Journal of Intelligent & Robotic Systems (2023) 108:23
https://doi.org/10.1007/s10846-023-01855-w

REGULAR PAPER

Leonardo Drone Contest Autonomous Drone Competition: Overview, Results, and Lessons Learned from Politecnico di Milano Team

Gabriele Roggi[1] · Salvatore Meraglia[1] · Marco Lovera[1]

Received: 27 October 2022 / Accepted: 16 March 2023 / Published online: 13 June 2023

Abstract
In this paper, the Politecnico di Milano solutions proposed for the *Leonardo Drone Contest* (LDC) are presented. The Leonardo Drone Contest is an annual autonomous drone competition among universities, which has already seen the conclusion of its second edition. In each edition, the participating teams were asked to design and build an autonomous multicopter, capable of accomplishing complex tasks in an indoor urban-like environment. To reach this goal, the designed systems should be capable of navigating in a Global Navigation Satellite System (GNSS)-denied environment with autonomous decision making, online planning and collision avoidance capabilities. In this light, the authors describe the first two editions of the competition, i.e., their rules, objectives and overview of the proposed solutions. While the first edition is presented as relevant for the experience and takeaways acquired from it, the second edition solution is analyzed in detail, providing both the simulation and experimental results obtained.

Keywords Unmanned Aerial Vehicles (UAVs) · Autonomous systems · Mobile robotics · Competition

1 Introduction

In recent years, the research on autonomous Unmanned Aerial Vehicles (UAVs) has seen great interest. Autonomous systems can play an important role in many applications, e.g., search-and-rescue [43], inspection [35], surveillance [14] and mapping [17].

Autonomous UAVs, as many other robot systems, are composed of the standard building blocks of state estimation, control, mapping and planning. In the latest years, as computational power grew and more efficient algorithms were proposed, vision started to be used for state estimation of aerial vehicles [6, 10]. With the improvement of vision algorithms, e.g., [20, 34], flight in unstructured 3D environments became possible [15, 42, 44], and cameras are now the most used sensor for state estimation in aerial vehicles [32]. In order to move in an partially known or unknown environment, a robot, in addition to having reliable and robust state estimation and control algorithms, needs to build an accurate map and generate collision-free trajectories on the map itself. These aspects have been tackled by different research groups, e.g., [36, 53]. Recently, advancements in all these research directions were integrated in fully autonomous UAVs capable of flying relying only on onboard sensors, even in cluttered environments. In this regard, in [32], the authors designed a full navigation system allowing the drone to reach a goal location, while avoiding obstacles. At the same time, in [37], the authors developed a vision-based autonomous UAV capable of GNSS-denied navigation, contributing also to important advancements for 3D global planning on real maps and local planning with re-planning capabilities.

At the same time, in the robotics community, there has been a significant growth in the number of challenge prizes and competitions. This has been done with the aim of stimulating innovation to meet a defined challenge and to provide solutions to problems that matter to roboticists and society [13].

In particular, drone racing has seen autonomous systems catching up fast with human performance. In this framework, while the first drone racing competition has been organized during the IROS conference in 2016 [33], the most important

✉ Gabriele Roggi
gabriele.roggi@polimi.it

Salvatore Meraglia
salvatore.meraglia@polimi.it

Marco Lovera
marco.lovera@polimi.it

[1] Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano, Via La Masa 34, Milano 20158, Italy

event has been the 2019 Lockheed Martin AlphaPilot challenge [1]. This challenge led to the first season of Artificial Intelligence Robotic Racing (AIRR), co-organized with the Drone Racing League (DRL) for human pilots [12]. In the literature, some of the approaches taken by the teams during this competition are available, e.g., [11, 18] and [38]. Other than drone racing, autonomous UAVs have been object of competitions in several conferences, e.g., IMAV (International Micro Air Vehicle) [40], and in the Mohamed Bin Zayed International Robotic Challenge (MBZIRC) [4]. In this context, a variety of autonomous drone operations has been explored ranging from wall building [31] to ground fires extinguishing [52] and from baloon popping [8] to the capture of other flying objects [51].

With similar objectives, namely to encourage the development of Artificial Intelligence (AI) applied to Unmanned Aerial Systems (UAS), Leonardo S.p.A. designed and launched, the *Leonardo Drone Contest* [3] in collaboration with six Italian universities: Politecnico di Milano, Politecnico di Torino, Università di Bologna, Scuola Superiore Sant'Anna di Pisa, Università degli Studi di Roma Tor Vergata and Università degli Studi Federico II di Napoli.

The universities' teams have been engaged for three years in the development of autonomous drone systems. The designed multicopter should be capable of navigating in a GNSS-denied environment with autonomous decision making, online planning and collision avoidance capabilities. It is worth also mentioning that no LiDAR scanners were allowed, meaning that for navigation and planning only visual sensors or range finders could have been employed.

Some of the contributions of this paper are:

- We provide two different hardware and software solutions for fully autonomous UAVs, addressing, in particular, indoor GNSS-denied navigation and collision avoidance in cluttered environments.
- We present a navigation solution for environments equipped with visual markers. The employed approach is shown as able to provide accurate position state estimates over long flights.
- With respect to relevant literature on fully autonomous UAV architectures, we also present a decision making layer for making the drone perform complex tasks.
- For each competition, we provide educational contributions describing the technical challenges faced and possible solutions.

This paper is organized as follows. In Sections 2 and 3, the first and second competition formats and rules are respectively presented. For each competition, the hardware setup and an overview of the software solution are introduced. For the first edition, we highlight some takeaways for future developments, while, in Section 4, the solution proposed for the second edition is presented in detail, discussing each of the implemented sub-modules, i.e., navigation, decision making and planning and control. In Section 5, we briefly describe the simulation environment and we present the experimental setting. Then, in Section 6, the results obtained in both simulations and experimental activities are discussed. Finally, in Section 7, the encountered technical challenges and the planned future activities are summarized. We refer the reader to our previous work [39], where the results obtained during the second contest itself are shown. Compared to the previous publication, this one discusses the selection of the employed open source libraries and adds details to the solution implementation. Furthermore, this paper also presents the architecture proposed for the first competition, highlighting the issues and lessons learned. This work also describes the simulation environment employed for the development of the solution and the results obtained on it. Finally, since in the competition no motion capture system was available, experiments have been performed in a laboratory environment with ground-truth for assessing localization and mapping performance.

Notation We define frames and notations that are used throughout the paper. We indicate the *map* frame with {M}, i.e., an East-North-Up (ENU) fixed frame; {O} represents the *odom* frame, i.e., an ENU frame which drifts over time with respect to {M}; {B} is a Forward-Left-Up (FLU) *body*-fixed frame; {C} is the *camera*-fixed frame and {T} is the visual marker-fixed frame. $R_{A/B} \in \mathrm{SO}(3)$ is the rotation matrix describing the attitude of the {A} frame with respect to the {B} frame, while $p_{A/B} \in \mathbb{R}^3$ is the position of the frame {A} with respect to frame {B}, resolved in {B}.

2 First Competition: Format

The first edition of the *Leonardo Drone Contest* took place in Turin on 17-18 September 2020. The competition was held in the indoor urban-like environment, shown in Fig. 1.

The field dimensions were $20\,\mathrm{m} \times 10\,\mathrm{m} \times 3\,\mathrm{m}$ and it was delimited by a net. The obstacles were at most three meters high with passages of at least one meter among them. The obstacles' shape, dimension and position were unknown. Six poles of different colors were placed at known positions on the field as represented in Fig. 2.

The contest was composed of two phases, held respectively in the two days of the competition. In the first phase, the drone had the aim of exploring autonomously the environment. At the same time, the drone task was to localize 10 QR codes, serving also as landing pads for the second phase. The landing pad was made of a $1\,\mathrm{m} \times 1\,\mathrm{m}$ blue square containing a $50\,\mathrm{cm} \times 50\,\mathrm{cm}$ QR marker. The QR markers contained a unique alphanumeric string each.

Fig. 1 The real competition field in Turin consisting of cardboard buildings with glued textures (from [3])

Just before the start of the second day, the teams were given a list of 5 QR codes by the competition judges. This list indicated the ordered sequence of pads to be reached by the drone and on which it was meant to land. Hence, the teams had the opportunity of planning an optimized path and upload it on the drone. The team performing the largest number of consecutive valid landings would have been identified as the winner. Note that a landing was considered valid if all the drone's contact points with the ground were inside the 1 m × 1 m landing pad.

2.1 Hardware Setup

The drone, specifically designed for the competition and developed in collaboration with the Politecnico di Milano spin-off company ANT-X [2], codename ROG-1, is a coaxial octocopter (Fig. 3). Its dimensions are 500 × 500 × 300 mm for a takeoff weight of 3.75 kg. The drone is equipped with a Pixhawk 4 Flight Control Unit (FCU) and an NVIDIA Jetson TX2 companion computer mounted on a Connect Tech Orbitty Carrier board. It is powered by a 16 000 mAh 4S Lipo battery, reaching a total flight time of 19 minutes.

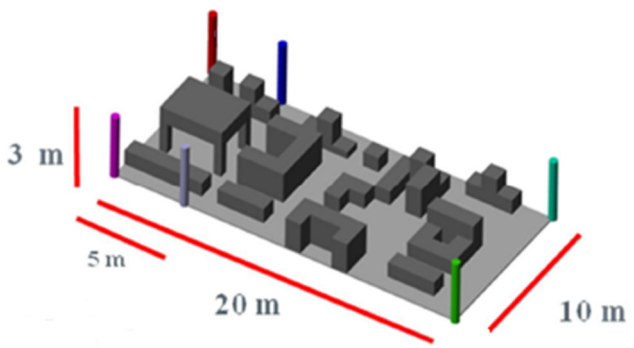

Fig. 2 Rendering example of the competition field indicating positions and colors of poles

The sensor suite is composed by a forward-looking Stereolabs ZED stereo camera inclined slightly downward (15 degrees with respect to the horizontal plane), a downward-looking OpenMV H7 Plus monocular camera and a TeraRanger Tower Evo equipped with 8 TeraRanger Evo 60 m range finders.

2.2 Software and Algorithms Overview

First of all, the Robot Operating System (ROS) was employed as framework for all the high-level software running on the UAV. For what concerns state estimation, since no GNSS signal was available, stereo visual odometry was used for determining the position and heading of the drone, which was then fused with the Inertial Measurement Unit (IMU) information for obtaining an estimate of the drone pose in the *odom* frame. Note that, while the visual odometry was computed on the companion computer, the fusion process was carried out on the FCU using the EKF2, i.e., the Extended Kalman Filter available on the PX4[1] firmware.

The open source RTAB-Map package [29] was employed for the localization in the *map* frame, using as input the stereo camera images. RTAB-Map stands for Real-Time Appearance-Based Mapping and it is a RGB-D, Stereo and LiDAR graph-based SLAM approach, in which loop closures are found using a bag-of-word approach (see [28] for more details). Note that the map produced by the RTAB-Map algorithm in the first phase of the competition was saved in the form of an Octomap [25].

An alternative localization solution, based on the colored poles, was originally designed. In this respect, the conditions of the competition resembled the Robocups. In the literature, many different localization approaches for these competitions were available [24, 48]. We implemented an EKF-localization approach, similar to [47], which exploited

[1] https://px4.io/

Fig. 3 The UAV platform for LDC first edition (ROG-1)

visual odometry in the kinematic model and the colored poles as visual landmarks. Distance, elevation and azimuth of the landmarks were computed via color segmentation and bounding box computation on monocular cameras' images. Even if the approach showed good performance in a simulation environment, which will be presented in Section 5, the solution was not implemented on the platform due to the following limitations. First, the solution would have required additional cameras to properly work. Possible occlusions due to the obstacles present on the field limited greatly the amount of measurements obtained, leading to the need of increasing the available field of view. Processing more camera images would have also implied a greater computation overhead for the overall system. Furthermore, while the azimuth and elevation were very accurate, distance estimates were not. This aspect limited the accuracy of the proposed solution. The robustness of the solution was also difficult to assess, since it was not possible to test the system on the competition field before the actual contest. A concern regarded possible changes in lighting conditions, which could have led to issues in the color segmentation task. Finally, adopting this strategy would have reduced the generality of the solution and limited its deployment in other application domains.

Two different planning algorithms were used depending on the phase of the competition considered. During the exploration phase, the Receding Horizon Next Best View Planner [9] was employed. Starting from stereo camera images, the aim of this algorithm was to produce a collision-free path from the current pose to the viewpoint which maximized the acquired knowledge of the environment. The computed path, in the form of a list of position set-points, was then sent to the FCU through *mavros*[2]. While the environment was explored, the downward-looking monocular camera was used for detecting the QR codes scattered around the field. At the same time, the QR codes' positions were computed and saved, given the best estimate of the drone position in the *map* reference frame. The overall system architecture for this phase is depicted in Fig. 4.

In the second day, once the list of landing pads to be reached was known, a path was planned using an offline A*, which took as input the retrieved Octomap. The path was then uploaded on the UAV. On-board the UAV an online local planner was running. Its aim was to avoid possible collisions, which could arise due to uncertainties in both the map and the UAV state estimate. The local planner was an implementation of Vector Field Histogram (VFH*) [49], which utilized the range information coming from the TeraRanger Tower. Due to the planar nature of the sensor, a 2D implementation of the planner was used, even if a 3D extension was proposed in the literature [50]. The resulting architecture for the landing phase is shown in Fig. 5.

As far as the results were concerned, on the first day, the UAV autonomously explored about 60% of the competition field, finding 7 out of the 10 QR markers placed on the field. On the second day, 4 out of 5 valid landings were performed, of which 3 consecutive, leading to the best result obtained

[2] https://github.com/mavlink/mavros

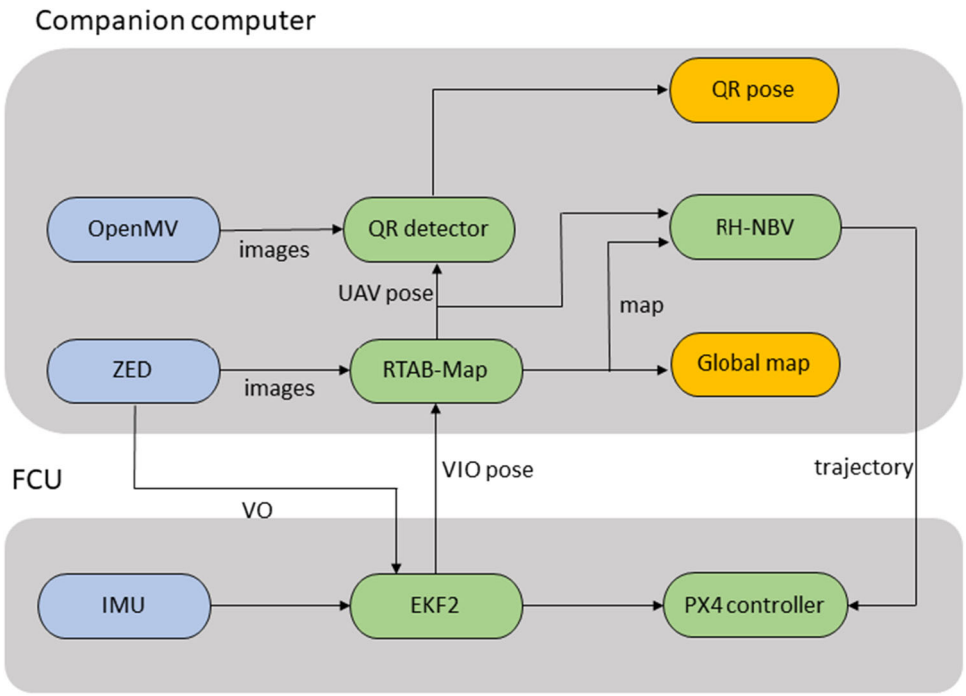

Fig. 4 System architecture for the exploration phase

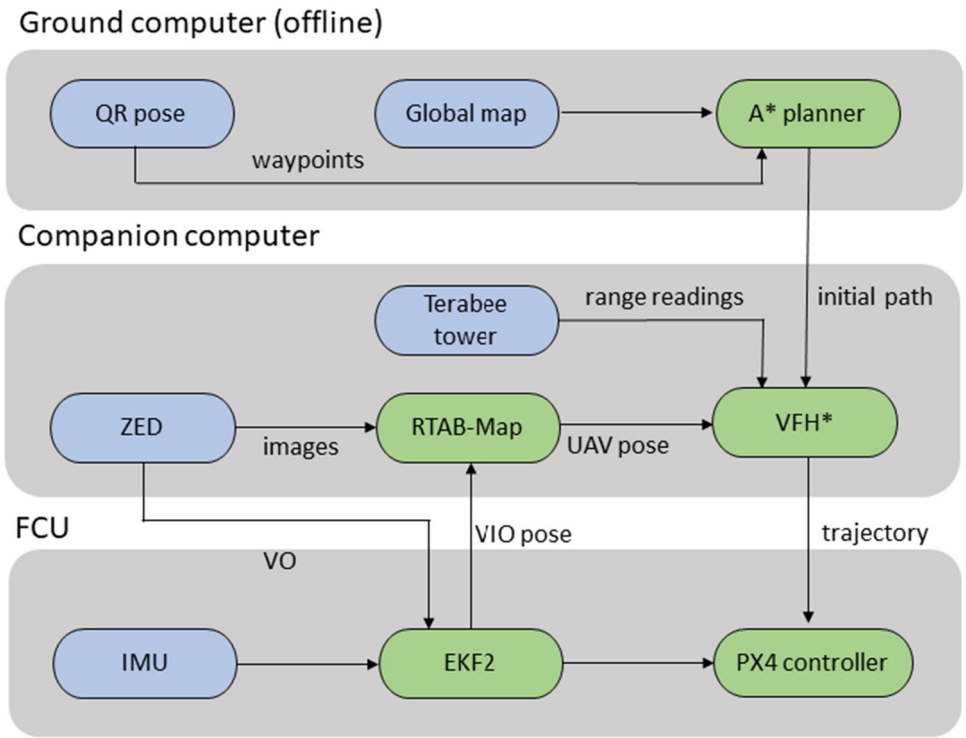

Fig. 5 System architecture for the landing phase

and the win for our team. Despite the achievements, some problems were evident:

- The major problem was related to the altitude estimation. The visual odometry showed a very strong drift and the SLAM algorithm was not able to compensate for it, leading to considerable variations in the altitude kept by the drone.
- While the local planner was successful for collision avoidance, it showed some problems in cluttered environments. The local nature of the algorithm made the vehicle, sometimes, unable to reach the goal, starting to visit the same place over and over. In addition to this, sub-optimal paths, in terms of distance travelled, were often selected.

3 Second Competition: Format

The second edition took place in Turin on 28-30 September 2021, in an indoor environment similar to the one of the previous competition. The contest was composed of four rounds over three days. In each round the drone had to autonomously take off and explore the environment searching for a specific ground robot, which in turn carried information about the task the drone had to perform. It is worth pointing out that, this time, some initial knowledge of the field was given to the teams in the form of a map, with associated East-North reference frame, as shown in Fig. 6.

As in the first year competition, the field dimensions were 20 m × 10 m, with a maximum available height of 3 m (due to a net on top of the field). The heights of the obstacles in the map were also known. The numbered squares on the map represented the landing pads, which were 1 m × 1 m cyan squares each containing a 50 cm × 50 cm ArUco marker (with corresponding identifier).

Three ground robots (in the form of vacuum cleaners), each identified by a unique 19 cm × 19 cm ArUco marker (ranging from 21 to 26), were moving randomly in unknown confined regions on the field. The information provided by the robots were given in the form of a string of 10 digits from 0 to 9. The i-th number indicated the reward associated with a valid landing on the landing pad identified by ArUco number i. The ground robot with the available information and an example of landing pad are shown in Fig. 7.

Two ground robots, whose identification ArUco code numbers were communicated to the team before starting the round, were collaborative agents. The remaining one was the *intruder* that carried the exact rewards associated to each landing. The drone exploration was assisted by a fixed pan-tilt-zoom (PTZ) camera placed in correspondence of the cross icon in Fig. 6, which could be controlled by the teams. After having found the *intruder* robot, the drone had to send a visible picture of the rewards to the team's ground control station. This action had to be executed within 30 minutes from the start of the round. At this point, the sequence of landing spots to maximize the reward had to be selected. There were no requirements for the computation of the sequence. Indeed, the sequence could be either selected by the team based on their judgement or it could be computed through an optimization problem. After having determined the sequence, the drone had to reach and land on top of the pads in the given order. The landing was considered valid if all the UAV contact points with the ground were inside the 1 m × 1 m square. The points corresponding to valid landings were summed up for obtaining the round result. The team obtaining the highest amount of points in the three (out of four) best rounds was proclaimed the winner. It is worth pointing out that, in each round, the UAV takeoff position was represented by a different

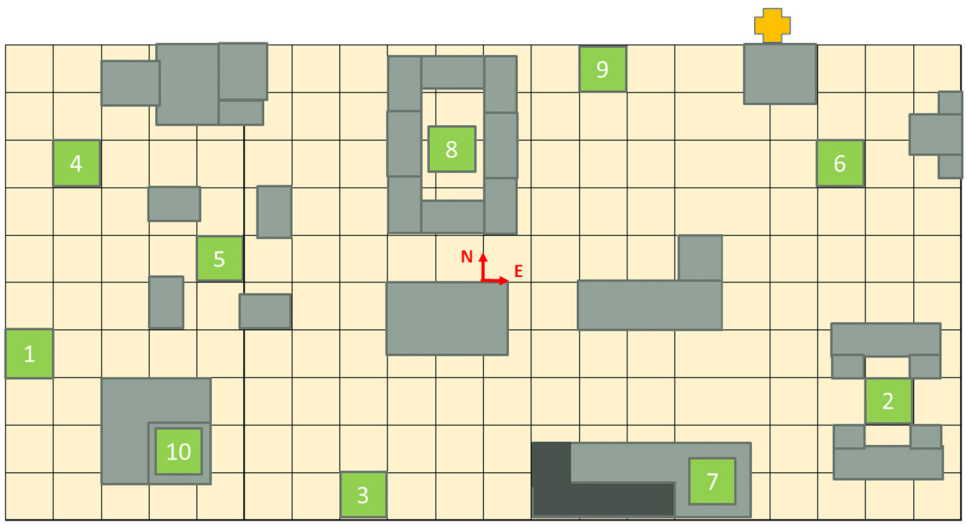

Fig. 6 Initial knowledge of the environment

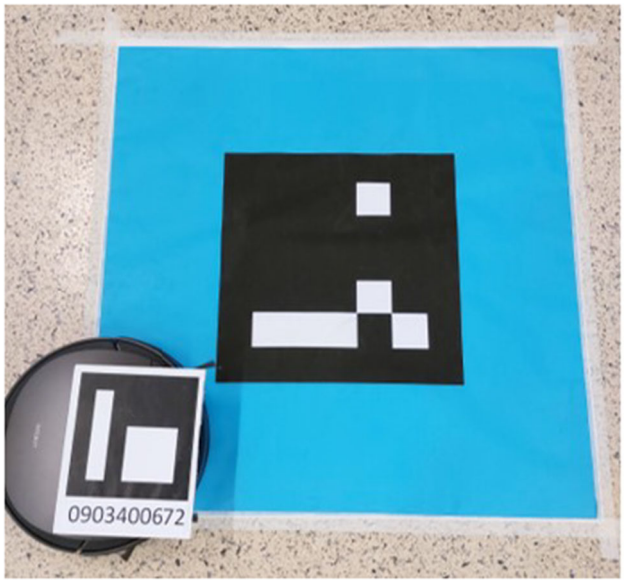

Fig. 7 The ground robot and landing pad

landing pad, communicated to the teams before the start of the run. The reward string and the ground robots' region of motion varied across rounds.

3.1 Hardware Setup

The UAV platform, designed as evolution of the drone which competed in the first edition, was developed in collaboration with ANT-X as well. It is again a coaxial octocopter (Fig. 8), codename ROG-2, but with a different propulsion system. Its dimensions and weight has been reduced leading respectively to a $430 \times 300 \times 200$ mm drone weighing 3.03 kg at takeoff. While the Pixhawk 4 FCU has been adopted again, the companion computer has been replaced with the more powerful NVIDIA Jetson Xavier NX. This second platform is equipped with the same battery of the previous version and it reaches a total flight time of about 20 minutes.

Fig. 8 The UAV platform for LDC second edition (ROG-2)

The sensor suite is composed by a forward-looking Stereolabs ZED stereo camera inclined slightly downward (15 degrees with respect to the horizontal plane), a Lidar-Lite v3 laser range finder and two OpenMV H7 Plus monocular cameras, one forward-looking and one downward-looking.

3.2 Software and Algorithms Overview

Some software components were kept for this improved solution. The ROS framework was still used, thanks also to the capabilities of publishing and receiving messages through TCP transport, which allowed to run a subset of the nodes on the companion computer and some on the ground control station. In particular, the ground control station was used for the teleoperation of the fixed PTZ camera and for sending the sequence of landing pads to be visited by the drone. Clearly, the control station was also employed for receiving and visualizing telemetry data.

For state estimation purposes, the same solution was adopted. However, in order to enhance the accuracy of the altitude estimation, in this competition edition, a range finder was employed as an altimeter. A localization algorithm was custom developed for computing an estimate of the drone position in the *map* frame. To reach this goal, the known positions of the ArUcos in the world map and the downward-looking monocular camera images were exploited.

This time, the RTAB-Map algorithm was only used for mapping purposes. The resulting map, in the form of an Octomap, has been employed for planning collision-free paths and trajectories to be fed to the PX4 drone controller.

The planner module, running online and on-board the robot, was composed by a Djikstra's global planner and an A* local planner. The global planner was meant for computing a list of intermediate waypoints for reaching the goal using the *a priori* information on the environment. The local planner, as opposed to the first competition, relied only on visual information, without the need of additional range finders.

Note also that an additional forward-looking monocular camera was equipped to search for ground robots. This camera acquired images at a different resolution with respect to the stereo camera used for navigation. In this framework, a decision making algorithm has been custom developed for identifying waypoints to be reached in order to maximize the probability of finding the ground robots. The algorithm had access to information coming from both the monocular cameras available on-board and on the fixed PTZ camera.

4 Second Competition: Detailed Solution

In this section, the proposed solution for the second edition of the contest is presented in detail. The overall architecture is composed by three main subsystems:

- **Navigation**. It takes as inputs the sensor measurements and returns as outputs the drone pose in the *map* frame and a global map.
- **Decision-making**. It manages the high-level objectives of the mission. It takes as inputs preliminary information on the environment and outputs the desired goal positions.
- **Planning and control**. The global map and the goal positions are used for planning collision free paths and trajectories to be fed to the drone controller.

4.1 Navigation

First of all, multiple ROS-compatible implementations have been analyzed for computing the visual odometry given the stereo images coming from the ZED stereo camera. In particular, we have considered:

- RTAB-Map, which was already presented for its visual SLAM implementation. The package also provides a stereo visual odometry pipeline (no loop closing enabled), which is the one we will consider in this analysis.
- ORB-SLAM2 [34] is a graph-based stereo visual SLAM algorithm compatible with monocular, stereo and RGB-D cameras. The SLAM system is based on keyframes containing a set of features and the camera pose. A local and global Bundle Adjustment (BA) are used to correct a recent set of keyframes and to optimize the map and trajectory respectively.
- LibViso2 [21] is a feature-based VO library for monocular and stereo cameras. Features are extracted by filtering the images with a corner and blob mask and performing non-maximum and non-minimum suppression on the filtered images.
- ZED-VO is the proprietary visual odometry software provided in the ZED SDK[3].

Given a stereo odometry implementation, its position and yaw output are then fused in the EKF2. The EKF2 returns the estimates of position and attitude of the UAV in the *odom* frame, $p_{B/O} = [p_x, p_y, p_z]^\top$ and $R_{B/O}$ respectively, which will be used for feedback control.

For what concerns the localization, a Kalman Filter (KF), which fuses information coming from drone's odometry, monocular cameras images and laser altimeter, has been implemented. The state of the filter is the bias term $x = b_{M/O} = [b_x, b_y, b_z]^\top$ that is the difference between the position of the robot in the *odom* reference frame and the *map* reference frame.

Assuming the evolution of the bias as a random walk process, the motion model results:

$$\dot{x} = \dot{b}_{M/O} = \eta_w, \tag{1}$$

where η_w is a white Gaussian noise with Power Spectral Density (PSD) $S_w(\omega) = W$. We write it in discrete time state space form as:

$$x_k = F x_{k-1} + w_{k-1}, \tag{2}$$

where $F = I$, i.e., the identity matrix, and $w_k \sim \mathcal{N}(0, Q)$ is a Gaussian random vector with covariance Q.

The images from the forward-looking and downward-looking cameras are analyzed through an ArUco marker detector based on the OpenCV[4] library. Once a marker related to a landing pad has been detected, its position with respect to the camera, written in the *camera* frame, is computed. The measurement model at instant k has been written as:

$$\begin{aligned} y_c &= p_{T/C} \\ &= R_{B/C}(R_{B/M}^\top(p_{T/M} - (p_{B/O} + b_{M/O}))) + p_{B/C} + v_c, \end{aligned} \tag{3}$$

where v_c is a zero mean Gaussian white noise with covariance R_c: $v_c \sim \mathcal{N}(0, R_c)$. This operation is possible by knowing the ArUco marker position $p_{T/M}$ in the approximate world map and assuming $R_{B/M} = R_{B/O}$. This was justified by the fact that the attitude $R_{B/O}$ showed a very slow drift during the experimental campaign carried out inside the Flying Arena for Rotorcraft Technologies (FlyART) of Politecnico di Milano.

In order to increase the altitude estimation accuracy, the laser altimeter measurements are also used. The measurement model at instant k is:

$$y_l = b_z + p_z + v_l, \tag{4}$$

where $v_l \sim \mathcal{N}(0, R_l)$ is a zero mean white Gaussian noise with covariance R_l. However, this model does not take into account the presence of obstacles on the field. Thus, when flying over obstacles, the measured distance does not correspond to the UAV altitude. To overcome this issue, a measurement outliers detector has been applied to interrupt the fusion process whenever the drone flies over an obstacle. Note that a similar approach has been also employed in

[3] https://www.stereolabs.com/developers/release/

[4] https://opencv.org/

the attempt of reconstructing the scale of monocular vision using range finder information in [16]. Our method was also used for rejecting possible outliers in the ArUco relative pose estimates. In particular, this check is performed through a χ^2-test based on the Mahalanobis distance of the measurement innovation [41].

Writing in state space form the measurement model obtained combining (3) and (4), we have:

$$y_k = [y_c, y_l]_k^\top = Cx_k + v_k, \tag{5}$$

with $v_k = [v_c, v_l]_k^\top$. Furthermore, we group the measurement noise covariance matrices as:

$$R = \begin{bmatrix} R_c & 0 \\ 0 & R_l \end{bmatrix}. \tag{6}$$

Writing the filter in the usual prediction-correction form, the following equations have been employed:

$$\begin{aligned} \hat{x}_k^- &= F\hat{x}_{k-1}^+ \\ P_k^- &= FP_{k-1}^+F^\top + Q \\ z_k &= y_k - H_k\hat{x}_k^- \\ Z_k &= H_kP_k^-H_k^\top + R \\ K_k &= P_k^-H_k^\top Z_k^{-1} \\ \hat{x}_k^+ &= \hat{x}_k^- + K_kz_k \\ P_k^+ &= (I - K_kH_k)P_k^-. \end{aligned}$$

where P^- and P^+ are the *a priori* and *a posteriori* state estimation error covariance matrices respectively, and $\hat{x}^-$ and $\hat{x}^+$ are the *a priori* and *a posteriori* estimates of the state.

Inliers are validated by checking the Normalized Estimation Error Squared (NEES) after the computation of z_k, namely:

$$z_k^\top Z_k^{-1} z_k \leq \chi_{th}^2, \tag{7}$$

with χ_{th}^2 equal to the 0.95 probability quantile of the χ^2 distribution. If the measurement passes the test, we proceed by computing the Kalman gain K_k and by updating the filter state and covariance, otherwise the measurement is discarded.

Finally, knowing the drone pose in the *map* frame given by the localization and having the ZED point cloud, RTAB-Map is used for producing a 3D map in the form of an Octomap. The overall navigation architecture is depicted in Fig. 9.

4.2 Decision-making

The decision-making module is responsible for assigning the waypoints to the planning and control module. In particular, the waypoints are computed through different approaches based on the output of a state machine, which manages the different phases of the competition. The workflow of the state machine and associated inputs are shown in Fig. 10 and described in the following.

When the operator sends the starting signal, the decision-making module provides a takeoff setpoint.

After takeoff, the UAV is supposed to find the *intruder* robot. The problem at hand, in general terms, can be formulated as the one of finding an optimal patrolling strategy [7]. Despite the existence of solutions to this problem in similar frameworks [26, 27], our problem can be simplified considering the fact that the ground robots to be monitored are constrained to move in relatively small areas. As a consequence, the drone does not need to visit multiple times the same area for verifying if an *intruder* has appeared. Thus, the problem can be treated as an art gallery problem or coverage planning problem in a known environment [23, 46]. In this work, we propose a greedy approach for steering the UAV towards the area of the field where the probability of finding a ground robot is greater. In the state machine of Fig. 10 this procedure is called *probabilistic exploration*. At each iteration, this algorithm outputs the (x, y) coordinates which maximize the probability of finding the *intruder* robot (the z-coordinate of the generated waypoints will be equal to the drone flight altitude). The probability values are stored in a grid map of 1 m resolution. This probability map, denoted as P_{map}, is initialized considering the approximate world map of Fig. 6 and the following assumptions:

1. The ground robots will not be positioned on obstacles or too close to them.
2. It will be more likely to find the ground robots in large open spaces.

Thus, the initial probability values are computed based on the number of free, i.e., with no obstacles, contiguous cells. A graphical representation of the initial map is shown in Fig. 11.

During the environment exploration, the information coming from all the available cameras, i.e., downward-looking and forward-looking cameras available on-board and the fixed PTZ camera, are used for updating the probability values. In particular, under certain conditions, the probability values of some cells are decreased to a user defined value. As a consequence, the probability values of the other cells in the map will increase to have an overall properly defined probability distribution (for a more detailed description, see Algorithm 1). The algorithm takes as input the initial probability map P_{map}.

At each iteration, the images coming from the cameras are processed searching for ArUco markers corresponding to ground robots. If one of these fiducial markers is found,

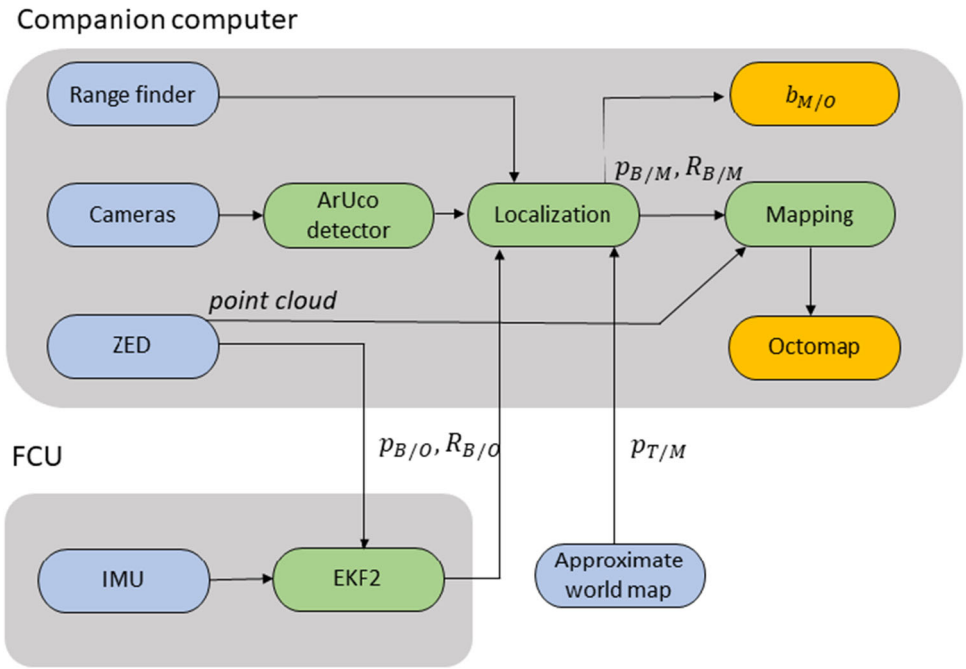

Fig. 9 Navigation architecture

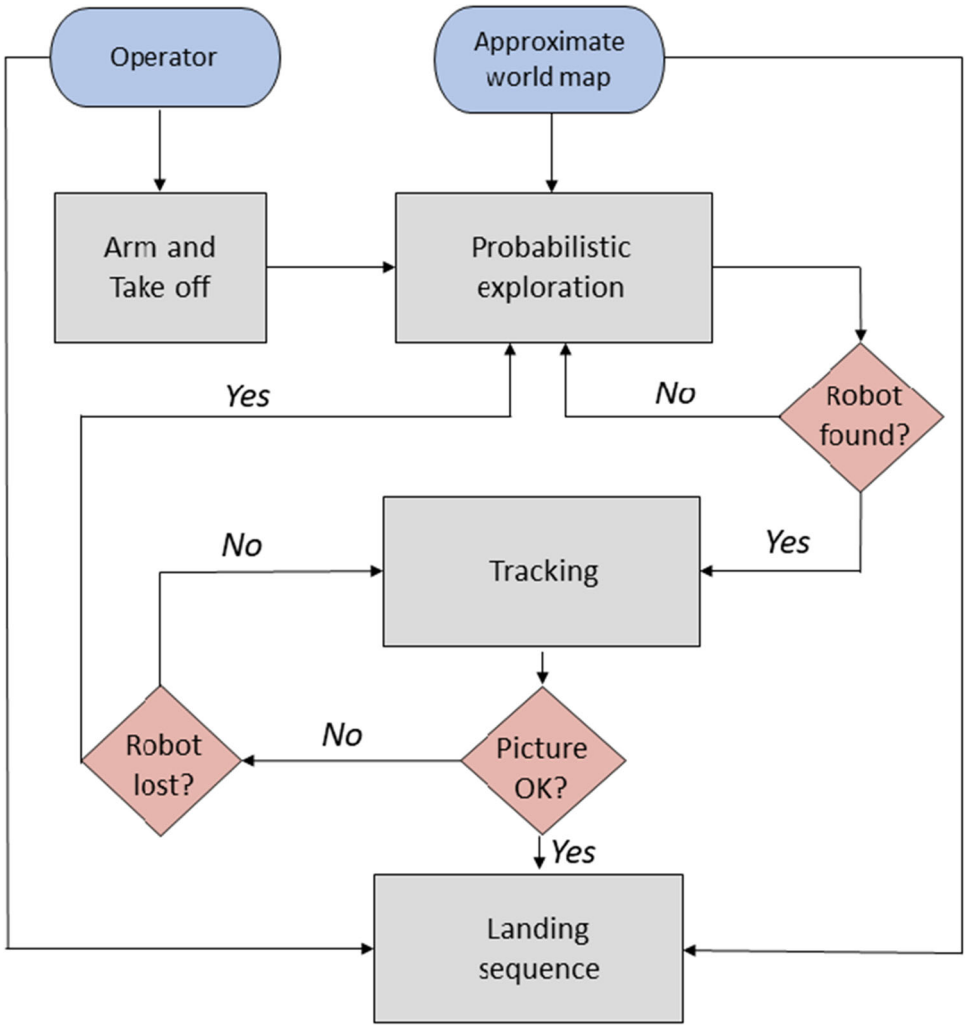

Fig. 10 Decision making architecture

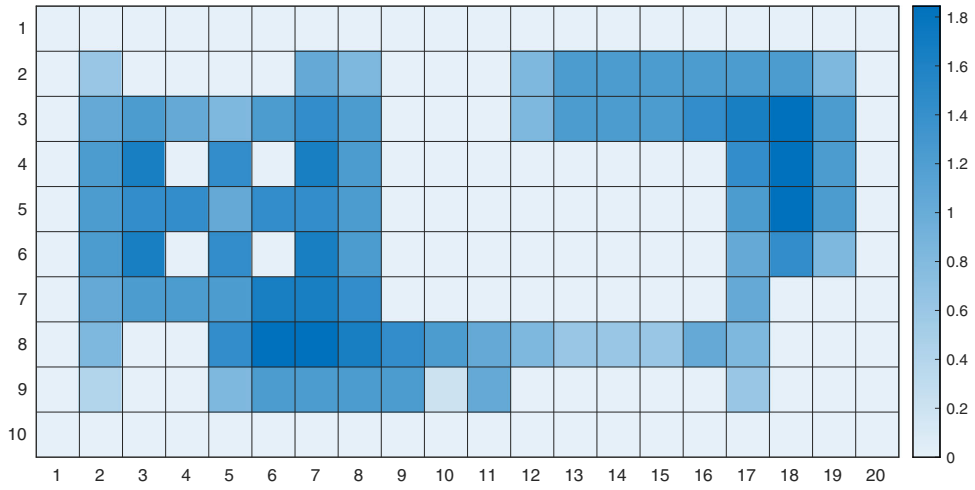

Fig. 11 Initial probability map

Algorithm 1 Probabilistic exploration.

Input P_{map}
Output waypoint

while *intruder* is not found **do**
  Process image and compute $p_{T/C}$ and $aruco_{id}$
  if not $aruco_{id}$ **then**
    P_{map} = DiscountCameraFovs(P_{map})
    wp = BestWaypoint(P_{map})
    return wp
  else
    if $aruco_{id}$ is *intruder* **then**
      $p_{T/M}$ = Camera2Map($p_{T/C}$)
      $wp = p_{T/M}$
      return wp
    else
      $p_{T/M}$ = Camera2Map($p_{T/C}$)
      P_{map} = DiscountArea(P_{map}, $p_{T/M}$)
      wp = BestWaypoint(P_{map})
      return wp
    end if
  end if
end while

the corresponding identified $aruco_{id}$ and position in the camera frame $p_{T/C}$ are retrieved. If the $aruco_{id}$ is the *intruder* one, i.e., not one of the two identification numbers communicated before the start of the round, the probability exploration algorithm outputs the waypoint corresponding to the position occupied by the ground robot in the *map* and terminates (the decision-making module will shift to the *tracking* phase). In this regard, the *Camera2Map* function is employed for finding the position of the ArUCo in the *map* frame, given its position in the camera frame. This can be done by applying the inverse measurement model of equation (3), namely:

$$p_{T/M} = p_{B/M} + R_{B/M} R_{B/C}^{\top} (p_{T/C} - p_{B/C}). \tag{8}$$

On the other hand, if the $aruco_{id}$ corresponds to a collaborative ground robot, the probability values of the area surrounding the robot are decreased (*DiscountArea* function). This update rule is based on the assumption that the robots will not be close to each other.

After the map update, the *BestWaypoint* function retrieves the waypoint corresponding to the cell of maximum probability:

$$\arg\max_{(i,j)} \; p_{map}(i, j). \tag{9}$$

In the case of no ArUco markers identified in the images, the *DiscountCameraFovs* function is employed. The discounted cells depend on the camera which has produced the observation. For the downward-looking camera, only the cell occupied by the drone at update time is discounted. This rule has been implemented considering that the resolution of the map is approximately equal to the area covered by the downward-looking camera field of view with the drone flying at an altitude of about 1 m. For the forward and fixed PTZ camera, first, the direction of the camera in the *map* frame is computed and approximated to the nearest 45 degree angle. Then, in the computed direction, the cells on which a robot could be correctly identified, if present, are discounted. These cells range from a minimum value d^-, determined by the field of view of the camera and the orientation of the camera itself, and a maximum value d^+, which corresponds to the maximum distance at which an ArUco marker can be detected (see Fig. 12).

In particular, d^- is computed as:

$$d^- = (p_z + b_z) / \tan(\theta_{cam} + fov_v/2) \tag{10}$$

where θ_{cam} and fov_v are respectively the pitch angle with respect to the horizontal plane and the vertical field of view of the considered camera. On the other hand, d^+ is determined experimentally.

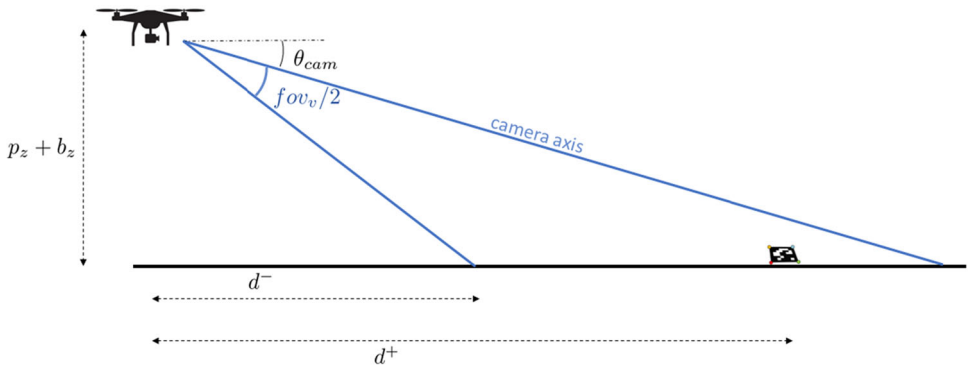

Fig. 12 Graphical representation of high oblique aerial field of view

Also in this case, the *BestWaypoint* function is called for computing the output waypoint after the map update.

During the *tracking* state, the decision-making module keeps computing the robot position in the *map* frame according to function *Camera2Map*. This phase ends when:

1. the robot has been lost, i.e., it is outside the field of view of the cameras. In this latter case, the tracking cannot continue since no information about the position of the ground robot can be obtained. Thus, the state machine will resort to the *probabilistic exploration* phase in the neighbourhood of the current position of the drone.
2. the robot is in the field of view and a readable picture of the reward string has been taken. In this case, the operator will decide the landing sequence to follow in order to maximize the reward considering the remaining endurance of the drone and the probability of success of each landing (this could be also computed through an optimization process on the drone available hardware). The output will be a sequence of waypoints in the initial approximate world map of the considered landing pad.

Finally, if the robot is in the field of view, but the reward string is not readable, the state machine will remain in the *tracking* phase.

4.3 Planning and Control

The planning and control module is meant for generating collision-free paths and trajectories that should then be followed by the UAV. For what concerns the planner, it takes as inputs the waypoints $p^0_{B/M}$ generated by the high-level decision making. An overview of the architecture is shown in Fig. 13.

It is a common cascaded approach composed by a global planner and a local planner. The global planner generates a path from the current position of the drone to the given waypoint. To this aim, it exploits the knowledge of the world approximated map with a coarse resolution (1 m). The planner is a 2D implementation of Djikstra's shortest path algorithm. Successively, the generated path is pruned for removing the collinear points. The remaining points are, one at a time, used as goal into a local planner. The local planner, based on the A* algorithm, is responsible for computing a collision-free 3D path using the produced Octomap. With respect to the planner used in the first competition, the A* algorithm showed a remarkable improvement in the performance: the possibility of having access to a global Octomap and of planning online allows the planner to quickly find the shortest path, even in cluttered environments.

Algorithm 2 Local planner.

Input *goal*
Output *path*

while *goal* is not reached **do**
  obst = GetObstacles()
  if IsValid($path_{prev}$, *obst*) **then**
    $path = path_{prev}$
    return *path*
  else
    if *goal* is in *obst* **then**
      while *goal* is not in *obst* **do**
        goal = PickRandTarget(*goal*)
      end while
    end if
    start = GetDronePos()
    path = ApplyAstar(*goal*, *start*, *obst*)
    if not *path* **then**
      ShrinkRadius()
      continue
    else
      return *path*
    end if
  end if
  $path_{prev} = path$
end while

The planner pseudocode is presented in Algorithm 2. The A* algorithm, using the *ApplyAstar* function, computes a path from the starting point, i.e., the drone current position in the *map* reference frame (retrieved through the *GetDronePos* function), to the next point of the global planner path. The most recent obstacles configuration in the form

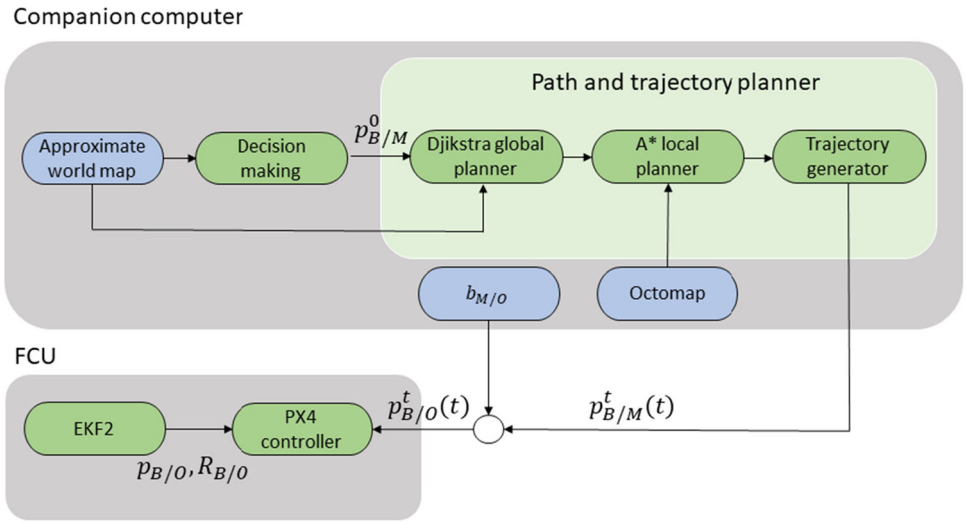

Fig. 13 Planning architecture

of an Octomap, retrieved through the *GetObstacles* function, is used by the planner. Note that a 3D implementation of A* is employed. It expands, for each node, the 26 contiguous cells in the 3D space. At each planner execution, the path is checked against newly emergent obstacles. It is worth mentioning that collision checking is conducted inflating the robot dimensions by a safety radius (*IsValid* function). If the previously computed path is still valid, it is further processed and sent to the drone controllers (as presented in the following of this section). Otherwise, a re-planning procedure is started. In addition to this, if the goal is one of the invalid points, a new target point, in the neighbourhood of the previous one, is selected (*PickRandTarget* function). Finally, in the case of difficulties in computing an admissible path, the algorithm tries to decrease the safety radius up to a saturation level through the *ShrinkRadius* function.

The resulting path from the local planner should be, then, turned into a smooth trajectory, $p^t_{B/O}(t)$, before sending it to the drone controller. This is done by computing the optimal trajectory passing through the points belonging to the local path, between the current drone position and the next point in the global path. The starting and ending points of the trajectory are constrained to be both in hover, i.e., with zero velocity. Considering the drone as a simple point-mass system, the time-optimal trajectory can be computed in closed form, and its result is a bang-bang acceleration trajectory [30]. We additionally impose a constraint on the maximum velocity, resulting in a trapezoidal velocity profile (refer to [45] for trajectory equations). The dynamics of the system is neglected since the system is limited in the attainable velocities and acceleration by the perception pipeline constraints. However, for further trajectory models please refer to [19].

It is worth noting that the generated trajectory is written in the *map* frame, while the UAV state estimate is in the *odom* frame. Consequently, the trajectory is transformed into the correct frame by exploiting the state of the localization filter $b_{O/M}$.

Note also that the yaw angle is commanded to have the first body axis, at each instant in time, oriented as the velocity vector. This is done for coping with the possibility of newly emergent obstacles during drone motion.

If during the execution of the trajectory the planner finds that the related path is no longer valid, the drone stops in its current position. Consequently, a collision-free path is computed in the updated situation. Thus, the controller will start tracking the newly produced trajectory.

Finally, the generated position set-points are sent to the UAV FCU via *mavros*. The flight controller is the one available in the PX4 firmware. A standard cascaded control architecture, in which position and attitude controllers are both P-PID, is implemented. For more details please refer to [5].

For what concerns the landing procedure, an additional controller is used. Once the landing pad position indicated on the approximate world map has been reached, a visual feedback is employed. Refer to Fig. 14 for the block diagram of the adopted landing procedure. The initial position setpoint $p^t_{B/O}$ is modified with the output of a PI controller p_{cam}. This controller takes as input the error between the drone positioning and the center of the marker, e_{cam}, information provided by the camera exploiting the ArUco tracker already employed for navigation and decision making. Lastly, $F(s)$, which represents the complementary sensitivity of the position control loop, will drive the drone to the desired position p^t_v. This control law is applied first for alignment at the drone

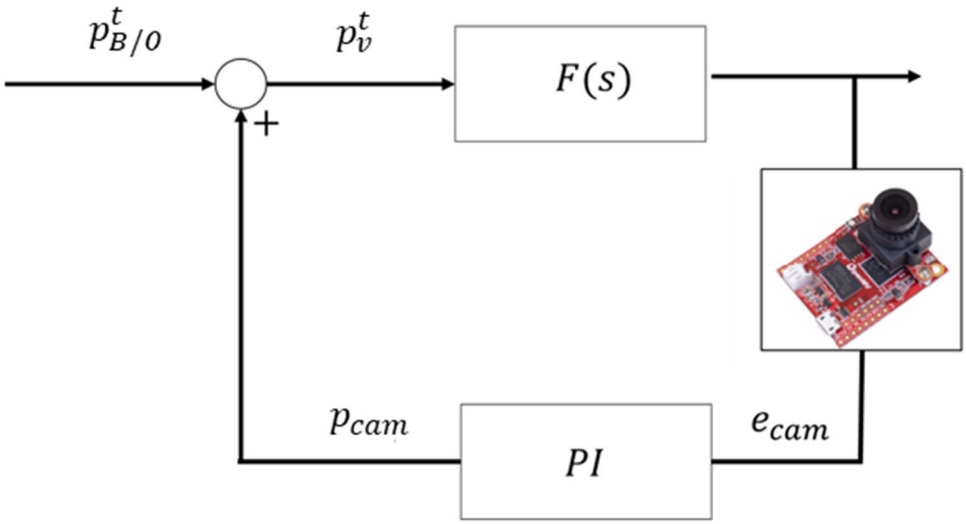

Fig. 14 Landing controller architecture

flight altitude and then tracking is also employed during descent.

5 Simulation and Experimental Setup

For simulating the overall solution and to test the compatibility of the different modules and codes, the PX4 Software In The Loop (SITL) simulation framework has been employed. Gazebo[5] has been selected as simulator, thanks to its easy and efficient interface with ROS. The simulated drone is equipped with all the sensors available in the real platform, namely a downward-looking monocular camera, a range finder altimeter and a stereo camera for visual odometry computation and mapping. On the simulator, other plugins have been integrated for reproducing the moving ground robots and the fixed camera on the field. All the phases of the competition have been simulated, a snapshot of which is available in Fig. 15. The simulated environment is a reproduction of the real competition field in Fig. 1 and of the approximate world map shown in Fig. 6.

For what concerns experimental tests, they are carried out inside the Flying Arena for Rotorcraft Technologies (FlyART) of Politecnico di Milano (Fig. 16).

FlyART is a 12 m × 6 m × 4 m indoor facility equipped with a Motion Capture system (Mocap) composed by 12 cameras, which will be used as ground truth in the following. A ground control station, connected to the same network of the drone, is also used for receiving the images of the ground robot and for sending the landing sequence to the drone. Four ArUco landing pads have been placed at known positions in the facility. A moving ground robot has been employed for performing experimental tests related to exploration and tracking. Finally, some cardboard obstacles have been put in place for testing both planning and mapping performance.

[5] https://gazebosim.org/

6 Results

In this section, first the selection of the visual odometry algorithm is discussed through results coming from a preliminary test. Simulation results about a full competition mission are then analyzed. Particular attention will be paid to some critical problems which arise on long duration flights. Finally, representative results obtained during a laboratory flight test campaign are shown.

6.1 Visual Odometry Comparison

An experiment was specifically conducted for comparing the stereo visual odometry algorithms presented in Section 3. In particular, the drone performs a circular trajectory at constant altitude, while keeping fixed yaw. The Euclidean norm of the difference between the visual odometry estimate and the ground truth position is taken as error metric. It is worth noting that the image undistortion and stereo rectification are performed on the Jetson TX2 Graphics Processing Unit (GPU) by the ZED SDK. The results are graphically shown in Fig. 17.

The mean absolute and relative error with respect to the travelled distance are reported in Table 1.

All the algorithms show good results in the proposed experiment, with the best performance obtained by the ZED-VO algorithm. Considering the available hardware, i.e.,

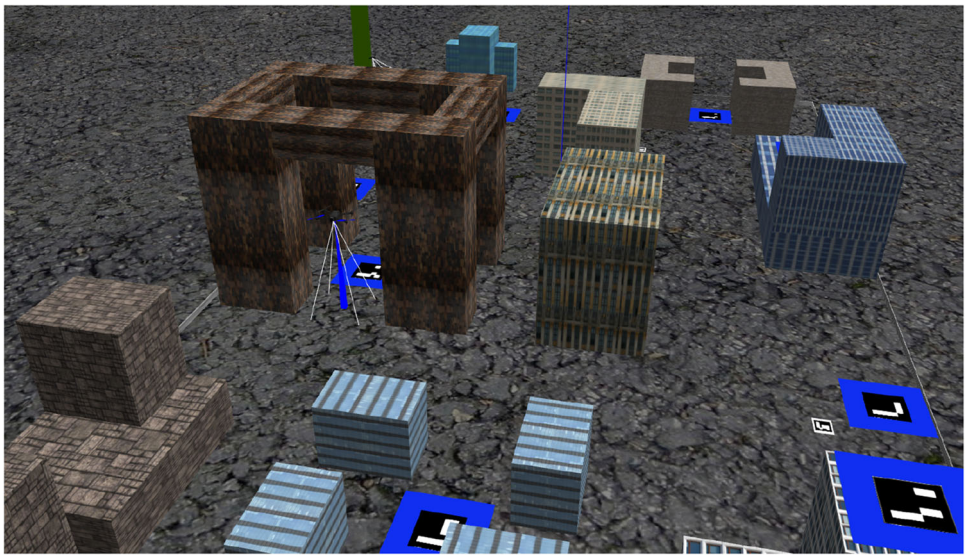

Fig. 15 A snapshot from the simulator

with GPU acceleration, this algorithm outperforms other open-source algorithms with CPU implementations. Similar results, using ground vehicles, can be also found in [22].

6.2 Simulation Results

The video of the complete simulation is available online[6]. In simulation, particular attention has been spent for dealing with aspects related to the localization. This because, due to the reduced dimensions of the laboratory flight test area compared to the competition field, simulation is the only way to test factors coming into play in the long-run. In addition to this, we focus on the z-axis, which has been highlighted as one of the most critical factors in the performance of the UAV in the first competition. Consider Fig. 18, in which the results of the localization are shown. The odometry has been deliberately simulated with high noise values in order to test the localization robustness to drift. Nevertheless, the localization is able to recover the "true" altitude, even rejecting measurements corresponding to the presence of obstacles under the drone. Note that the sum of the bias $b_{O/M}$, state of the filter presented in Section 4.1, and the position estimated by the EKF2 $p_{B/O}$ is represented as the output of the localization process.

As already mentioned in Section 3, the localization output is used for retrieving the transformation between the *map* and *odom* reference frames. This information is then used for transforming the setpoints in the *odom* reference frame. The result of this procedure is shown in Fig. 19. The setpoint is clearly changing for compensating the drift of the altitude estimate. The drone is, thus, able to keep the altitude constant, solving one of the problems which arose in the first competition.

6.3 Experimental Results

The video of the experimental test is available online[7]. The experiment can be split up in its main parts according to the decision-making algorithm. After takeoff, the UAV conducted an exploration phase of the laboratory environment, looking for the *intruder* robot. After having found it with the forward-looking camera, the tracking phase begun. The UAV is driven towards the robot and it starts taking pictures of it. After that, we can see the drone holding the position while we were selecting the landing sequence from the ground control station. Finally, the drone reached and landed on each of the selected pads according to the given sequence.

For what concerns the navigation performance, the East and North localization position components are shown in Figs. 20 and 21 respectively. In these plots, the localization is compared with the visual odometry output, which is in turn affected by drift. In order to be in the same reference frame of the localization position, the odometry curve has been shifted by an offset, namely the drone initial position in the *map* frame, which corresponds to the position of one of the ArUco landing pads available in the laboratory. At the same time, taking off from one of the ArUco markers, the localization curve shows a sharp transient toward the value of the ground truth.

For the Up direction, the altitude estimation results are plotted in Fig. 22. The localization algorithm shows good performance, even if the visual odometry already started a remarkable drift in the short time frame of the experiment.

[6] Visit https://www.youtube.com/watch?v=N7o6_CEzCn4

[7] Visit https://www.youtube.com/watch?v=lGsVxvEKR4A

Fig. 16 The FlyART of Politecnico di Milano

In Fig. 23 we can see the Octomap, output of the RTAB-Map SLAM algorithm, generated by the drone during the mission. It resulted in a good representation of the laboratory environment (see Fig. 23), in which the cardboard obstacles and arena borders are indeed correctly mapped.

As far as the performance of the vision-based landing are concerned, all the four landings can be considered valid according to the competition rules. The ground truth path of the drone during the experiment with the associated 4 landings are shown in Fig. 24.

7 Technical Challenges and Future Works

During the solution development, many technical challenges were faced. For example, the choice of the range finder (utilized for altimeter purposes) was crucial. On the competition floor, which was covered by a carpet, many commercial solutions (Terabee Teraranger 3 m, VL53L1X) showed some limitations: they were either unable to return a measurement or significantly inaccurate, especially with the increase in the flight altitude. The selected Lidar-Lite v3, instead, has shown remarkable accuracy and robustness to changes in flight conditions.

From the software point of view, the range finder measurements' rejection mechanism was a critical aspect in the design of the solution. Clearly, the range finder measurements were not fused directly in the EKF2 to avoid making fast upward/downward moves when flying over an obstacle.

The first designed solution consisted of a finite-state machine (FSM), whose switching logic was based on the relative difference between consecutive measurements. Knowing the FSM's state and the lowest obstacle's height, the idea was

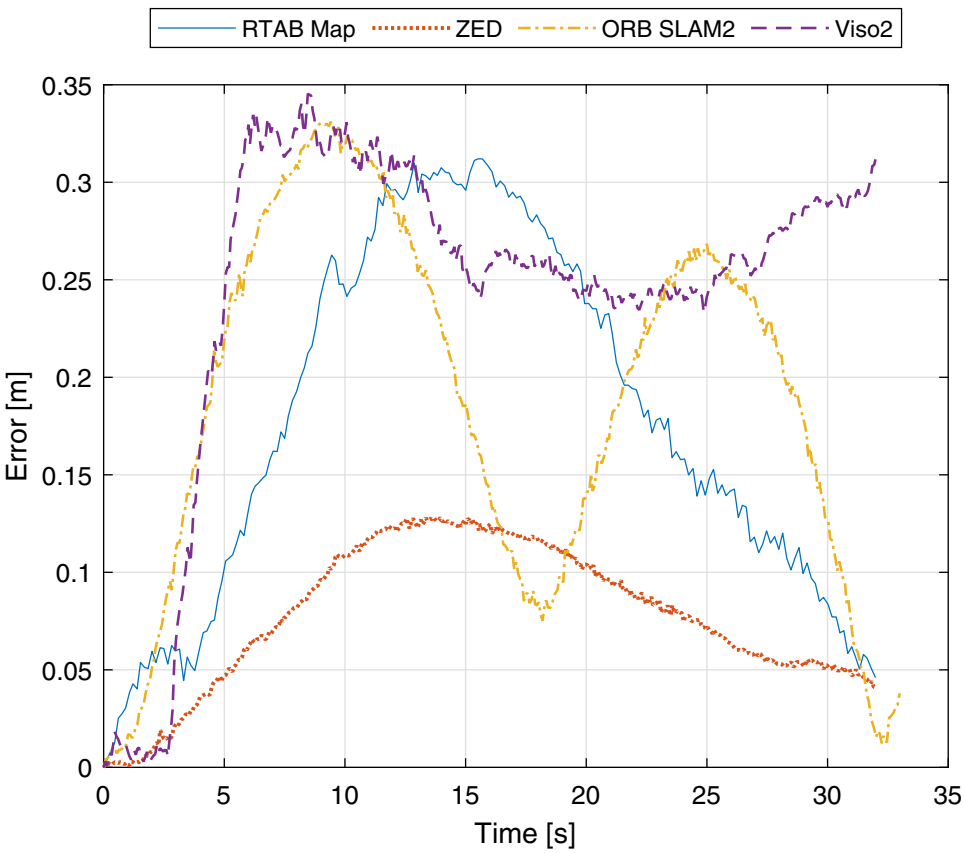

Fig. 17 Comparison among stereo visual odometry algorithms

Table 1 Mean absolute and relative error of analyzed stereo visual odometry algorithms

Algorithm	Mean error [m]	Relative error [%]
RTAB-Map	0.175	2.70
ZED-VO	0.078	1.21
LibViso2	0.244	3.77
ORB-SLAM2	0.184	2.84

to correct the flight altitude, even during passages over obstacles. However, this approach showed a strong sensitivity to the quality of the measured altitude. For example, when flying over an obstacle, the laser altimeter sometimes measured a smooth transition instead of an abrupt change, as required by the FSM.

To solve this problem, we started using a statistical outlier approach for rejecting the outliers, i.e., measurements corresponding to the sensing of a building below the drone (as presented in Section 4). The proposed procedure only presents a limitation regarding the time spent hovering over an obstacle. In fact, after rejecting many range measurements with a consequent increase of the estimation error covariance matrix components, a measurement could satisfy the NEES check. If this condition is met, the estimated z-position component would decrease, and the drone altitude would increase accordingly.

In this context, note that the altimeter measurements and the corresponding outlier rejection mechanism can be applied at the localization level (as we did in Section 4) or right before fusing the stereo camera information with the IMU data in the EKF2.

Despite the achievements of the second competition (for competition results, refer to our previous paper [39]), some criticalities were still present.

It is worth noting that, during the competition, outside the field, spectators could move around to see the experimental platforms during the tasks execution. This aspect impacted the accuracy of the visual odometry employed for state estimation.

In addition to this, on one side of the field, a uniform black textureless wall was present. In that area, the lack of features made the visual odometry output quality very low. Overall, the flight quality was greatly affected compared to the laboratory environment. This problem was particularly evident during landings, where inaccurate position estimation caused some of them to be invalid in contrast to what was seen in the laboratory (see Fig. 24).

The designed vision-based controller allows the drone to align accurately with the underlying ArUCo marker, but during the descent phase, the controller could not compensate for the drone motion due to state estimation inaccuracies. The performance is even worsened because the ArUCo marker pose estimate gets interrupted when the marker leaves the camera field of view during drone descent.

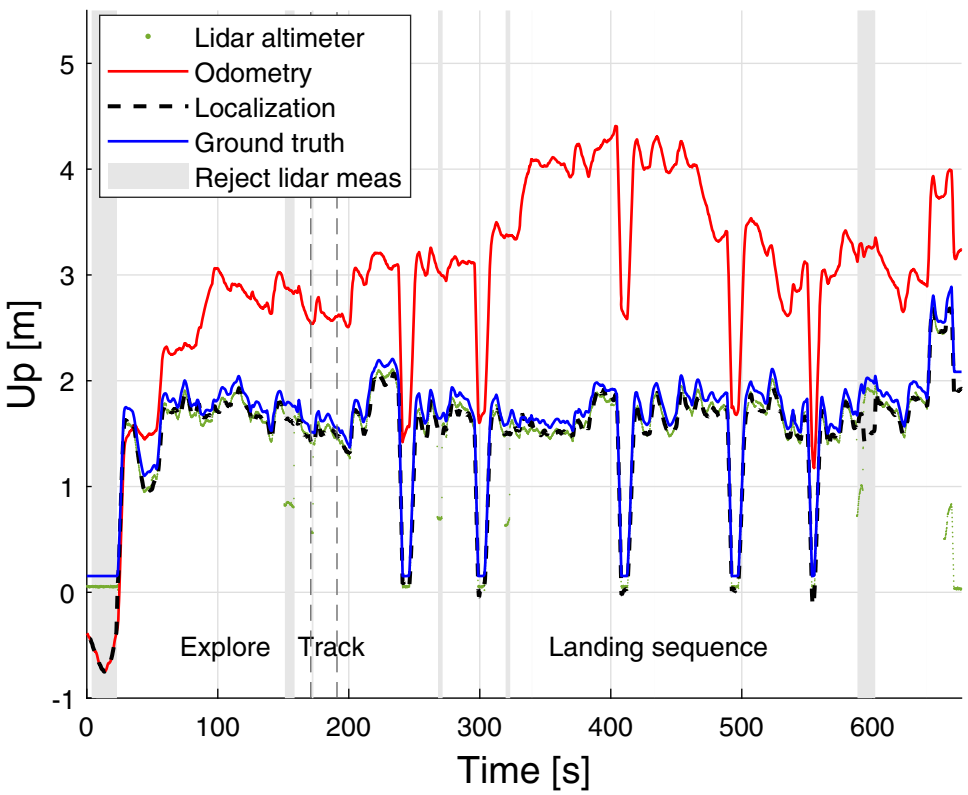

Fig. 18 Comparison between odometry and localization (Up axis) in simulation

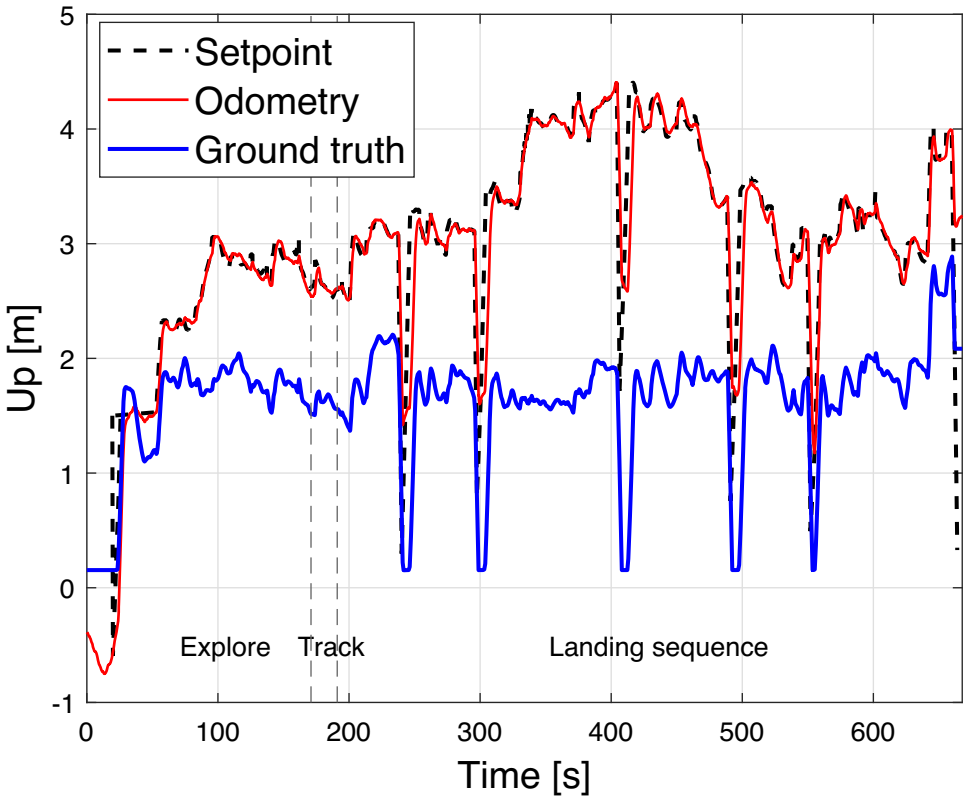

Fig. 19 Setpoint (Up axis) sent to the controller. Odometry and ground truth also provided

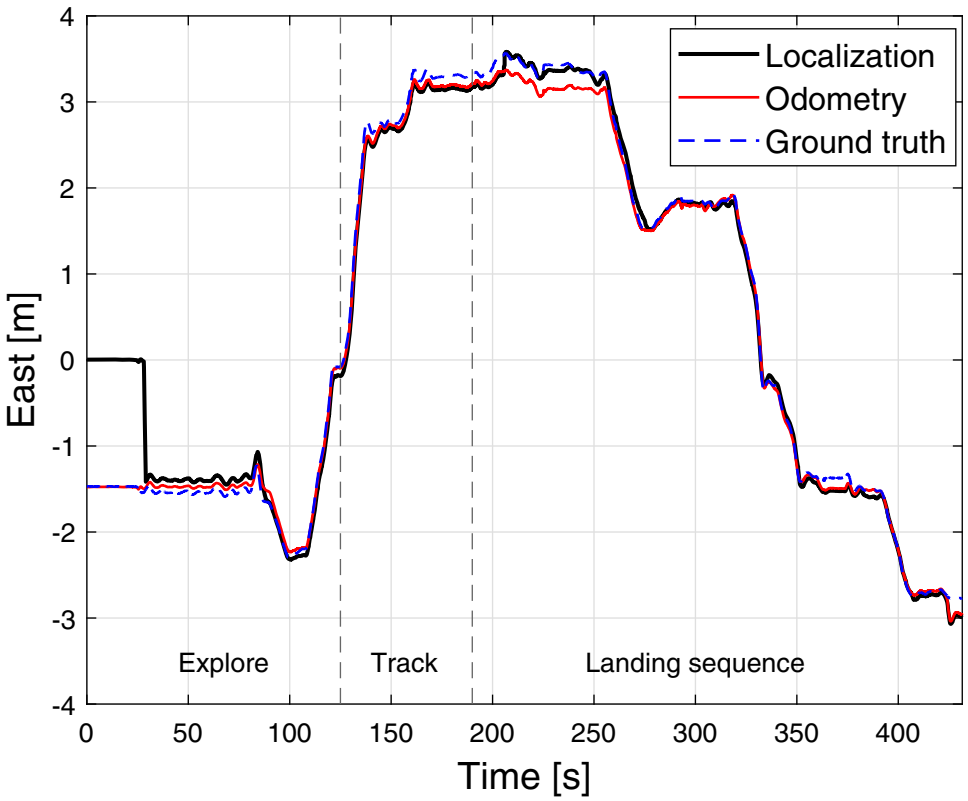

Fig. 20 Comparison between odometry and localization (East axis)

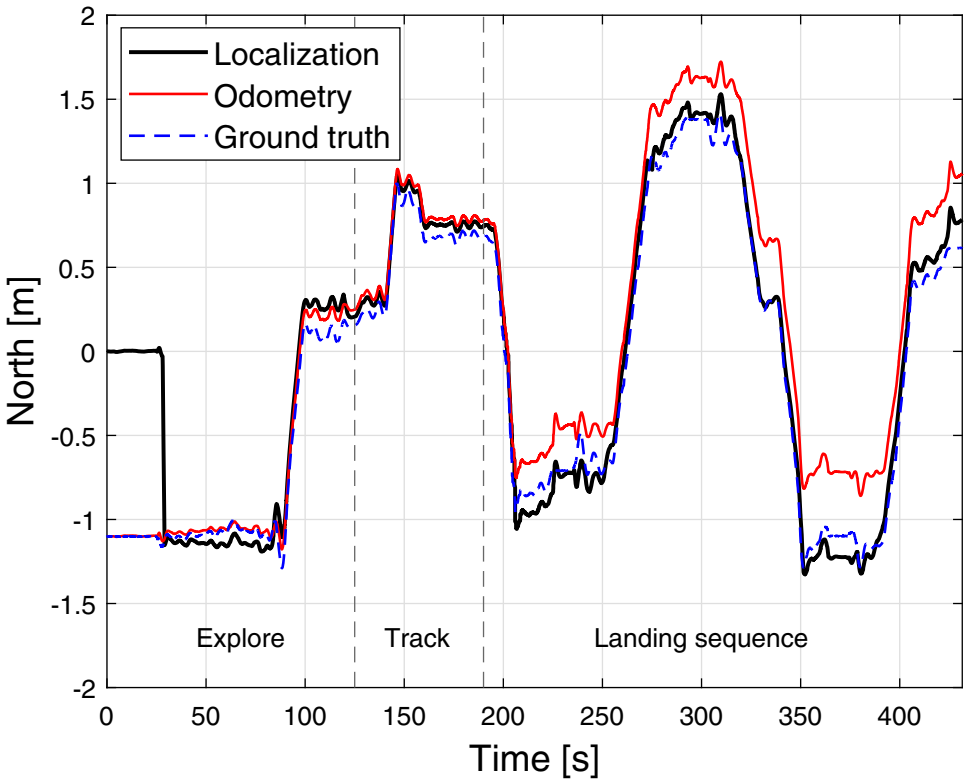

Fig. 21 Comparison between odometry and localization (North axis)

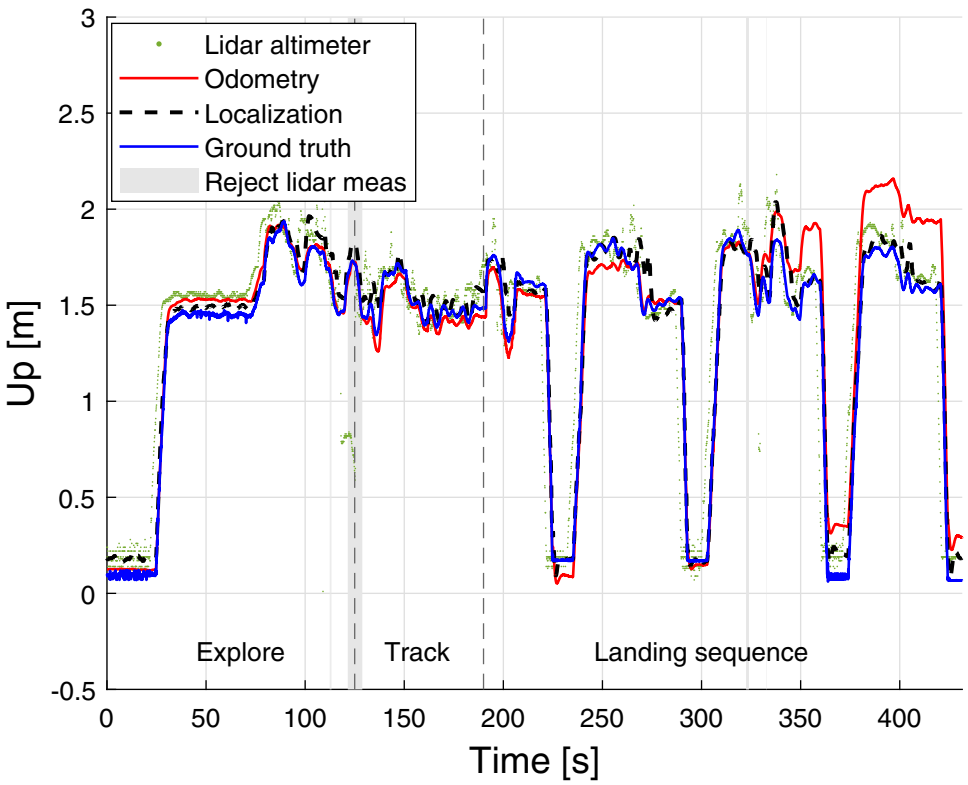

Fig. 22 Comparison between odometry and localization (Up axis). Added patches in which the range finder measurements are rejected

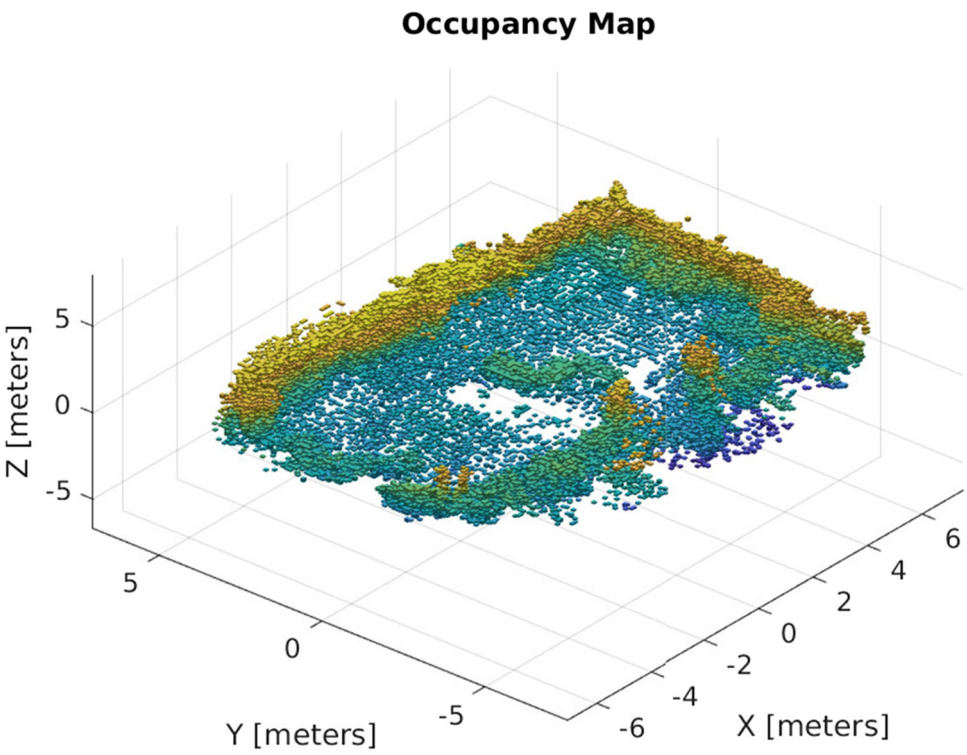

Fig. 23 Octomap obtained during the experiment

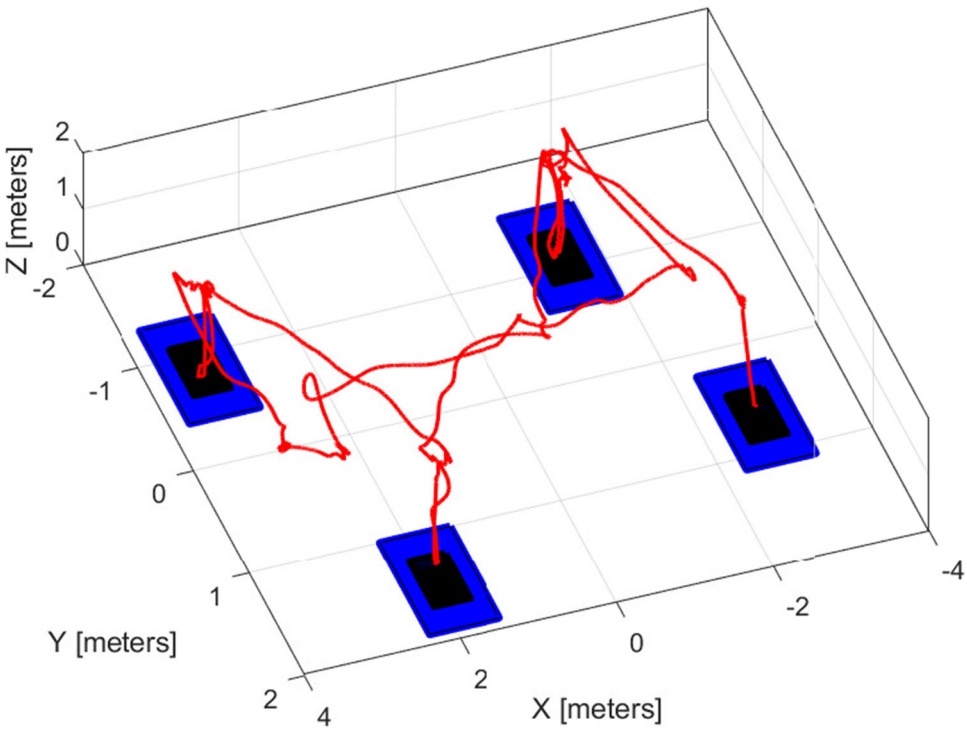

Fig. 24 Ground truth path and landings

In view of the third and last LDC competition, future works could regard the improvement of visual odometry performance, changing the stereo camera orientation to a downward-looking configuration. This change will help solve the lack of features and the presence of moving people and/or objects. On the other hand, the resulting architecture will also need another camera for mapping and planning purposes.

Alternatively, optical flow sensors could also be tested for improving state estimation. These sensors could complement the forward-camera visual odometry in the aforementioned situations. This solution has some significant advantages: optical flow sensors are cheaper, lighter and require less energy and computations with respect to a possible additional stereo camera.

To reduce the impact of state estimation on the landing accuracy, the control scheme of Fig. 14 could be re-designed for sending velocity commands instead of position ones.

Moreover, additional work could be devoted to the planning module. This module can be simplified by removing the global planner acting on the world approximated map and by keeping the A* computing the shortest collision-free path from the drone current position to the end goal. This change will increase the robustness of the system and the generality of the solution.

Finally, an important aspect to be tackled is the reduction of the computational power employed. The solution employed up to 90% of the available CPU. The software which is mainly responsible for such a high load on the processor is the ArUco tracker. It continuously searches for markers, both landing pads and moving robots, over three different images from the downward-looking, forward-looking and fixed PTZ cameras. A possible solution regards the development of a customized version of the tracker, replacing the OpenCV implementation and possibly exploiting the available GPU resources.

8 Conclusions

In this paper, the solutions proposed by Politecnico di Milano team at the first two editions of the *Leonardo Drone Contest* are presented. First, the aim of the competitions, their rules and objectives have been described. Successively, the authors presented the two platforms specifically designed for the contests, with particular focus on the equipped sensors and software architectures. While for the first edition the focus was on the lessons learnt during the competition, the solution developed for the second edition was discussed in details. The modular architecture, comprising of navigation, planning and decision-making modules was analyzed and the performance obtained by the autonomous system were assessed through both simulation and experimental activities in a laboratory environment. Finally, some technical challenges which arose during the competition and planned future activities were presented.

Acknowledgements The authors thank the spin-off company of Politecnico di Milano ANT-X for the collaboration in the development of the hardware platforms. The authors thank prof. Matteo Matteucci of Politecnico di Milano for the help in the conceptualization of the solution.

Author Contributions All authors contributed to the research, development, and testing of the proposed systems. Gabriele Roggi is the author of the proposed software architectures and contributed to the development of the hardware platforms. Salvatore Meraglia contributed to the preliminary design of the solution and to the experimental activities. Marco Lovera is the head of the Aerospace Systems and Control Laboratory of Politecnico di Milano, and he provided us with the necessary guidance, funding and final proofreading. The first draft of the manuscript was written by Gabriele Roggi, and all authors commented on previous versions of the submitted manuscript.

Funding Open access funding provided by Politecnico di Milano within the CRUI-CARE Agreement. The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Data Availability The manuscript has no associated data.

Declarations

Ethics Approval and Consent to Participate All applicable institutional and national guidelines were followed.

Consent for Publication Informed consent was obtained from all the co-authors of this publication.

Competing Interest The authors declare that they have no conflict of interest.

References

1. Alphapilot AI Drone Innovation Challenge. https://www.lockheedmartin.com/en-us/news/events/ai-innovation-challenge.html. Accessed 28 June 2022
2. ANT-X website. https://antx.it/. Accessed 28 June 2022
3. Leonardo Drone Contest autonomous drone competition. https://www.leonardo.com/it/innovation-technology/open-innovation/drone-contest. Accessed 17 Jan 2022

4. Mohamed Bin Zayed International Robotics Challenge (MBZIRC). https://www.mbzirc.com/. Accessed 17 Jan 2022
5. PX4 documentation. https://docs.px4.io/v1.12/en/. Accessed 06 May 2022
6. Achtelik, M., Bachrach, A., He, R., Prentice, S., Roy, N.: Stereo vision and laser odometry for autonomous helicopters in GPS-denied indoor environments. Unmanned Systems Technology XI **7332**, 733219 (2009)
7. Basilico, N.: Recent trends in robotic patrolling. Current Robotics Reports pp. 1–12 (2022)
8. Beul, M., Bultmann, S., Rochow, A., Rosu, R.A., Schleich, D., Splietker, M., Behnke, S.: Visually guided balloon popping with an autonomous MAV at MBZIRC 2020. IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) pp. 34–41 (2020)
9. Bircher, A., Kamel, M., Alexis, K., Oleynikova, H., Siegwart, R.: Receding horizon "next-best-view" planner for 3D exploration. IEEE International Conference on Robotics and Automation (ICRA) pp. 1462–1468 (2016)
10. Blösch, M., Weiss, S., Scaramuzza, D., Siegwart, R.: Vision based MAV navigation in unknown and unstructured environments. 2010 IEEE International Conference on Robotics and Automation pp. 21–28 (2010)
11. De Wagter, C., Paredes-Vallés, F., Sheth, N., de Croon, G.: The artificial intelligence behind the winning entry to the 2019 AI robotic racing competition. arXiv preprint arXiv:2109.14985 (2021)
12. De Wagter, C., Paredes-Vallés, F., Sheth, N., de Croon, G.: Learning fast in autonomous drone racing. Nature Machine Intelligence **3**(10), 923–923 (2021)
13. Dias, J., Althoefer, K., Lima, P.U.: Robot competitions: What did we learn? IEEE Robotics & Automation Magazine **23**(1), 16–18 (2016)
14. Doitsidis, L., Weiss, S., Renzaglia, A., Achtelik, M., Kosmatopoulos, E., Siegwart, R., Scaramuzza, D.: Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision. Autonomous Robots **33**, 173–188 (2012)
15. Engel, J., Sturm, J., Cremers, D.: Accurate figure flying with a quadrocopter using onboard visual and inertial sensing. Proceedings of the Workshop on Visual Control of Mobile Robots (ViCoMoR) at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2012)
16. Engel, J., Sturm, J., Cremers, D.: Scale-aware navigation of a low-cost quadrocopter with a monocular camera. Robotics and Autonomous Systems **62**(11), 1646–1656 (2014)
17. Faessler, M., Fontana, F., Forster, C., Mueggler, E., Pizzoli, M., Scaramuzza, D.: Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle. J. Field Rob. **33**(4), 431–450 (2016)
18. Foehn, P., Brescianini, D., Kaufmann, E., Cieslewski, T., Gehrig, M., Muglikar, M., Scaramuzza, D.: Alphapilot: Autonomous drone racing. Autonomous Robots pp. 1–14 (2021)
19. Foehn, P., Romero, A., Scaramuzza, D.: Time-optimal planning for quadrotor waypoint flight. Science Robotics **6**(56), eabh1221 (2021)
20. Forster, C., Zhang, Z., Gassner, M., Werlberger, M., Scaramuzza, D.: SVO: Semidirect visual odometry for monocular and multicamera systems. IEEE Transactions on Robotics **33**(2), 249–265 (2016)
21. Geiger, A., Ziegler, J., Stiller, C.: Stereoscan: Dense 3D reconstruction in real-time. IEEE Intelligent Vehicles Symposium (IV) pp. 963–968 (2011). https://doi.org/10.1109/IVS.2011.5940405
22. Giubilato, R., Chiodini, S., Pertile, M., Debei, S.: An evaluation of ROS-compatible stereo visual SLAM methods on a nVidia Jetson TX2. Measurement (2019). https://doi.org/10.1016/j.measurement.2019.03.038
23. González-Banos, H.: A randomized art-gallery algorithm for sensor placement. Proceedings of the seventeenth annual symposium on Computational geometry pp. 232–240 (2001)
24. Hong, W., Zhou, C., Tian, Y.: Robust Monte Carlo Localization for humanoid soccer robot. IEEE/ASME International Conference on Advanced Intelligent Mechatronics pp. 934–939 (2009). https://doi.org/10.1109/AIM.2009.5229889
25. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots (2013)
26. Hoshino, S., Ishiwata, T., Ueda, R.: Optimal patrolling methodology of mobile robot for unknown visitors. Advanced Robotics **30**(16), 1072–1085 (2016)
27. Hoshino, S., Ugajin, S., Ishiwata, T.: Patrolling robot based on bayesian learning for multiple intruders. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) pp. 603–609 (2015)
28. Labbé, M., Michaud, F.: Appearance-based loop closure detection for online large-scale and long-term operation. IEEE Transactions on Robotics **29**(3), 734–745 (2013). https://doi.org/10.1109/TRO.2013.2242375
29. Labbé, M., Michaud, F.: RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. Journal of Field Robotics **36**(2), 416–446 (2019)
30. LaValle, S.M.: Planning algorithms. Cambridge University Press, Cambridge, U.K. (2006)
31. Lenz, C., Schwarz, M., Rochow, A., Razlaw, J., Periyasamy, A.S., Schreiber, M., Behnke, S.: Autonomous wall building with a UGV-UAV team at MBZIRC 2020. IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) pp. 189–196 (2020)
32. 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. Journal of Field Robotics **35**(1), 101–120 (2018)
33. Moon, H., Martínez-Carranza, J., Cieslewski, T., Faessler, M., Falanga, D., Simovic, A., Scaramuzza, D., Li, S., Ozo, M.M.O.I., de Wagter, C., de Croon, G.C., Hwang, S., Jung, S., Shim, H., Kim, H., Park, M., Au, T.C., Kim, S.J.: Challenges and implemented technologies used in autonomous drone racing. Intelligent Service Robotics **12**, 137–148 (2019)
34. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015)
35. Nikolic, J., Burri, M., Rehder, J., Leutenegger, S., Huerzeler, C., Siegwart, R.: A UAV system for inspection of industrial facilities. 2013 IEEE Aerospace Conference pp. 1–8 (2013)
36. Oleynikova, H., Burri, M., Taylor, Z., Nieto, J., Siegwart, R., Galceran, E.: Continuous-time trajectory optimization for online UAV replanning. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) pp. 5332–5339 (2016). https://doi.org/10.1109/IROS.2016.7759784
37. 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. Journal of Field Robotics **37**(4), 642–666 (2020)
38. Rezende, A., Miranda, V.R., Rezeck, P.A., Azpúrua, H., Santos, E.R., Gonçalves, V.M., Macharet, D.G., Freitas, G.M.: An integrated solution for an autonomous drone racing in indoor environments. Intel. Serv. **14**(5), 641–661 (2021)
39. Roggi, G., Meraglia, S., Lovera, M.: Leonardo Drone Contest 2021: Politecnico di Milano team architecture. International Conference on Unmanned Aircraft Systems (ICUAS) pp. 191–196 (2022)
40. Sampedro, C., Bavle, H., Rodríguez-Ramos, A., Carrio, A., Fernández, R.A.S., Sanchez-Lopez, J.L., Campoy, P.: A fully-autonomous aerial robotic solution for the 2016 international micro air vehicle

competition. 2017 International conference on unmanned aircraft systems (ICUAS) pp. 989–998 (2017)
41. Santamaria-Navarro, A., Loianno, G., Solà, J., Kumar, V., Andrade-Cetto, J.: Autonomous navigation of micro aerial vehicles using high-rate and low-cost sensors. Autonomous Robots **42**(6), 1263–1280 (2018)
42. Scaramuzza, D., Achtelik, M., Doitsidis, L., Friedrich, F., Kosmatopoulos, E., Martinelli, A., Achtelik, M., Chli, M., Chatzichristofis, S., Kneip, L., et al.: Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in GPS-denied environments. IEEE Robotics & Automation Magazine **21**(3), 26–40 (2014)
43. Schedl, D.C., Kurmi, I., Bimber, O.: An autonomous drone for search and rescue in forests using airborne optical sectioning. Science Robotics **6**(55) (2021)
44. Schmid, K., Lutz, P., Tomić, T., Mair, E., Hirschmüller, H.: Autonomous vision-based micro air vehicle for indoor and outdoor navigation. Journal of Field Robotics **31**(4), 537–570 (2014)
45. Sciavicco, L., Siciliano, B.: Modelling and control of robot manipulators. Springer Science & Business Media, Heidelberg (2001)
46. Semsch, E., Jakob, M., Pavlicek, D., Pechoucek, M.: Autonomous UAV surveillance in complex urban environments. IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology **2**, 82–85 (2009)
47. Sousa, A., Costa, P., Moreira, A., Carvalho, A.: Self localization of an autonomous robot: using an EKF to merge odometry and vision based landmarks. IEEE Conference on Emerging Technologies and Factory Automation **1**, 7–233 (2005). https://doi.org/10.1109/ETFA.2005.1612524
48. Stronger, D., Stone, P.: A comparison of two approaches for vision and self-localization on a mobile robot. IEEE International Conference on Robotics and Automation pp. 3915–3920 (2007). https://doi.org/10.1109/ROBOT.2007.364079
49. Ulrich, I., Borenstein, J.: VFH*: Local obstacle avoidance with look-ahead verification. IEEE International Conference on Robotics and Automation (ICRA) **3**, 2505–2511 (2000)
50. Vanneste, S., Bellekens, B., Weyn, M.: 3DVFH+: Real-time three-dimensional obstacle avoidance using an octomap. Proceedings of the 1st International Workshop on Model-Driven Robot Software Engineering (1319), 91–102 (2014)
51. Vrba, M., Stasinchuk, Y., Báča, T., Spurnỳ, V., Petrlík, M., Heřt, D., Žaitlík, D., Saska, M.: Autonomous capture of agile flying objects using UAVs: The MBZIRC 2020 challenge. Robotics and Autonomous Systems **149**, 103970 (2022)
52. Walter, V., Spurný, V., Petrlík, M., Báča, T., Žaitlík, D., Saska, M.: Extinguishing of ground fires by fully autonomous UAVs motivated by the MBZIRC 2020 competition. International Conference on Unmanned Aircraft Systems (ICUAS) pp. 787–793 (2021). 10.1109/ICUAS51884.2021.9476723
53. Zhou, B., Gao, F., Wang, L., Liu, C., Shen, S.: Robust and efficient quadrotor trajectory generation for fast autonomous flight. IEEE Robotics and Automation Letters **4**(4), 3529–3536 (2019). https://doi.org/10.1109/LRA.2019.2927938

Gabriele Roggi graduated in Aeronautical Engineering from Politecnico di Milano, Milano, Italy, in 2019, and received the Ph.D. degree in Aerospace Engineering from Politecnico di Milano in 2023. He is currently Research Fellow with the Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano. His research activity involves the design, implementation and experimental validation of guidance, navigation and control algorithms for multirotor UAVs.

Salvatore Meraglia graduated in Space Engineering from Politecnico di Milano, Milano, Italy, in 2019, and received the Ph.D. degree in Aerospace Engineering from Politecnico di Milano in 2023. He is currently Research Fellow with the Dipartimento di Scienze e Tecnologie Aerospaziali, Politecnico di Milano. His research encompasses control theoretic investigations and experimental validation of a wide variety of aerospace systems, including space vehicles and multirotor UAVs.

Marco Lovera received the M.Sc. in Electronics Engineering at Politecnico di Milano, Milano, Italy, 1993 and the Ph.D. degree in Computer Science and Control Engineering at Politecnico di Milano, Milano, Italy, 1998. After a one-year period in industry he joined the Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, in 1999. Since 2015, he is with the Dipartimento di Scienze e Tecnologie Aerospaziali of the Politecnico di Milano, where he leads the Aerospace Systems and Control Laboratory (ASCL).