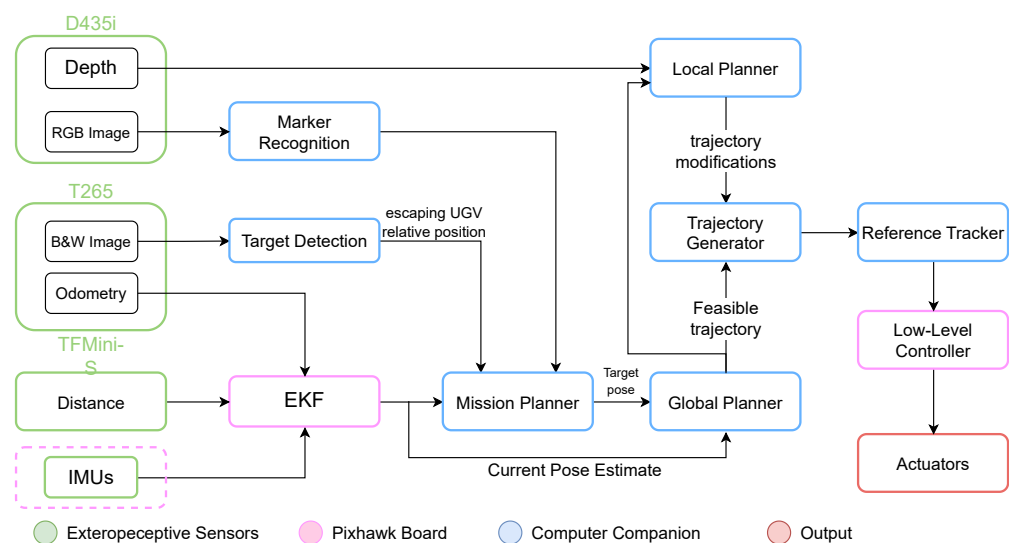


Figure 2. Overall architecture of the system. Data flows from the sensor's inputs (green boxes) to the main modules of the system in their respective boards (pink boxes for the MCU, blue boxes for the companion computer) up to the controlled output (red box).

3. Core Modules of the Navigation Stack

In this section, a detailed overview of the navigation stack is presented. Figure 3 depicts the information flow that composes the overall structure of the navigation stack. Each time a goal end pose is supplied, a pathfinder queries an offline-computed topological graph to find intermediate waypoints that connect the current UAV pose to the desired goal pose. In this passage, the selected crossing points as well as the path that connects them lie in the obstacle-free region of the pre-acquired map. This path is used as a nominal reference for the UAV trajectory, while unpredicted obstacles are handled online by the influence of the obstacle avoidance module on the actual trajectory to be tracked. The waypoints

selected from the topological graph are interpolated by a dynamically feasible trajectory that drives the UAV in the obstacle-free region of the previously known map. Unpredicted obstacles are handled by the local planner, which alters the nominal trajectory by steering the UAV along free regions.

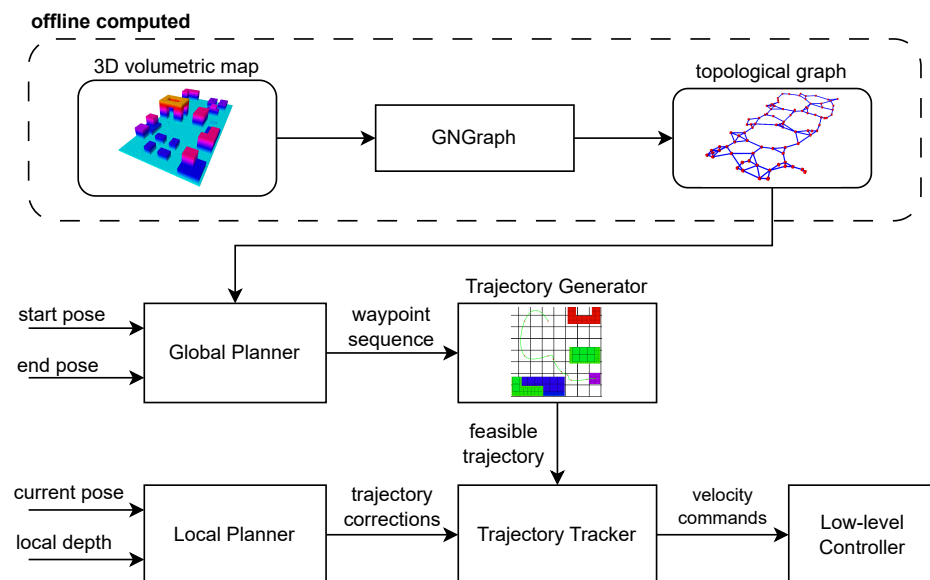


Figure 3. Core elements of the navigation stack. A navigation graph, computed offline over a given 3D map, is queried to produce a feasible trajectory that can be modified by the local planner for avoidance of unknown obstacles.

3.1. Offline Path Planning

The path planning is composed of two successive parts: an offline precomputing of the topological graph and an online trajectory generation and tracking. The former is approached using a graph-based solution, which can model the essential notions of navigational knowledge using route graphs, as presented by Werner et al. [36].

Our previous work, GNGraph [8], was a map-based general method that computed a low-dimensional topological graph of the environment for path planning. This solution exploits an unsupervised Hebbian learning algorithm, and is devised for a fast run-time global planning query on low-computing power embedded systems. The resulting graph is extracted from the environment's Euclidean Signed Distance Field (ESDF) map, and represents a clearance roadmap of the collision-free surroundings close to its 3D Generalized Voronoi Diagram (GVD). A clearance roadmap is a suitable solution for the challenge in question: considering the partially-known environment, the GVD maintains the safest distance from the known obstacles; a collision-avoidance module can then operate in the presence of unknown obstacles.

For this work, the shortest path search algorithm was changed from A* to Dijkstra's algorithm. This decision was taken because A* searches for the optimal path through a heuristic function, while in the case of the contest environment, considering its content size and topology, it is more feasible to run Dijkstra, which guarantees optimality without the need to define a heuristic.

The waypoint sequence between two extreme points, that is, the starting position and goal position, is obtained by first finding the closest vertices of the graph to the starting and goal positions in the environment considering the Euclidean distance between each of them. For this, it is assumed that the segment that connects starting and goal positions to their nearest graph vertices relies on the obstacle-free space. Then, the Dijkstra algorithm finds the shortest path that connects both nodes. It is guaranteed that at least one path is found, as GNGraph ensures the graph's connectivity using the spectral analysis included in its stopping learning criteria.

3.2. Online Trajectory Planning

The second step of the path planning is achieved online, working with the information computed by the method in Section 3.1. While navigating in the environment, the pathfinding algorithm is applied in the topological graph in order to obtain a feasible sequence of waypoints that allows the UAV to move from the current (starting) point to any desired final position. Note that the topological graph dictates only desired 3D positions of the UAV, while the orientation is left free and can be programmed in order to fulfill a secondary task, such as pointing the front-facing camera in any desired direction.

From the obtained sequence of positions, a feasible trajectory that obeys UAV dynamic constraints such as maximum velocities and accelerations must be computed. Time-Optimal Path Parametrization (TOPP) is a class of methods that has the aim of generating a feasible trajectory in the configuration space that interpolate an ordered sequence of desired configurations. The planned trajectory lies in the flat space that is the outcome of the application of the Differential Flatness theory [37] to the quadrotor dynamics. As commonly used in the literature, the chosen flat states in this work are the position of the UAV's center of mass and heading angle, both of which are taken with respect to a fixed frame. Indeed, the 3D position and orientation cannot be controlled independently; the UAV's roll and pitch angles used to obtain the desired position and yaw are constrained by its dynamics. These two elements generate a four-dimensional space that composes the hyperplane in which the trajectory is planned.

The feasible first- and second-order pose derivatives constrain the allowed set, while the ordered sequence of waypoints obtained by the exploration of the GNGraph imposes intermediate cross-points as position-only constraints. The current UAV position $\mathbf{p}_{ref}(0)$, together with the M desired waypoints \mathbf{w}_i and the final position $\mathbf{p}_{ref}(t_f)$ are interpolated with a cubic spline \mathcal{S} that defines the UAV trajectory as $\mathbf{p}_{ref}(t) = \mathcal{S}(t)$ in $t \in [0, t_f]$.

The heading angle can be imposed as a function of the current mission; during the exploration phase, i.e., while the UAV is navigating in previously unknown environments, the heading can be fixed towards the direction of motion by imposing the desired yaw angle ψ as

$$\psi_{ref}(t) = \arctan \frac{v_y(t)}{v_x(t)}, \quad (1)$$

where (v_x, v_y) are the horizontal components of the reference UAV velocity with respect to the map reference frame. This allows the front camera to detect unattended obstacles. During scanning phases, i.e., while the quadrotor is seeking some element of interest in a known direction, the desired heading can instead be directly imposed for each waypoint \mathbf{w}_i .

Among TOPP algorithms, TOPP-RA [38] exploits Reachability Analysis, a linear control systems theory that allows for the identification of feasible states starting from constrained inputs. In the studied case, feasible trajectories for the UAV position and heading angle are generated as time-dependent parametric curves using this method, where velocity and acceleration constraints are enforced over a sampling grid. Together with their time derivatives, the computed trajectories are sampled to generate the time-varying setpoint employed as a reference for the low-level controller, which is discussed in Section 3.4.

3.3. Local Planning for Obstacle Avoidance

The developed local planner module allows the robot to navigate in environments where the map is partially known a priori. The unknown obstacles that compose the environment are represented by columns; these are placed at random points in the environment in a scattered manner such that their composition does not generate a concave configuration. The module interacts with the global planner and sends additive speed references to change the planned trajectory in real-time based on local sensing sensors. The perception of obstacles is carried out by the obstacle detector system (https://github.com/tysik/obstacle_detector (accessed on 17 July 2023)), which reads the depth information from the RGB-D camera and performs a clusterization to generate point

cloud groups that identify the obstacles. Because the size range of the obstacles is known a priori, the clusterization algorithm fits the point cloud data with the expected dimension to detect obstacles.

After the obstacles have been detected within the camera's field of view, the respective centroids are used as reference points for the obstacle avoidance algorithm.

Artificial Potential Field (APF) [39] represents a simple approach for implementing an obstacle avoidance module able to interface easily with a generic global planner. The method is a control strategy for autonomous systems based on the artificial potential function that guides the motion of the system. The APF has been shown to be effective in a wide range of applications, including robotic navigation, formation control, and multi-agent coordination. Our implemented local planner is based on this method and allows the autonomous platform to achieve real-time performance with unpredicted obstacles, showing robustness in the presence of location uncertainty while exploiting solely local sensing information. In addition, the implemented approach only requires a few parameters to be tuned.

The planar position $r_i = [x_i, y_i]^T$ of each i -th detected obstacle is expressed with respect to the local reference system attached to the quadrotor. For each i -th obstacle, the corresponding planar velocity

$$v_i = \begin{cases} -\left[\frac{k_r}{x_i^2}, \frac{k_r}{y_i^2}\right]^T & \text{if } \|r_i\| \leq R \\ [0, 0]^T & \text{otherwise} \end{cases} \quad (2)$$

is computed only if the obstacle is detected within a certain threshold of the distance R from the geometric center of the drone. The gain k_r is the tuning parameter that modulates the motion aggressiveness in avoiding obstacles. The vector modulus of each velocity v_i is saturated if a certain threshold v_{max} in magnitude is exceeded; in this way, the contribution of the repulsion velocity is limited and the safety of autonomous navigation can be ensured by tuned limits. Then, the average speed \bar{v} is computed on the actual repulsive contributions to calculate the overall repulsive velocity for N obstacles

$$\bar{v} = \frac{1}{N} \sum_{i=1}^N v_i. \quad (3)$$

The average operator ensures that the generated repulsive velocity does not exceed the established threshold v_{max} for any distribution of detected obstacles. The computed overall repulsive velocity is sent to the global planner's reference tracking control in order to obtain feasible trajectories that evade unknown obstacles in the environment, as described in the next Subsection.

3.4. Trajectory Reference Tracking and Low-Level Control

The desired trajectory that the UAV has to track, as computed in Section 3.2, can be expressed with the desired position $p_{ref}(t)$ and heading angle $\psi_{ref}(t)$, the function of the time $t \in [t_0, t_f]$, and their time derivatives $v_{ref}(t)$ and $\dot{\psi}_{ref}(t)$.

The lower level control of the UAV is handled by the PX4 autopilot, which communicates desired propeller velocities to the ESCs with several nested control loops. The chosen entry point for the position and velocity references designated by the high-level control is the UAV's center of mass velocity. Here, a PID controller implemented in the PX4 is able to map the desired UAV velocities to actuation reference signals.

Therefore, it is sufficient to produce a desired UAV center-of-mass velocity v_d together with a desired heading rate $\dot{\psi}_d$ to feed to the control loops of the PX4. The position and velocity references produced by the trajectory planning in any time instant t are converted to the desired center of mass velocities as

$$v_d = \dot{p}_{ref}(t) + k_p p_{ref}(t), \quad (4a)$$

$$\dot{\psi}_d = \dot{\psi}_{\text{ref}}(t) + k_\psi \psi_{\text{ref}}(t). \quad (4b)$$

As computed in Section 3.3, the additional contribution of the local planner can be added to Equation (4a) after each position and velocity vector is written in the reference system fixed to the UAV frame. Indeed, velocity variations due to the avoidance of obstacles are identified via local sensing, as they are already expressed in the UAV frame and are not impacted by navigation errors.

3.5. UGV Tracking

The tracking problem is stated as an unknown target moving freely on the ground constrained inside the partially known environment. The target's state and kinematics are entirely unknown; therefore, its dynamics is assumed to be a double integrator in discrete time with a sampling period Δ_t .

The target UGV position is measured using the intrinsic camera matrix to project the position from the pixel 2D image plane to the 3D camera plane and then estimated with a Kalman filter. The state vector is composed of positions and velocities in the horizontal plane, such as $x = [p_x \ p_y \ \dot{p}_x \ \dot{p}_y]^T$, while the state transition matrix A and the observation matrix H are

$$A = \begin{bmatrix} 1 & 0 & \Delta_t & 0 \\ 0 & 1 & 0 & \Delta_t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \quad (5)$$

In the prediction step

$$\begin{aligned} \hat{x}_{k|k-1} &= A\hat{x}_{k-1}, \\ P_{k|k-1} &= AP_{k-1}A^T + Q \end{aligned} \quad (6)$$

the Kalman filter produces the a priori prediction state $\hat{x}_{k|k-1}$ and state estimation error covariance $P_{k|k-1}$ considering the covariance Q of the process noise.

In the update step

$$\begin{aligned} K_k &= P_{k|k-1}H^T(H P_{k|k-1}H^T + R)^{-1}, \\ P_k &= (I_{4 \times 4} - K_k H)P_{k|k-1}, \\ \hat{x}_k &= \hat{x}_{k|k-1} + K_k(y_k - H\hat{x}_{k|k-1}) \end{aligned} \quad (7)$$

the current a priori prediction is combined with the current observation y_k to obtain the a posteriori state estimate \hat{x}_k and the estimate covariance matrix P_k considering the covariance of the observation noise R .

3.6. Mission Planner

The high-level mission logic is managed by means of a Finite State Machine (FSM), the representation of which is shown in Figure 4. The FSM dictates the overall UAV behavior while taking on all the challenge tasks exposed in Section 1.1.

The mission starts in the Idle state. When the UAV is set to the PX4 offboard mode and receives the takeoff command, the system moves to the Hover state until the Start Search command is received, at which point the system moves into the Search state. In the latter state, the drone starts exploring the environment and looking for the moving target in the down-facing camera stream. When the target is found, the system moves into the Tracking state. The tracking of the target must last for 10 s; if the target is lost, the drone starts exploring again (i.e., the system moves again into the Search state). If the tracking is performed continuously for 10 s, the system moves into the Wait for Mission state and the drone stays in hover.

At this point, the Ground Control Station (GCS) can send a mission to the drone. A mission consists of a sequence of actions: either land, or take a photo. In both cases, a target coordinate is specified as a cell of the competition field. The system automatically picks up the first action in the list and moves to the desired location. If the action is a land request, then the drone lands, waits 10 s, and takes off again; if the action is a photo request, the drone starts an exploration phase looking for the target that must be shot. During the challenge, the environment is partially known in terms of obstacles; however, we only have an approximate location of the target to be shot. For this reason, a series of waypoints is used as a reference for the exploration phase. When the target is found, the drone takes a picture and the operator at the GCS accepts the picture or sends the retry command.

When an action is completed, the system again enters the `Select Action` state and the loop repeats until all actions have been executed. Finally, the drone moves to a safe location (away from obstacles) and performs a final landing before returning to the `Idle` state.

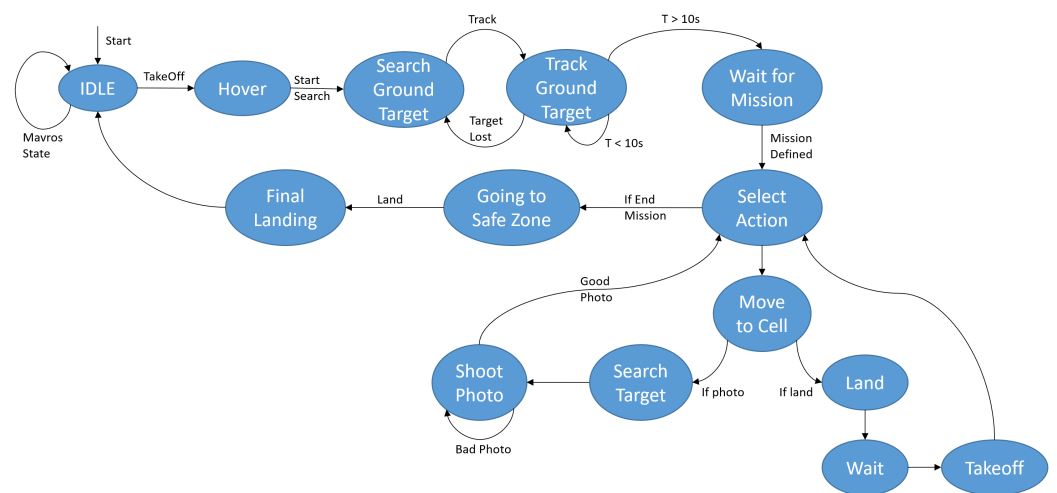


Figure 4. Schematic representation of the FSM managing the whole mission logic. The blue circles represent states, while the arrows represent transitions.

4. Evaluation and Experiments

The effectiveness of the different modules described in this work was first tested in simulated environments, then validated in real-world scenarios. The PX4 Software In The Loop (PX4-SITL) allows testing and comparison of the proposed algorithm in simulation and in the real world without any modification. PX4-SITL is based on the RotorS simulator [40], a Gazebo-based simulation environment that provides reliable multirotor models. The simulated quadrotor, mimicking the real one, is equipped with a face-forward stereo camera that provides real-time color images and depths of the simulated environment as well as with a downfacing camera.

The partially known 3D map of the Drone Contest is reproduced in Blender and imported in Gazebo with a dynamical spawning of the additional unknown obstacles in random positions; these are regulated as stated in Section 1.1. The escaping UGV is implemented in simulation with a Turtlebot 3 (<https://www.turtlebot.com/turtlebot3> (accessed on 17 July 2023)), whereas during the competition the real robot is a Roomba (<https://www.irobot.it/roomba> (accessed on 17 July 2023)) with original firmware. Both are constrained in rectangular subsections of the map's ground.

4.1. Global Trajectory Planner

Example trajectories are shown in Figure 5, where a top view of the competition field is represented. In Figure 5a, the computed trajectory moves the UAV from the starting cell I12 to the left side of the map in order to look for the escaping UGV. The trajectory is generated through the method described in Section 3.2 by interpolation of the red waypoints. The ordered list is obtained prepending the user-defined sequence G8-G6-D6-D8, with the

waypoint sequence selected from the GNGraph obtained as in Section 3.1, which connects the starting point (I12 to the first user-defined search position (G8).

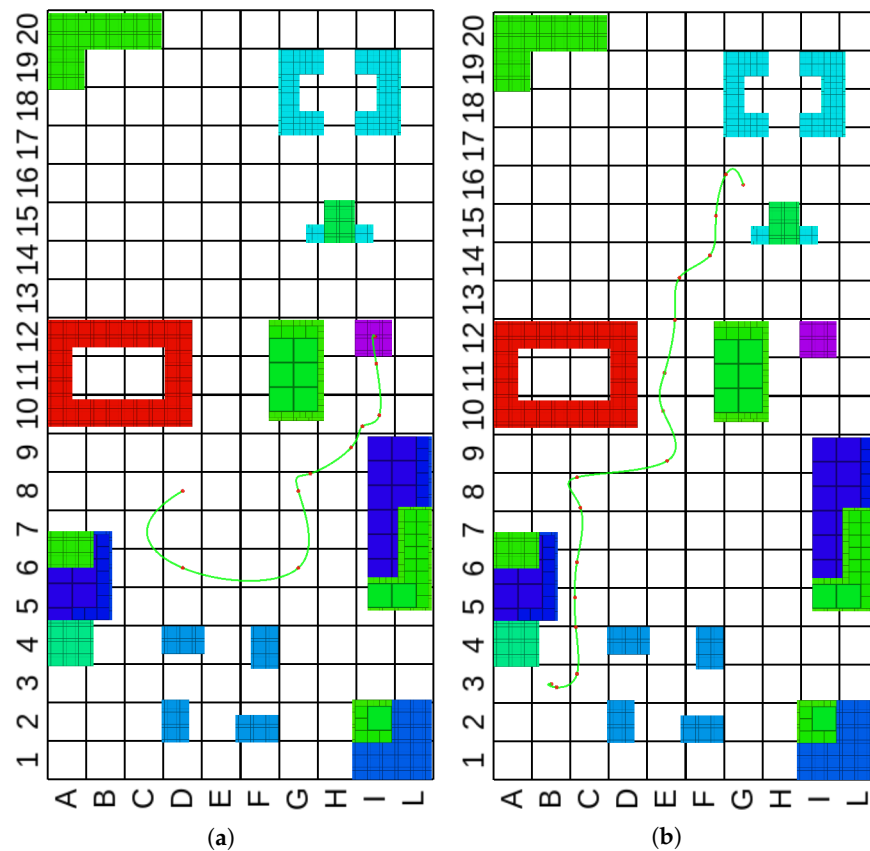


Figure 5. UAV planned trajectories, in red, for (a) beginning UGV search performed when taking off from the starting square I12 and (b) long-distance navigation from B3 to G16. The red dots are the intermediate waypoints selected from the GNGraph, that the trajectory has to interpolate.

Figure 5b shows the planned trajectory from the start position B3 to the target G16. The entire list of intermediate waypoints is inferred from the GNGraph algorithm. At any time instant, the heading angle is set tangent to the green curve in order to always monitor the direction of motion for unknown obstacles with the forward-facing camera.

These examples show the versatility of the algorithm with a different number of waypoints that can be explicitly requested, inferred from the topological graph, or both. Short (<5 m) and long (up to 5 m) trajectories are planned on the onboard companion computer in less than 700 ms.

4.2. Local Planner

The corrections of a simulated obstacle avoidance experiment are shown in Figures 6. The trajectory replanning is due to the presence of two unknown cylindrical obstacles, represented in red in the top-down view and in white in the gazebo environment. Obstacles are placed in the simulation such that the desired trajectory (in green) intersects them. When obstacles are detected, the local planner module makes the drone deflect the reference trajectory, sending additive speed commands to the global planner. The parameters of the local planning module are tuned in simulation to guarantee UAV readiness in avoiding the detected obstacles and at the same time to ensure compliance with the desired kinematic constraints in terms of the maximum speed limits of the vehicle. From the plots in Figure 7, it is apparent that the vehicle does not violate the kinematic constraints of the global planner (± 0.5 m/s) when no obstacles are detected; in the presence of obstacles, the vehicle must not exceed the sum of the limits of the global and local planner (± 1.1 m/s).

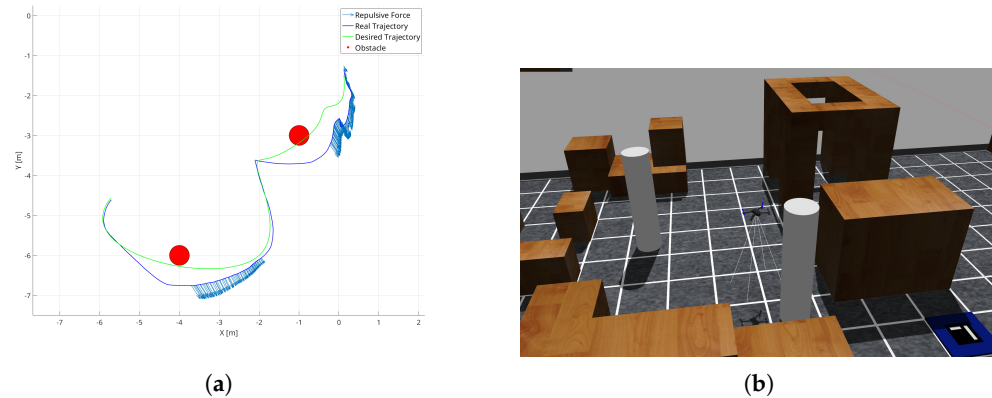


Figure 6. Local Planner module performing obstacle avoidance starting from the local position $[0, -1]$ in simulation. The desired trajectory in green is the reference provided to the quadrotor, while the real (corrected by obstacle avoidance) trajectory is in blue. (a) Trajectory due to obstacle avoidance, top view and (b) scenario viewed in Gazebo Simulator.

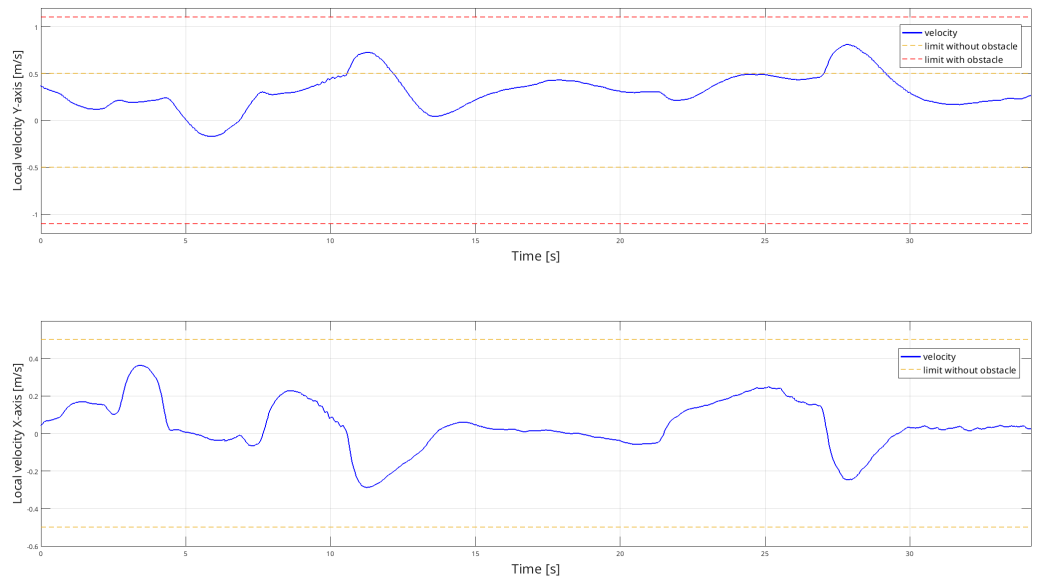


Figure 7. Vehicle speed expressed in the local coordinate frame during the obstacle avoidance experiment in Gazebo Simulator. The vehicle respects the speed thresholds set both in the absence and in the presence of obstacles.

4.3. Overall System in Real Environment

The proposed framework was used during the competition in the indoor environment of the 2022 Leonardo Drone Contest. With the help of the Rviz tool, we developed a graphical user interface (Figure 8) to monitor the progress of the mission and the status of the drone in real time. On the right of the GUI, the operator can see the streams from the forward-facing camera and the undistorted down-facing camera, which are enabled for searching markers. In the center it can be seen where the drone is located with respect to the map, which is known a priori. For each planning, the selected nodes of the roadmap are colored in red while the computed trajectory of the global planner is shown in green. The cyan sphere indicates the current reference position for the low-level tracking control. The unknown obstacle mapped by the detector module is represented by concentric red and green cylinders. On the left panel, the operator has the possibility of deciding which box the drone has to land in by typing the coordinates on the interface.

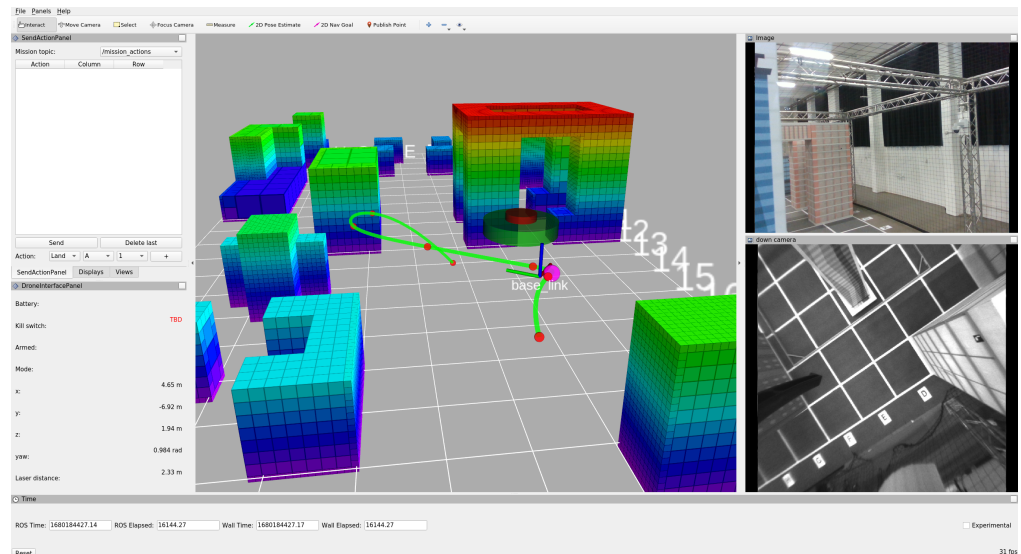


Figure 8. RVIZ GUI used to monitor the drone’s mission in real-life experiments. In the right column, both camera FOVs are shown. In the central window, the environment’s known obstacles are represented with voxels and an unknown obstacle is depicted with a dark red and green cylinder. The drone’s trajectory and reference points are shown in light green and red, respectively.

The expected performance was verified in the real scenarios in terms of both readiness to avoid unknown obstacles and ability to return to the desired trajectory when no more obstacles were detected. Figure 9 show a sequence of manoeuvres along the planned trajectory in the presence of an unknown obstacle.

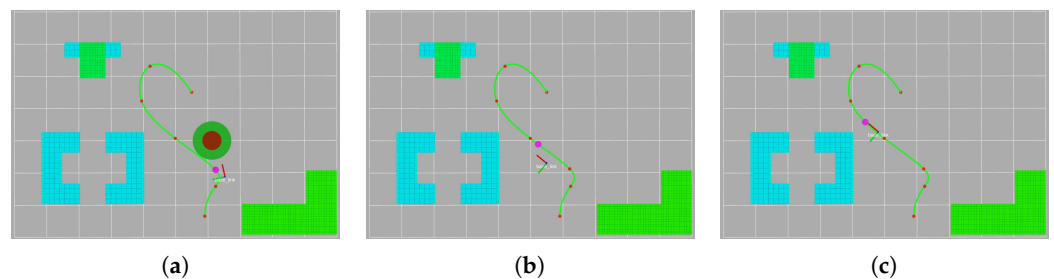


Figure 9. A sequence of images during the tracking of a reference trajectory in the presence of an unknown obstacle. The UAV detects the obstacle (a), sends speed commands to the global planner to change the planned trajectory (b), and returns to the previous trajectory after avoiding the obstacle (c).

5. Conclusions

This paper has presented a versatile solution for addressing complex autonomous tasks, specifically focusing on the case of the 2022 Leonardo Drone Contest. The proposed vision-based framework demonstrates the ability to navigate efficiently in partially unknown GNSS-denied real-world environments. By leveraging an offline-computed topological graph and online obstacle avoidance, the system effectively handles both planned and unpredicted obstacles.

The complexity of the mission tasks was tackled by using a Finite State Machine that effectively activates the needed modules of the stack. The entire software stack was designed, developed, and tested entirely in the ROS framework, with the middleware exploited on the drone companion computer to run and arrange the communication among all the software modules.

Localization and navigation capabilities were initially tested in simulation and their robustness was subsequently validated in real-world scenarios, specifically, the Leonardo Drone Contest challenge, further confirming the effectiveness of the proposed system. The proposed framework and the choices related to the hardware and the sensors represent

a widely tested innovative solution that ensures autonomous flight in real-world GNSS-denied environments even in partially unknown scenarios.

In summary, this work provides a comprehensive and adaptable solution for autonomous tasks that has demonstrated its efficacy in challenging environments, thereby setting a foundation for further advances in the field of UAV autonomy.

Author Contributions: Conceptualization, M.M., M.T.L., E.P.H.-A., G.B. and M.S.; methodology, M.M., M.T.L., E.P.H.-A. and G.B.; software, M.M., M.T.L., E.P.H.-A. and G.B.; validation, M.M., M.T.L., E.P.H.-A. and G.B.; resources, E.P.H.-A., M.S. and C.A.A.; writing—original draft preparation, M.M., M.T.L., E.P.H.-A., G.B. and M.S.; writing—review and editing, M.M., M.T.L. and M.S.; supervision, M.S.; project administration, M.S.; funding acquisition, M.S. and C.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not yet publicly available at the time of writing but they will.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

| | |
|-------|------------------------------------|
| UAV | Unmanned Aerial Vehicle |
| MAV | Micro-Aerial Vehicle |
| UGV | Unmanned Ground Vehicle |
| GNSS | Global Navigation Satellite System |
| ROS | Robot Operating System |
| RViz | ROS visualization tool |
| SAR | Search And Rescue |
| FSM | Finite State Machine |
| FCU | Flight Control Unit |
| FPGA | Field-Programmable Gate Array |
| RS | RealSense |
| LiDAR | Light Detection and Ranging |
| IMU | Inertial Measurement Unit |
| VIO | Visual Inertial Odometry |
| FOV | Field Of View |
| EKF | Extended Kalman Filter |
| TOPP | Time-Optimal Path Parametrization |
| APF | Artificial Potential Field |
| ESC | Electronic Speed Controller |
| GCS | Ground Control Station |
| GUI | Graphical User Interface |

References

1. Silano, G.; Baca, T.; Penicka, R.; Liuzza, D.; Saska, M. Power Line Inspection Tasks with Multi-Aerial Robot Systems via Signal Temporal Logic Specifications. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4169–4176. [\[CrossRef\]](#)
2. Agha, A.; Otsu, K.; Morrell, B.; Fan, D.D.; Thakker, R.; Santamaria-Navarro, A.; Kim, S.K.; Bouman, A.; Lei, X.; Edlund, J.; et al. NeBula: Quest for Robotic Autonomy in Challenging Environments; TEAM CoSTAR at the DARPA Subterranean Challenge. *arXiv* **2021**. [\[CrossRef\]](#)
3. Hudson, N.; Talbot, F.; Cox, M.; Williams, J.; Hines, T.; Pitt, A.; Wood, B.; Frousheger, D.; Lo Surdo, K.; Molnar, T.; et al. Heterogeneous Ground and Air Platforms, Homogeneous Sensing: Team CSIRO Data61's Approach to the DARPA Subterranean Challenge. *Field Robot.* **2022**, *2*, 595–636. [\[CrossRef\]](#)
4. Rouček, T.; Pecka, M.; Čížek, P.; Petříček, T.; Bayer, J.; Šalanský, V.; Azayev, T.; Heřt, D.; Petrlík, M.; Bába, T.; et al. System for multi-robotic exploration of underground environments CTU-CRAS-NORLAB in the DARPA Subterranean Challenge. *arXiv* **2021**. [\[CrossRef\]](#)

5. Tranzatto, M.; Dharmadhikari, M.; Bernreiter, L.; Camurri, M.; Khattak, S.; Mascarich, F.; Pfreundschuh, P.; Wisth, D.; Zimmermann, S.; Kulkarni, M.; et al. Team CERBERUS Wins the DARPA Subterranean Challenge: Technical Overview and Lessons Learned. *arXiv* **2022**. [[CrossRef](#)]
6. Balaram, B.; Canham, T.; Duncan, C.; Grip, H.F.; Johnson, W.; Maki, J.; Quon, A.; Stern, R.; Zhu, D. Mars helicopter technology demonstrator. In Proceedings of the 2018 AIAA Atmospheric Flight Mechanics Conference, Kissimmee, FL, USA, 8–12 January 2018; p. 0023.
7. Alarcón, E.P.H.; Ghavifekr, D.B.; Baris, G.; Mugnai, M.; Satler, M.; Avizzano, C.A. An Efficient Object-Oriented Exploration Algorithm for Unmanned Aerial Vehicles. In Proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS), Athens, Greece, 15–18 June 2021; pp. 330–337. [[CrossRef](#)]
8. Herrera-Alarcón, E.; Satler, M.; Vannucci, M.; Avizzano, C. GNGraph: Self-Organizing Maps for Autonomous Aerial Vehicle Planning. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10721–10728. [[CrossRef](#)]
9. Shen, S.; Michael, N.; Kumar, V. Autonomous multi-floor indoor navigation with a computationally constrained MAV. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 20–25. [[CrossRef](#)]
10. Tomic, T.; Schmid, K.; Lutz, P.; Domel, A.; Kassecker, M.; Mair, E.; Grix, I.; Ruess, F.; Suppa, M.; Burschka, D. Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue. *IEEE Robot. Autom. Mag.* **2012**, *19*, 46–56. [[CrossRef](#)]
11. Fraundorfer, F.; Heng, L.; Honegger, D.; Lee, G.H.; Meier, L.; Tanskanen, P.; Pollefeys, M. Vision-based autonomous mapping and exploration using a quadrotor MAV. In Proceedings of the IEEE International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 7–12 October 2012; pp. 4557–4564. [[CrossRef](#)]
12. Meier, L.; Tanskanen, P.; Heng, L.; Lee, G.H.; Fraundorfer, F.; Pollefeys, M. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Auton. Robot.* **2012**, *33*, 21–39. [[CrossRef](#)]
13. Meier, L.; Honegger, D.; Pollefeys, M. PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms. In Proceedings of the IEEE International Conference on Robotics and Automation, Seattle, WA, USA, 26–30 May 2015; pp. 6235–6240. [[CrossRef](#)]
14. Loiano, G.; Brunner, C.; McGrath, G.; Kumar, V. Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU. *IEEE Robot. Autom. Lett.* **2017**, *2*, 404–411. [[CrossRef](#)]
15. Ge, R.; Lee, M.; Radhakrishnan, V.; Zhou, Y.; Li, G.; Loiano, G. Vision-based Relative Detection and Tracking for Teams of Micro Aerial Vehicles. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 380–387. [[CrossRef](#)]
16. Thakur, D.; Tao, Y.; Li, R.; Zhou, A.; Kushleyev, A.; Kumar, V. Swarm of Inexpensive Heterogeneous Micro Aerial Vehicles. In *Proceedings of the Experimental Robotics*; Siciliano, B., Laschi, C., Khatib, O., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 413–423.
17. Liu, X.; Chen, S.W.; Nardari, G.V.; Qu, C.; Ojeda, F.C.; Taylor, C.J.; Kumar, V. Challenges and Opportunities for Autonomous Micro-UAVs in Precision Agriculture. *IEEE Micro* **2022**, *42*, 61–68. [[CrossRef](#)]
18. Moon, H.; Martinez-Carranza, J.; Cieslewski, T.; Faessler, M.; Falanga, D.; Simovic, A.; Scaramuzza, D.; Li, S.; Ozo, M.; De Wagter, C.; et al. Challenges and implemented technologies used in autonomous drone racing. *Intell. Serv. Robot.* **2019**, *12*, 137–148. [[CrossRef](#)]
19. Rojas-Perez, L.O.; Martinez-Carranza, J. Metric monocular SLAM and colour segmentation for multiple obstacle avoidance in autonomous flight. In Proceedings of the 2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS), Linköping, Sweden, 3–5 October 2017; pp. 234–239. [[CrossRef](#)]
20. Li, S.; Ozo, M.M.; De Wagter, C.; de Croon, G.C. Autonomous drone race: A computationally efficient vision-based navigation and control strategy. *Robot. Auton. Syst.* **2020**, *133*, 103621. [[CrossRef](#)]
21. Kaufmann, E.; Gehrig, M.; Foehn, P.; Ranftl, R.; Dosovitskiy, A.; Koltun, V.; Scaramuzza, D. Beauty and the Beast: Optimal Methods Meet Learning for Drone Racing. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 690–696. [[CrossRef](#)]
22. Jung, S.; Cho, S.; Lee, D.; Lee, H.; Shim, D.H. A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge. *J. Field Robot.* **2018**, *35*, 146–166. [[CrossRef](#)]
23. Foehn, P.; Brescianini, D.; Kaufmann, E.; Cieslewski, T.; Gehrig, M.; Muglikar, M.; Scaramuzza, D. AlphaPilot: Autonomous drone racing. *Auton. Robot.* **2022**, *46*, 307–320. [[CrossRef](#)]
24. Roggi, G.; Meraglia, S.; Lovera, M. Leonardo Drone Contest 2021: Politecnico di Milano team architecture. In Proceedings of the 2022 International Conference on Unmanned Aircraft Systems, ICUAS 2022, Dubrovnik, Croatia, 21–24 June 2022; pp. 191–196. [[CrossRef](#)]
25. Mohta, K.; Watterson, M.; Mulgaonkar, Y.; Liu, S.; Qu, C.; Makineni, A.; Saulnier, K.; Sun, K.; Zhu, A.; Delmerico, J.; et al. Fast, autonomous flight in GPS-denied and cluttered environments. *J. Field Robot.* **2018**, *35*, 101–120. [[CrossRef](#)]
26. Mohta, K.; Sun, K.; Liu, S.; Watterson, M.; Pfrommer, B.; Svacha, J.; Mulgaonkar, Y.; Taylor, C.J.; Kumar, V. Experiments in Fast, Autonomous, GPS-Denied Quadrotor Flight. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 7832–7839. [[CrossRef](#)]

27. Quigley, M.; Mohta, K.; Shivakumar, S.S.; Watterson, M.; Mulgaonkar, Y.; Arguedas, M.; Sun, K.; Liu, S.; Pfrommer, B.; Kumar, V.; et al. The open vision computer: An integrated sensing and compute system for mobile robots. In Proceedings of the IEEE International Conference on Robotics and Automation, Montreal, QC, Canada, 20–24 May 2019; pp. 1834–1840. [\[CrossRef\]](#)
28. Oleynikova, H.; Lanegger, C.; Taylor, Z.; Pantic, M.; Millane, A.; Siegwart, R.; Nieto, J. An open-source system for vision-based micro-aerial vehicle mapping, planning, and flight in cluttered environments. *J. Field Robot.* **2020**, *37*, 642–666. [\[CrossRef\]](#)
29. Kompis, Y.; Bartolomei, L.; Chli, M. Fully Autonomous Live 3D Reconstruction with an MAV: Hardware- and Software-Setup, 2021-12-02. In Proceedings of the 9th International Conference on 3D Vision (3DV 2021), Online, 1–3 December 2021. [\[CrossRef\]](#)
30. Sandino, J.; Vanegas, F.; Maire, F.; Caccetta, P.; Sanderson, C.; Gonzalez, F. UAV framework for autonomous onboard navigation and people/object detection in cluttered indoor environments. *Remote Sens.* **2020**, *12*, 3386. [\[CrossRef\]](#)
31. Stephens, B.; Nguyen, H.N.; Hamaza, S.; Kovac, M. An Integrated Framework for Autonomous Sensor Placement With Aerial Robots. *IEEE/ASME Trans. Mechatron.* **2022**, *28*, 38–49. [\[CrossRef\]](#)
32. Campos-Macías, L.; Aldana-López, R.; de la Guardia, R.; Parra-Vilchis, J.I.; Gómez-Gutiérrez, D. Autonomous navigation of MAVs in unknown cluttered environments. *J. Field Robot.* **2021**, *38*, 307–326.
33. Ko, C.; Han, S.; Choi, M.; Kim, K.S. Integrated path planning and tracking control of autonomous vehicle for collision avoidance based on model predictive control and potential field. In Proceedings of the 2020 20th International Conference on Control, Automation and Systems (ICCAS), Busan, Republic of Korea, 13–16 October 2020; pp. 956–961.
34. Nieuwenhuisen, M.; Droschel, D.; Beul, M.; Behnke, S. Autonomous navigation for micro aerial vehicles in complex GNSS-denied environments. *J. Intell. Robot. Syst.* **2016**, *84*, 199–216. [\[CrossRef\]](#)
35. Nikolic, J.; Rehder, J.; Burri, M.; Gohl, P.; Leutenegger, S.; Furgale, P.T.; Siegwart, R. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 431–437. [\[CrossRef\]](#)
36. Werner, S.; Krieg-Brückner, B.; Herrmann, T. Modelling navigational knowledge by route graphs. In *Spatial Cognition II*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 295–316.
37. Mellinger, D.; Kumar, V. Minimum snap trajectory generation and control for quadrotors. In Proceedings of the IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 2520–2525. [\[CrossRef\]](#)
38. Pham, H.; Pham, Q.C. A New Approach to Time-Optimal Path Parameterization Based on Reachability Analysis. *IEEE Trans. Robot.* **2018**, *34*, 645–659. [\[CrossRef\]](#)
39. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In Proceedings of the 1985 IEEE International Conference on Robotics and Automation, St. Louis, MO, USA, 25–28 March 1985; Volume 2, pp. 500–505.
40. Furrer, F.; Burri, M.; Achtelik, M.; Siegwart, R. Rotors—A modular gazebo mav simulator framework. *RObot Oper. Syst. Complet. Ref.* **2016**, *1*, 595–625.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.