Journal of Intelligent & Robotic Systems (2024) 110:10
https://doi.org/10.1007/s10846-023-02042-7

REGULAR PAPER

Towards Autonomous Firefighting UAVs: Online Planners for Obstacle Avoidance and Payload Delivery

Michael Mugnai[1] · Massimo Teppati Losè[1] · Massimo Satler[1] · Carlo Alberto Avizzano[1]

Received: 30 November 2022 / Accepted: 16 December 2023

Abstract
Drone technology is advancing rapidly and represents significant benefits during firefighting operations. This paper presents a novel approach for autonomous firefighting missions for Unmanned Aerial Vehicles (UAVs). The proposed UAV framework consists of a local planner module that finds an obstacle-free path to guide the vehicle toward a target zone. After detecting the target point, the UAV plans an optimal trajectory to perform a precision ballistic launch of an extinguishing ball, exploiting its kinematics. The generated trajectory minimises the overall traversal time and the final state error while respecting UAV dynamic limits. The performance of the proposed system is evaluated both in simulations and real tests with randomly positioned obstacles and target locations. The proposed framework has been employed in the 2022 UAV Competition of the International Conference on Unmanned Aircraft Systems (ICUAS), where it successfully completed the mission in several runs of increasing difficulty, both in simulation and in real scenarios, achieving third place overall. A video attachment to this paper is available on the website https://www.youtube.com/watch?v=_hdxX2xXkVQ.

Keywords Obstacle avoidance · Trajectory planning · Optimal path · Payload delivery · UAV · Firefighting · ICUAS 22 · Competition

1 Introduction

Robotic technology has recently been used in numerous civil applications, including search and rescue and firefighting operations [1, 2]. Several factors contribute to more than 50% of firefighter deaths, including smoke inhalation, overexertion, stress, fire explosion, or even being trapped by constrained passages [3]. Nowadays, the development of autonomous systems for emergency situations becomes a key element in reducing response time, increasing safety for people, assets and rescuers, improving fire repression's effectiveness and reducing intervention costs.

Unmanned Ground Vehicles (UGV) systems can cooperate for environment monitoring and object detection to support the work of firefighters [4]. The mobility of this type of robot represents a limitation for extinguishing fires in high-rise buildings where the fire spot location is difficult to detect and interact with from the ground. The small size and high maneuverability of Unmanned Aerial Vehicles (UAVs) allow flying autonomously in difficult-to-reach and hazardous complex environments. This is a key element in the support for fire-extinguishing operations and monitoring tasks from a raised and obstacle-free point of view. In addition, these aerial vehicles can perform extinguish operations in high-rise buildings, overcoming the constraints of the actual fire trucks and their working place limits [5].

Existing approaches in firefighting UAV focus on integrating semi-autonomous platforms into firefighting human teams [6, 7], where teleoperated aerial systems, equipped with localization and local sensing modules, simplify the pilot intervention in remote environments. Recent studies [8] propose human-drone interaction strategies based on the

Michael Mugnai and Massimo Teppati Losè contributed equally to this work

✉ Massimo Teppati Losè
massimo.teppatilose@santannapisa.it

Michael Mugnai
michael.mugnai@santannapisa.it

Massimo Satler
massimo.satler@santannapisa.it

Carlo Alberto Avizzano
carloalberto.avizzano@santannapisa.it

[1] Department of Excellence in Robotics and A.I., Scuola Superiore Sant'Anna, Perceptual Robotics Laboratory at the Institute of Mechanical Intelligence, Pisa 56100, Italy

Published online: 13 January 2024

communication between the firefighting team and the UAV through a set of gestures in order to achieve specific tasks. This innovative research opens the space for the development of aerial systems capable of completing individual firefighting tasks completely autonomously that still maintain a degree of control by the operator.

Robotics competitions encourage the development of autonomous and innovative solutions for various firefighting problems. The recent emergence of these competitions is trying to advance the state of the art about motion planning, exploration, fire detection and extinguishing action in urban firefighting scenarios by providing near to real-world testing environments and bench-marking [9]. During the Fire Challenge of the MBZIRC 2020 competition[1] participants had to autonomously detect, approach and extinguish a series of simulated and real fires in an urban building, by employing a collaborative team of 3 UAVs and one UGV. The most common approach for the task of fire extinguishing inside a building was performed by a UAV equipped with a water bag and a pump [10, 11]. Recent studies [12] investigated the use of fire extinguishing balls for building fires and wildfires and discussed the efficiency of a swarm of drones dropping the balls to optimal points to control fires.

Concerning the water-pump mechanism mounted in a UAV, the ballistic release of an extinguishing ball through an open window can bring advantages to help control the spread and get the flames to a more manageable level. Once the UAV locates where the target is, a ballistic launch is performed by identifying the launch position and driving the UAV toward it with prescribed velocities and acceleration. This allows the deployment of extinguishing balls not just in vertical descending directions from the UAV but to a certain degree, even in the forward direction, as in the example scenario in Fig. 1. The motion of the UAV must ensure a certain degree of precision in the delivery of the payload and perform safe trajectories without harming the drone. The drone's payload can be considerably reduced with respect to water-pump mechanisms, and the release mechanism can be more straightforward, e.g. using a magnet. On the other hand, the release of a ball toward the detected target can be more complex in confined spaces, especially if the extinguishing agent cannot be released from above. In such cases, the aerial vehicle must plan an aggressive manoeuvre to execute the precision ballistic launch of the ball while avoiding collision with the surroundings. Table 1 summarizes the main characteristics of other state of the art designs and implementations of unmanned aerial vehicles for autonomous firefighting missions.

This paper proposes collision avoidance and payload delivery modules for an autonomous fire drone system to extinguish high-rise building fires in complex urban scenarios. A vision-based UAV framework that can navigate and reactively avoid obstacles with limited onboard computing resources in an unknown environment is proposed and tested both in simulation and in a real scenario. Starting from no knowledge of its surroundings, the UAV has to first navigate through an unknown, obstacle-dense, environment, employing only onboard sensors and the target zone direction as information. When the area of interest has been reached, the UAV localizes the spot target and plans the optimal trajectory to follow for the precise ballistic launch of the ball to the target, minimising metrics of interest such as overall mission time, overall accelerations while observing dynamics limits such as maximum velocities, accelerations and obstacle-free space.

[1] http://www.mbzirc.com/

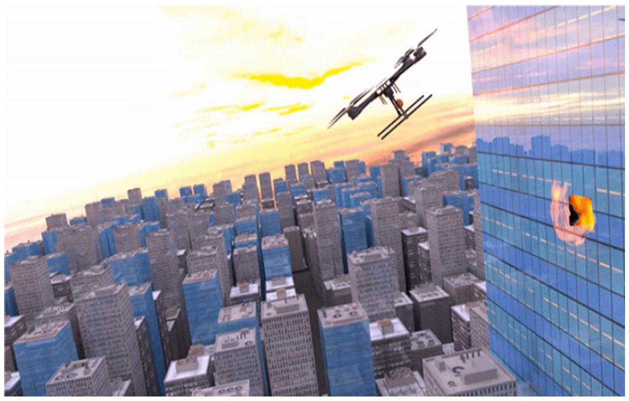


Fig. 1 The UAV is performing an aggressive trajectory to deliver the fire extinguish ball in a high-rise building. An efficient trajectory planner allows the UAV to launch autonomously the ball toward the frontal targets, releasing the payload with a simple magnetic mechanism and without colliding with obstacles

This research work was motivated by the participation of authors team, SantDrone, in the 2022 UAV Competition of International Conference on Unmanned Aircraft Systems[2] (ICUAS'22), aimed at fire fighting challenges in urban scenarios.

The remaining of the paper is organized as follows: Section 2 describes the inspected problem and gives details about the competition scenario and the UAV characteristics. Section 3 provides a brief overview of related state-of-the-art methods useful to tackle the case study.

Section 4 presents the proposed solution highlighting the two main components, i.e. the Reactive Local Planner and the Trajectory Planner and Payload delivery. The former is employed for obstacle avoidance, whereas the latter is used to generate optimal paths able to achieve ballistic launches. Finally, Section 5 shows the result starting from an in-depth study of each method in many simulation cases and then going further in real environment experiments. Section 6 summarises the overall contribution.

[2] http://www.uasconferences.com/2022_icuas/uav-competition/

Table 1 Characteristics comparison of current solutions of firefighting UAVs

	Extinguish system	Localization system	Obstacle detection system
Spurny et al. [10]	Water sprayer from inside	Visual-Inertial Odometry + GPS + 2D LiDAR	2D LiDAR + Stereo Camera
Qin et al. [13]	Water storage tank	RTK GPS	–
Ours	Extinguish ball launch	Motion capture	Stereo Camera

2 Problem Statement

The 2022 UAV Competition of the International Conference on Unmanned Aircraft Systems (ICUAS 22) focuses on pushing the limits of autonomous quadrotor platforms for firefighting missions. The challenge is divided into two phases: during the qualification phase, each competitor develops and tests its solution in a simulation environment. The five teams that complete the mission with the overall biggest score can access live trials in an indoor arena at the conference venue. The scoring considers both the overall mission time and the precision of the delivery, expressed as the minimum distance achieved between the marker and the ball during its launch.

In both phases, the arena is divided into 3 zones as shown in Fig. 2 whose bounding boxes are provided to participants. To successfully complete the mission, the UAV needs to perform the following three tasks in a row:

- Obstacle Avoidance: the UAV spawns at a random position in Zone 1 (obstacle-free) and has to find a way to cross Zone 2, which contains static obstacles, in order to reach Zone 3. There is no prior information about obstacles locations or shapes.

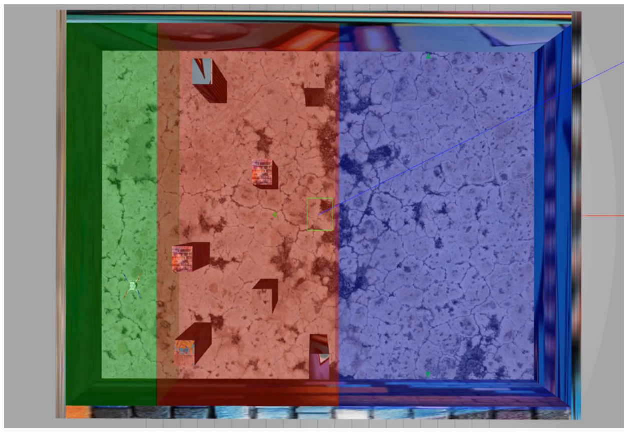


Fig. 2 Top view of the simulation scenario with the highlighted zones: zone 1 (green), zone 2 (red) and zone 3 (blue). The quadrotor, with a ball attached to its bottom through a magnet, starts from the left side, has to cross the centre part (that contains obstacles) and has to find the target; then it has to perform a ballistic launch of the ball over the detected point

- Target detection: after the UAV passes the 2nd Zone it will need to search in the area of Zone 3 for the target (“fire”), labeled with an Alvar tag. The area is clear of obstacles, except for the boundaries of the arena itself. Once the search in 3D space is completed and the target has been found, its 3D position will need to be reconstructed in the global coordinate frame.
- Precision delivery: once the UAV knows the target position, it needs to plan and execute a ballistic launch trajectory to deliver the fire-extinguishing ball to the target. While performing the deliver trajectory, the UAV shouldn’t collide with the environment boundary (walls and ceiling) of the arena. The trajectory planning takes into account the platform constraint such as UAV dynamics and space limits. The fire extinguishing ball release mechanism is performed using magnets.

The drone must conduct the entire mission without any human intervention. The solution must ensure that the UAV maintains a certain safety margin for every obstacle in the arena, established by the judges.

2.1 Quadrotor Characteristics

The organizers provided the UAV platform and the teams were not allowed to make any hardware modifications. The quadrotor, as shown in Fig. 3a–b, is based on the *Hammer Geo RTF* by *Kopterworx*. It is controlled by a Cube Orange Pixhawk flight controller, interfaced with sensors and actuators through a Kore Multi-Rotor Carrier Board. It carries an i7 Intel NUC single-board computer that is connected to the flight controller via Mavlink protocol. The entire solution algorithms run onboard the NUC, while the localization of the quadrotor is provided from a motion capture system. An Intel RGBD camera D435 is mounted forward-facing on the UAV with a horizontal field of view (FOV) of 87°, vertical FOV of 59° and a depth range that spans from 0.3m to 3m. An Arduino Nano board, mounted on the bottom of the vehicle, handles the payload release for the third task.

The teams had to develop a solution to successfully complete the entire mission in the shortest time and without any crush on the obstacles. The overall score takes into account the mission time and the precision of the delivery, in terms of the minimum distance achieved between the ball and the

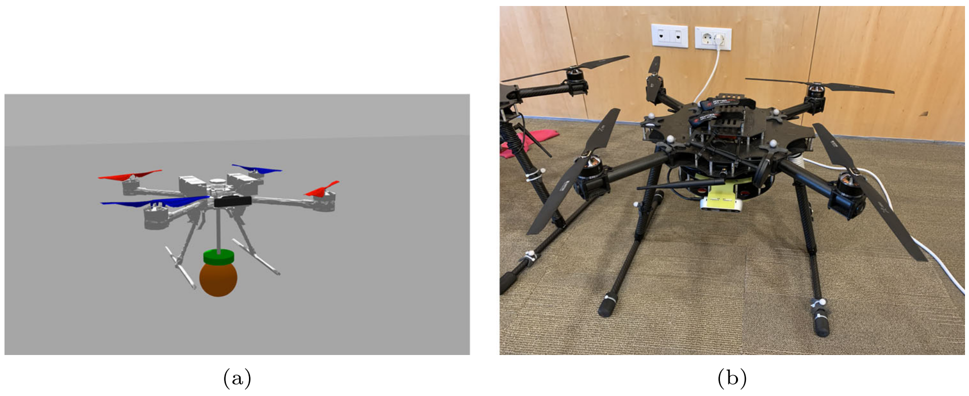


Fig. 3 On the left the quadrotor model used in simulation, on the right the real world multi-copter

target. Any collision makes the entire run invalid. The developed solutions must adapt to the low-level control framework provided by the organisers.

3 Related Works

From the separate study of the three tasks of the competition, it is possible to analyze different state-of-the-art approaches to develop feasible methods enough lightweight to be run onboard a low-budget UAV.

In the literature, different techniques for UAV navigation in unknown environments purely employ onboard sensing and computation. UAVs can estimate the 3D map online, i.e. as the vehicle proceeds in its mission, with SLAM algorithms and let the global planner compute the collision-free path from the current position to an arbitrary goal [14]. Recent works [15] integrate an efficient object-oriented exploration to the mapping strategy to minimize the time spent on the target localization while avoiding obstacles. Loquercio et al. [16] propose a novel approach based on imitation learning to compute high-speed trajectories in unknown complex environments without building the map. The method enables fast reaction times as information about the environment becomes available, according to the range limitation of the depth camera. UAVs can also learn movement policy based on deep reinforcement learning using RGB data [17]. These methods need an onboard GPU to perform run-time inference. Several authors propose other real-time obstacle avoidance algorithms computationally efficient enough to run onboard a small-scale quadrotor with limited CPU resources, most notably Vannese et al. [18] and Baumann [19] have made significant improvements to Vector Field Histogram algorithm [20], extending the method for 3D densely-populated obstacle courses.

On the ballistic launch side instead, the literature relies on Time Optimal Path Parametrization (TOPP) techniques, where the problem is formulated as the search for the fastest way to traverse the admissible space while respecting system constraints. Methods can rely on Numerical Integration [21], Convex Optimization [22] and Reachability Analysis [23]. Numerical Integration algorithms are based on Pontryagin's Maximum Principle, which states that the time-optimal path parameterization consists of alternatively maximally accelerating and decelerating segments. This principle leads to fast implementations, since optimal controls can be explicitly computed, while they are searched during Convex Optimisation approaches. However, methods based on Numerical Integration fail to find feasible solutions as the number of constraints increases. Reachability Analysis instead studies feasible velocities that can be achieved while crossing a computed grid of intermediate points and then select the control sequence that maximises velocity, therefore minimising traversal time. The final state defines the launch parameters for the extinguishing ball, therefore it is constrained to the desired values chosen as in Section 4.2.1.

4 Methods

In this section, the core modules are described. Section 4.1 describes the obstacle avoidance system and its integration in the waypoint global planning structure. The optimal trajectory planning module, implemented both for the waypoint planning and the payload delivery procedure, is described in Section 4.2. The flow between each of this modules is exposed in Section 4.3, together with the useful data that is shared between modules.

4.1 Reactive Local Planner

The implemented obstacle avoidance algorithm, based on the 3DVFH+ algorithm [18, 19], is purely local and reactive, which means that the algorithm has only access to the sensors data of the current time step to compute obstacle avoidance maneuvers, without the need to build a global map of the environment. The algorithm also stores information about the obstacles seen in previous time steps, contrasting problems related to the limitation of the horizontal field of view of the depth camera. The developed local planner for the first phase is composed of the obstacle avoidance system and the pure pursuit waypoint follower [24].

4.1.1 Obstacle Avoidance Module

In several steps, the obstacle avoidance system processes the obstacles information and the current position and orientation of the UAV to compute an obstacle-free direction. First, the 3D point cloud generated by the RGBD sensor is converted into a 2D primary polar histogram through spherical coordinates. Each histogram cell is identified by azimuth and elevation angles and holds the number of points in the corresponding sector with the desired angular resolution α. The memory strategy keeps track of previously seen obstacles by propagating the previous polar histogram to the current drone location and by building the memory histogram. Both the current and the memory histogram are combined into a final histogram based on the number of past time steps in which the cells of the memory histogram were updated.

Each occupied cell (i, j) is enlarged by $\gamma_{i,j}$ angle to take into account the robot size in the local planner algorithm. Given the radius r_r of the circumference that encloses the robot geometry, an additional safety margin r_s and the distance $d_{i,j}$ of each occupied cell, the enlargement angle

$$\gamma_{i,j} = \left\lfloor \frac{1}{\alpha} \arcsin\left(\frac{r_r + r_s}{d_{i,j}}\right) \right\rfloor \tag{1}$$

is computed for each occupied cell.

In the last step, the collision-free direction is selected among all the possible free cells of the final polar histogram within the sensor FOV. The cost function evaluates all the candidate directions according to Eq. 2 as proposed in [18], and the one with the lowest cost is chosen as the optimal direction in that time step. At the current time step k the algorithm considers a possible candidate direction $\delta_i(k)$ among the set of free-obstacle directions $\Delta(k)$, the target direction $\delta_T(k)$, the previously selected direction $\bar{\delta}(k-1)$ and the current yaw angle $\theta(k)$ of the UAV. All these elements compose the cost function

$$J(k) = \mu_1 \left|\delta_i(k) - \delta_T(k)\right| + \mu_2 \left|\delta_i(k) - \frac{\theta(k)}{\alpha}\right| + \mu_3 \left|\delta_i(k) - \bar{\delta}(k-1)\right| \tag{2}$$

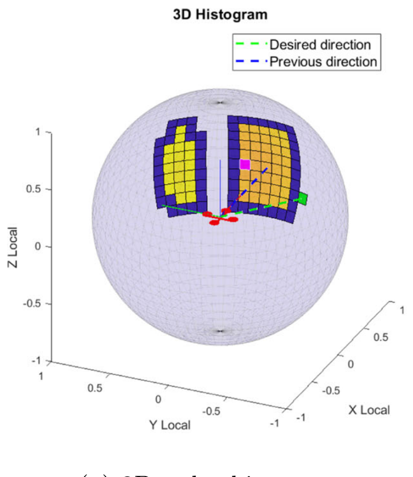
that is minimized to compute the optimal direction in the current time step k

$$\bar{\delta} = \arg\min_{\delta_i \in \Delta} J. \tag{3}$$

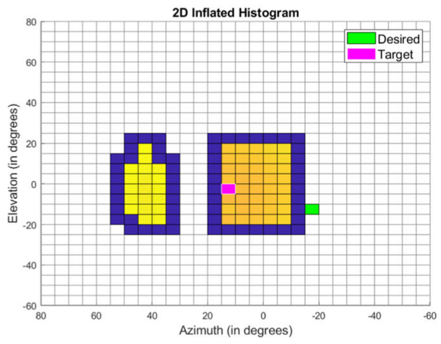
Once an appropriate angular resolution α is chosen, the number of cells in which to evaluate the cost function is limited, therefore the minimisation problem can be solved by just computing each possible value and selecting the optimal direction as that one that achieves the lowest cost value.

The weights μ_i are used to which of the three contributions has a larger impact on the final path shape: μ_1 defines a more goal-oriented robot's behavior, while μ_2 and μ_3 make the UAV keep the current direction, generating a smoother trajectory.

Figure 4 shows the result of the 3DVFH+ algorithm in a generic time step when the drone detects two obstacles, encoded with orange cells. The additionally selected cells due to safety margin are marked in blue. The resulting desired direction $\bar{\delta}$ is represented by the green dashed line and the Pure Pursuit algorithm employs it as the chosen direction of motion, as it is described in Section 4.1.2.



(a) 3D polar histogram



(b) 2D polar histogram

Fig. 4 3D and 2D polar histograms with occupied cells are highlighted in the orange colour band (intensity depends on the distance). The magenta square indicates the desired direction related to the target waypoint, while the green square indicates the direction selected by the algorithm through cost function minimization

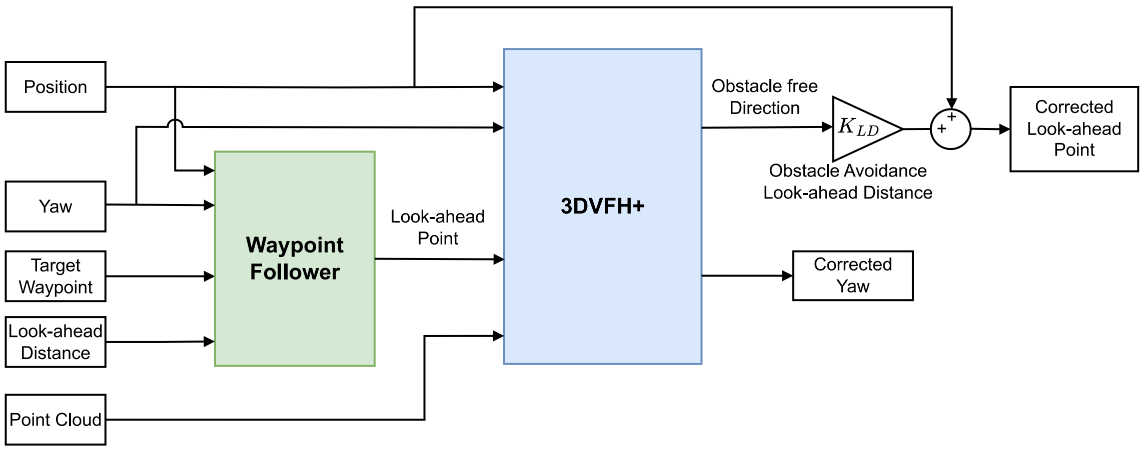


Fig. 5 General scheme of the implemented local planner for the first phase competition. The look-ahead point from Purse Pursuit, the point cloud data and the UAV pose are the inputs of the local planner. It computes the obstacle-free direction and the yaw angle for a collision-free flight to send to the flight controller for waypoint guidance

4.1.2 Local Planner Structure

The aim of the local planner module is to let the UAV pass the cluttered zone (zone 2 in Fig. 2) avoiding obstacles and maintaining a safety margin distance with both them and the walls. The goal position is chosen based on the spawn location of the drone so that the target waypoint is set to the coordinates in front of the drone on the border between zone 3 and zone 2 (see Fig. 2).

The path-tracking method of the local planner is based on the Pure Pursuit algorithm. Given the desired waypoint, the position and the yaw angle of the UAV, the algorithm computes the desired look-ahead point. 3DVFH+ corrects this reference to avoid obstacles. The result from the 3DVFH+ algorithm is a unit vector versor expressed in the local coordinate frame of the quadrotor. It is multiplied by a tunable constant called Obstacle Avoidance Look-ahead Distance K_{LD} and then added to the current UAV position to compute the desired position in the global frame. The magnitude of K_{LD} determines how fast the drone avoids obstacles: a greater look-ahead distance allows the drone to fly faster with a greater risk of colliding with an obstacle; lower values result in slower flights, but the quadrotor has a lower risk of colliding with an obstacle. The complete structure of the local planner is shown in Fig. 5.

4.2 Optimal Trajectory Planner for Payload Delivery

A simple free-fall problem is formulated to define the release condition for the ball and let it intercept the target point. The chosen UAV state, i.e. the center of mass position and velocity, is imposed as the final state of the trajectory that is generated for the delivery task. The problem considers both the overall traversal time and the position, velocity and acceleration limits of the UAV.

4.2.1 Launch Parameters

Ball's position and velocity at the time instant in which the release command is issued are the initial condition, $p_r \in \mathbb{R}^3$ and $v_r \in \mathbb{R}^3$ respectively, for the evolution

$$p(t) = p_r + v_r t + \frac{1}{2} g t^2, \tag{4}$$

where g is the gravity acceleration vector that the ball is subject to, as shown in Fig. 6.

Ball spin and air drag are not considered in this work, since the former is not controlled by the simple release of the magnet, and the latter is an effect not considered in the simulator offered by the organization of the competition. Once the desired collision point $c \in \mathbb{R}^3$ is defined, the release position p_r is imposed relatively to the collision point (in the proposed work, it is 1.5 m higher and 1 m in front of the target). The release velocity v_r must have its unit vector parallel with the vertical plane where p_r and the marker position lies. For horizontal velocities, is trivial to demonstrate that the collision time can be computed as $T_f = \sqrt{2h/g}$, therefore the module of v_r can be directly computed from Eq. 4. The obtained values for p_r and v_r are imposed as the final state for the launch trajectory, defined in the next subsections.

4.2.2 Trajectory Formulation

Desired trajectories in the d-dimensional space can be encoded by continuous curves $\mathcal{C}$ defined through splines, i.e. parametric curves defined with local support functions.

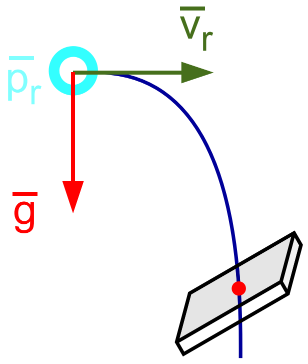


Fig. 6 Ball launch trajectory, from the release point p_r to the marker position, with initial velocity v_r and subject to the gravitational acceleration vector g

Summarizing [25], any parametric curve of degree p can be expressed as

$$\mathcal{C}(u) = \sum_{i=0}^{n} N_{i,p}(u) Q_i, \tag{5}$$

where the free parameter $u \in [0, 1)$ identifies each point that belongs to the curve through the combination of *basis functions* $N_{i,p}(u)$ applied over *control points* $Q_i \in \mathbb{R}^d, \forall i = 0, \dots, n$.

Basis functions have local support, implying that any alteration on the *control polygon* $\mathcal{Q} = [Q_0, \dots, Q_n]$, i.e. the organized set of control points, affects only a section of the curve shape and not its entirety. Indeed, it can be proven that any generic point of the curve is influenced by at most $p + 1$ basis functions. These influence intervals are defined through the segmentation of the curve parameter u along its feasible set. This partitioning is called *knot vector* $\mathcal{U} = [u_0, \dots, u_m]$, i.e. the non-decreasing set of *knot points* u_i, with $m = n + p + 1$.

The basis functions are defined recursively through the *De-Boor recursive formulation* [NURBS Book25, Sec.2.1], starting from that one with degree 0 as

$$N_{i,0}(u) = \begin{cases} 1 & u_i \le u < u_{i+1} \\ 0 & elsewhere \end{cases} \qquad i \in [0, m-1]. \tag{6}$$

The other basis functions with degree $j > 0$ are defined as linear combinations[3] between two basis functions of degree $(j - 1)$, such as for any $j \in [1, p]$,

$$\begin{aligned} N_{i,j}(u) = & \frac{u - u_i}{u_{i+j} - u_i} N_{i,j-1}(u) \\ & + \frac{u_{i+j+1} - u}{u_{i+j+1} - u_{i+1}} N_{i+1,j-1}(u). \end{aligned} \tag{7}$$

[3] Fractions between knot points may produce 0/0. In this case their value is assumed to be zero.

It is important to highlight that any desired derivative of the curve $\mathcal{C}(u)$ can be computed without any actual differentiation, just by employing the same basis functions in Eqs. 6 and 7. Indeed, the generic derivative of order k $(\le p)$ of the curve, that can be written as

$$\mathcal{C}^{(k)}(u) = \sum_{i=0}^{n} N_{i,p}^{(k)}(u) Q_i, \tag{8}$$

is composed by basis functions of degree $j \in [1, p]$ that are derived k times, such as

$$\begin{aligned} N_{i,j}^{(k)}(u) = & \frac{j}{u_{i+j} - u_i} N_{i,j-1}^{(k-1)}(u) \\ & - \frac{j}{u_{i+j+1} - u_{i+1}} N_{i+1,j-1}^{(k-1)}(u). \end{aligned} \tag{9}$$

It is important to notice that not only the curve $\mathcal{C}(u)$, but also all its derivatives are expressed with the same set of knot vector $\mathcal{U}$ and control polygon $\mathcal{Q}$.

The shape of the control polygon $\mathcal{Q}$ therefore identifies the resulting parametric curve and can be considered as the entry-point of an optimisation problem that shapes the trajectory while respecting constraints both on positions and any desired derivative.

The knot vector is chosen to be formulated independently from the control polygon, such as

$$\underbrace{(p+1)}_{\text{zeros}} + \underbrace{(n-p)}_{\text{equispaced in } (0,1)} + \underbrace{(p+1)}_{\text{ones}}. \tag{10}$$

The inner $(n - p)$ terms are chosen to be equispaced in $(0, 1)$, but they could be instead a function of distances between consecutive control points, since the knot vector identifies switching points between basis functions, and than they best perform if related to the control polygon length. This would lead to a nonlinear dependency of the overall spline formulation of the curve $\mathcal{C}(u)$ with respect to the control polygon $\mathcal{Q}$, otherwise linear, therefore this option is not followed in this paper.

The free parameter $u \in [0, 1)$ is linked with the elapsed time t, i.e. the time needed to cover the curve from the starting point $(u = 0)$ up to the current one, by taking into account the release time T_r, initially unknown and therefore subject to optimisation. Assuming a linear dependency between curve parameter u and elapsed time t, i.e. $t = T_r u$, the trajectory $\mathcal{P}(t)$ can be written as a function of the spline curve $\mathcal{C}(u)$, such as

$$\mathcal{P}(t) = \mathcal{C}(t/T_r). \tag{11}$$

Velocities $\mathcal{V}(t)$ assumed by the agent while performing chosen trajectories can be then computed through the chain rule of derivation, such as

$$\mathcal{V}(t) = \frac{1}{T_r}\, \mathcal{C}^{(1)}(t/T_r), \tag{12}$$

just as other derivatives that can be of interest, as accelerations

$$\mathcal{A}(t) = \frac{1}{T_r^2}\, \mathcal{C}^{(2)}(t/T_r). \tag{13}$$

4.2.3 Constrained Optimal Problem

Optimal trajectories can be planned through splines once only a few elements are fixed. Spline degree p must be set a-priori, since it defines the knot vector $\mathcal{U}$ and the basis functions $N_{i,p}(u)$ as in Eq. 7, used to generate the parametric curve and all its derivatives. The trajectory $\mathcal{P}(t)$ can be rewritten in terms of the spline curve $\mathcal{C}(u)$ from Eqs. 5 and 11, therefore as a function of the free parameter u and the control points $Q_i \in \mathbb{R}^d$, $\forall i = 0, \ldots, n$. These parameters are currently unknown and therefore constitute the symbolic variables that need to be optimised.

Formally, the optimisation process aims to find out the best control polygon $\mathcal{Q}$ that minimises some desired metrics on the resulting trajectory. On this matter, elements of interest are the overall traversal time T_r and the accelerations assumed during the entire evolution.

While performing the trajectory, some constraints over positions, velocities and accelerations must be imposed; therefore the trajectory must be sampled over r_i $(i = 1, \ldots, n)$ verification points, represented with red points in Fig. 7.

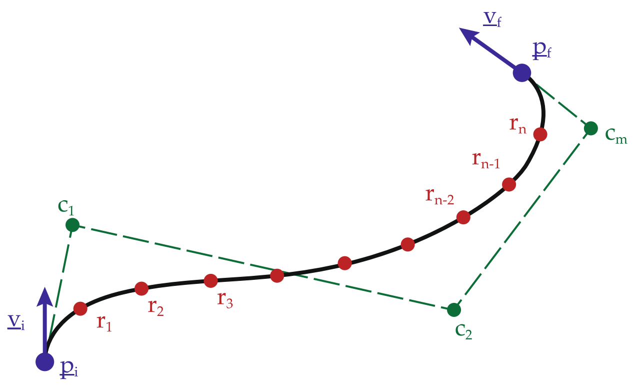


Fig. 7 Any trajectory $\mathcal{P}(t)$ can be defined through the parametric curve $\mathcal{C}(u)$ (black line), where extremities (blue points) are the given initial position $\vec{p}_i$ and the desired release position $\vec{p}_r$; both come together with respective desired velocities $\vec{v}_i$ and $\vec{v}_r$ (blue vectors). Control points (in green) are the optimised variables, while spline samples (in red) are points in which the constraints are verified

The cost function

$$k_t\, T_r^2 + \sum_{i=1}^{n} k_a\, \|\mathcal{A}(t_i)\|^2 \tag{14}$$

is the expression that, if minimised, identifies trajectories that satisfy a trade-off between minimum traversal time and minimum accelerations, weighting these two elements respectively with the scalars k_t and k_a.

Feasible trajectories must respect position, velocity and acceleration constraints that can be encoded as inequalities over these verification points, leading to the complete optimal problem formulation

$$\begin{aligned}
\min_{Q_0,\ldots,Q_n} \quad & k_t\, T_r^2 + \sum_{i=1}^{n} k_a\, \|\mathcal{A}(t_i)\|^2 && \text{(15a)}\\
\text{w.r.t.} \quad & p_{\min} \le \mathcal{P}(t_i) \le p_{\max} && \text{(15b)}\\
& v_{\min} \le \mathcal{V}(t_i) \le v_{\max} && \text{(15c)}\\
& a_{\min} \le \mathcal{A}(t_i) \le a_{\max} && \text{(15d)}\\
& \mathcal{P}(0) = p_i, \quad \mathcal{V}(0) = v_i && \text{(15e)}\\
& \mathcal{P}(T_r) = p_r, \;\; \mathcal{V}(T_r) = v_r && \text{(15f)}
\end{aligned}$$

where the time-dependent UAV position, velocity and acceleration, computed with Eqs. 11, 12 and 13, define the problem constraints Eqs. 15b, 15c and 15d as function of the curve spline $\mathcal{C}(u)$ and its derivatives. The trajectory is constrained in the initial time $t = 0$ and release time $t = T_r$ to the starting UAV position and velocity and to the given target values with Eqs. 15e and 15f, respectively.

4.3 Mission Planner

The developed modules are activated as an event-based sequence, so that the entire firefighting mission is accomplished autonomously by the aerial vehicle. The mission planner module has been implemented through a state machine, the diagram of which is shown in Fig. 8.

The mission starts when the autopilot of the UAV switches to offboard mode. In this state, the UAV receives commands from the on-board computer, where all the modules runs according to the state machine. For the entire duration of the mission, the system is informed of its current pose at 100 Hz with the precision of a motion capture system ($\approx$ 1 cm). The RGB and depth images are acquired at a frequency of 10 Hz. The waypoints for the inspection of the environment and the kinematic constraints are set at the start of the mission. The reference trajectory is sampled with a frequency of 100 Hz. The state machine and all its modules are implemented in ROS Noetic and run within the on-board computer, inter-

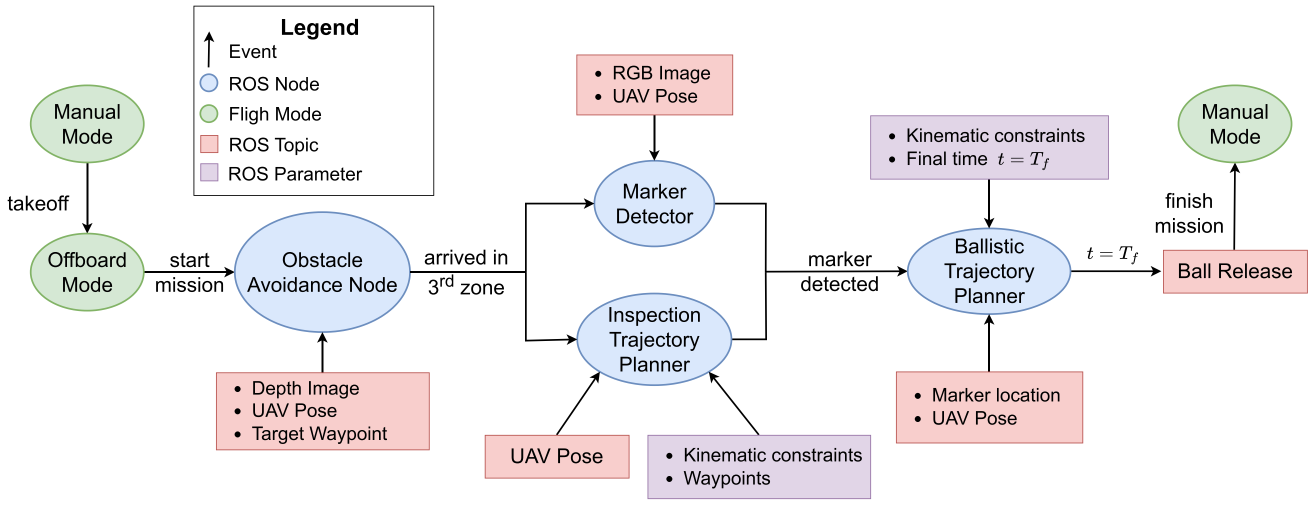


Fig. 8 Scheme of the firefighting mission planner's state machine and modules structure. Boolean conditions trigger the events that let each module activate, one at time

facing with the flight controller through a specific package[4], developed by the organizers of the competition. The payload delivery is achieved by sending a release command through ROS topic that deactivates the magnet that holds the ball, making it free-fall.

5 Results

Both obstacle avoidance and payload delivery have been extensively tested in simulated scenarios of increasing difficulties, achieving the top-5 score for the first phase of the competition, among the other 52 participating teams from more than 22 Countries. Extensive tests on both real and simulated scenarios are performed, allowing us to achieve 3rd place overall during competition finals[5].

The employed simulator platform is Gazebo 11, the official simulator of ROS Noetic. To simulate the propeller dynamics, the *rotors_simulator*[6] package is used. The UAV's attitude and position are gathered directly from the simulator. To simulate a depth camera, the *openni_kinect* plugin is used. The UAV is controlled through the Software-in-the-loop method, using *uav_ros_stack*[7]. Both the simulated and the real UAV flight controller runs Ardupilot[8]. This firmware supports Software-In-The-Loop (SITL) approach, allowing an easier transition from simulation to the real platform.

5.1 Obstacle Avoidance

The developed framework has been tested on simulation in three different arenas to evaluate the performance of the proposed solution. Obstacles disposition changes for each arena, in this way, it is possible to find the parameters combination that allows the drone to safely pass the obstacle zone regardless of the arrangement of obstacles. Obstacles are gray parallelepipeds, of a height equal to the maximum permissible UAV altitude (4 m).

Figure 9a–c show the trajectories of the drone toward the line goal that defines the beginning of phase 3 in three different scenarios for a total of 81 simulations. For each run, the drone is started at different positions (marked in green) to cover all possible cases while moving towards the goal line: in particular, the only waypoint provided to the navigation system has fixed x-coordinate in 2 m, while the y-coordinate varies with the drone y-coordinate. The quadrotor altitude is controlled at 2.5 meters. The safety margins for the histogram enlargement are set in the following way: the parameter for the quadrotor radius is set to 0.55 meters, based on the robot geometry, and the safety distance from each obstacle is set to 0.15 meters. The weights for the cost function in Eq. 2 are chosen through experiments and set to: $\mu_1 = 6$, $\mu_2 = 2$, $\mu_3 = 2.5$. To guarantee a goal-oriented behaviour, the sum of μ_2 and μ_3 must be strictly less than μ_1 [26]. The resolution of the histogram is set to 0.2 m, according to the obstacle dimensions and shape.

At each time step the local planner computes the best direction based on the obstacles that are currently detected. The path is chosen based on the histogram that is evaluated as explained in Section 4.1.1. Since the depth camera has a limited range, some obstacles can be detected only after that the UAV has covered some distance. This is the reason why

4 https://github.com/larics/uav_ros_control

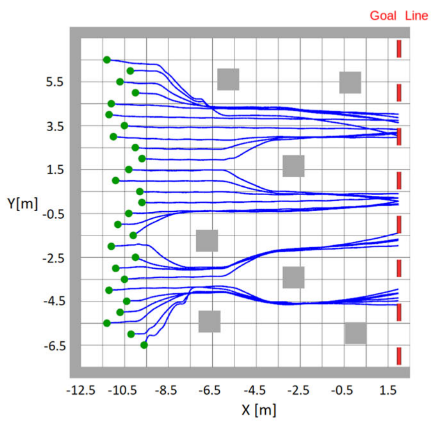
5 http://www.uasconferences.com/2022_icuas/uav-competition-finals/

6 http://wiki.ros.org/rotors_simulator

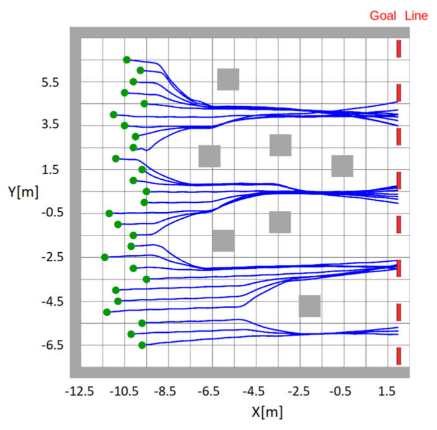
7 https://github.com/larics/uav_ros_stack

8 https://ardupilot.org/

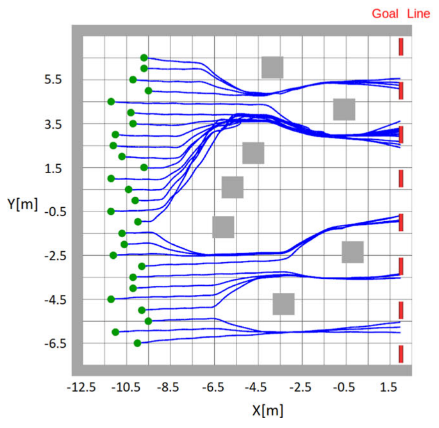
Fig. 9 Trajectories for different start positions in the 3 different arenas **(a-c)** of progressive difficulty. In **(d)** the bar plot shows the percentage of failures for each arena with two values of K_{LD}



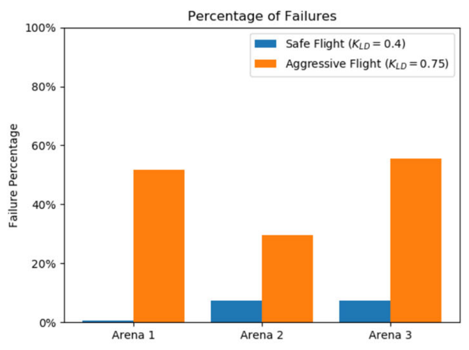
(a) Arena 1



(b) Arena 2



(c) Arena 3



(d)

trajectories are not optimal in the sense of minimum traveled distance. In the cases where the UAV starts near the edges of the arena (in the bottom and the top of the Fig. 9a–c), the traveled trajectory deviates from a straight line due to the safety margins.

In Fig. 9d are reported the results of simulations performed with two different values of the Obstacle Avoidance Look-ahead Distance K_{LD}. The safety margin is constrained as the sphere centered at the geometric center of the UAV, whose radius is the sum of the robot radius r_r and the safety margin r_s. A run is considered failed if the sphere intersects one obstacle. The K_{LD} set to 0.4 is a good trade-off between respecting safety margins and minimizing the time to complete the mission.

5.2 Trajectory Planning

The trajectory planner is employed for both the exploration phase and for the ballistic launch of the extinguishing ball. The optimal control problem stated in Eq. 15 is written in CasADi [27] and solved with IPOPT, a powerful large-scale nonlinear optimizer based on the Interior Point method [28].

5.2.1 Exploration for Marker Detection

Both in simulation and real scenarios, the detection of the launch point for the payload is traced back to the pose detection of an Alvar marker, performed through open-source methods. When the UAV passes the obstacle zone, it starts to scan the environment driving through pre-planned waypoints, in order to inspect the area and find the marker. The desired waypoints employed by the trajectory planner are chosen by the a-priori knowledge of the environment boundaries. In both simulation and real experiments, these waypoints are chosen in order to allow the drone to perform a perimeter scan, that is repeated until it detects the marker. The appropriate linear and angular velocity can be constrained easily with the proposed trajectory planner, reducing the motion blur on the front camera and therefore facilitating the marker detection.

5.2.2 Trajectory for Ballistic Launch

Once the target point is detected, the launch position and velocity for the extinguishing ball and therefore the UAV are

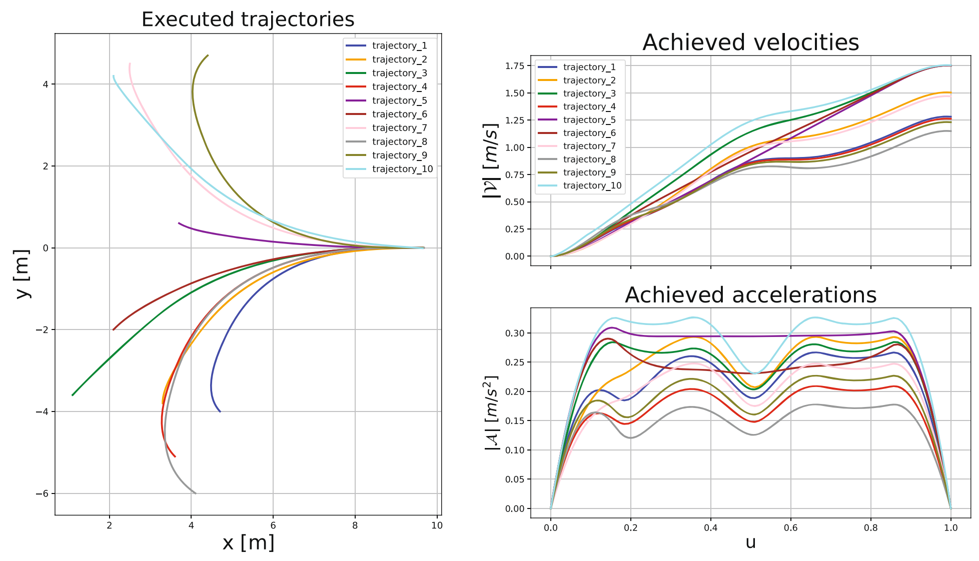


Fig. 10 Test trajectories resulting from different starting position and zero initial velocities for the UAV, with the same final desired position and different target velocity. On the right, the obtained achieved velocities and accelerations while performing the trajectories shown on the left. The *u* in the abscissa represents the percentage of traveled trajectory

computed by resolving the corresponding free-fall problem for the given relative distance, that is imposed. The resulting desired release position p_r and velocity v_r are constraints in the final state of the optimal problem Eq. 15.

Figure 10a shows the computed trajectories that have to be performed in order to reach the same target position with different velocities, starting in random positions and zero initial velocity. Achieved velocities and accelerations assumed while each trajectory is performed are exposed in Fig. 10b. By requesting zero acceleration at the final time, the payload delivery can be performed more robustly in the event of payload release delays. The velocity and acceleration constraints, set to $2\frac{m}{s}$ and $0.35\frac{m}{s^2}$ respectively, are respected in each test. The Table 2 evinces initial distance from the target, the overall trajectory length and the traversal time. The overall traversal time of each test is consistent with the initial distance between the UAV and the target position.

5.3 Experimental Tests

For live trials at the conference venue in Dubrovnik, Croatia, the organizers set up an indoor arena of size 10 x 7.5 x 3.5 meters with a motion capture system and provided the quadrotor that executes the solution algorithm, as shown in Fig. 11a. During the first two days, the 5 finalist teams set up and integrate the respective solutions, performed the overall sanity check of the code and made preliminary autonomous flights on the real platform. The final competition were held on June 23$^{\text{rd}}$, where each team had 30 minutes of flight available in the morning and other 30 minute in the afternoon to perform multiple runs. The final scoring scheme takes into account the average score of all runs, in a similar way of the simulation phase.

Regarding the experimental tests of the obstacle avoidance task, the UAV must maintain a clearance of at least 1 meter from the obstacles and the walls of the arena (measured from

Table 2 Metrics of interest of the above test trajectories

	Traversal time [s]	Initial distance [m]	Trajectory length [m]
trajectory_1	10.61	7.05	7.57
trajectory_2	9.66	8.14	8.05
trajectory_3	9.84	10.07	9.69
trajectory_4	13.53	8.58	9.55
trajectory_5	6.80	6.83	5.98
trajectory_6	8.67	8.63	7.98
trajectory_7	11.43	9.18	9.30
trajectory_8	15.92	8.77	10.58
trajectory_9	12.47	7.70	8.64
trajectory_10	8.68	9.39	9.09

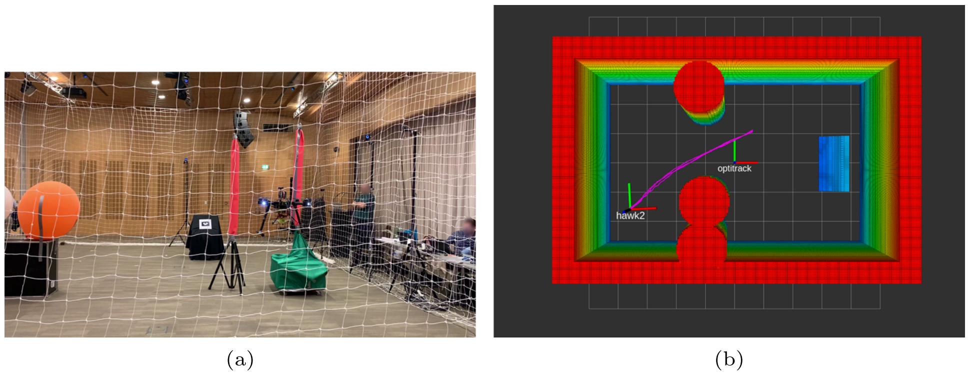


Fig. 11 The **(a)** indoor arena at the conference, where the finals are contended. In **(b)** the obstacle avoidance trajectories in three runs during the final phase are shown with the visualization tool RViz. The UAV starts from the pose *hawk2* and traverses the purple line. Drone pose is measured by the motion capture system and it is provided with respect to the global fixed frame *optitrack*. During the obstacle avoidance phase, this global pose is needed only for challenge evaluation purposes, while the module relies only on local sensing

the geometric center of the UAV). Based on the layout of the obstacles, the organizers have created an octomap to check that safety margins are respected during the run. Figure 11b shows trajectories of three runs during the obstacle avoidance task. The obstacle avoidance algorithm shows similar performance to the simulation results regardless of the shape of the obstacles demonstrating that the method is independent of the form of the obstacles.

Figure 12 shows the sequence of a valid launch of the ball, performed by releasing the magnet at the final position of the trajectory computed by the optimal planner. Unlike in the simulation phase, the marker was inclined by 45°, to facilitate the detection task. The delivery trajectory is considered valid by the judges if the ball reaches at least a corner of the marker, 20 cm wide. The correct tracking for the UAV of both the desired position and velocity at the launch instant has proven to produce an acceptable ballistic trajectory of the ball. The lack of a tracking system for the ball itself prevents further analysis of the ballistics of this phase. Imposing zero acceleration in the launch instant reduces launch errors that can incur due to unpredicted delays of the magnet release.

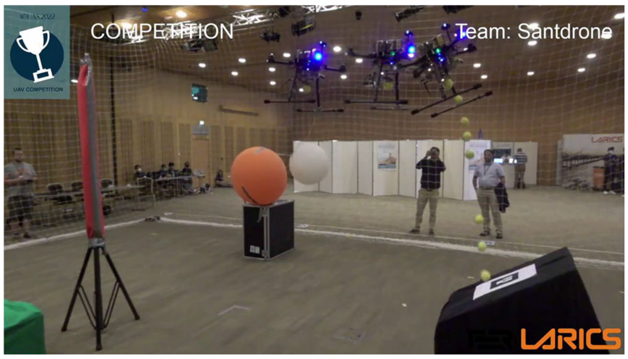


Fig. 12 Image sequences of a valid ball launch during the competition. The ball performs a parabolic motion to reach the detected marker, exploiting the UAV kinematics

6 Conclusion and Discussion

In this paper, an obstacle avoidance and motion planner method to perform complex firefighting missions for UAVs is presented. The proposed real-time solution is composed of two different planners. The local planner allows the quadrotor to navigate in an unknown environment reactively, with reduced hardware resources and relying only on a depth sensor with limited field of view and a localization system. The global trajectory planner instead computes the optimal trajectory that minimises the overall traversal time and maximises the accuracy of the ball delivery, while respecting the quadrotor kinematics. The proposed system was extensively tested in simulation and in real world during ICUAS'22 Competition, successfully completing the mission in runs of increasing difficulty, proving robustness on environmental changes. Our team achieved top-5 score for the simulation phase of the competition, among the other 52 participating teams, and the 3rd position in the final phase. The proposed solution represents a robust and efficient approach to solve firefighting tasks for fully autonomous aerial vehicles with limited onboard hardware, and a feasible solution for any exploration and delivery task that has to deal with a hard-to-reach environment and complex payload delivery. The proposed framework can be utilized for real-case firefighting scenarios leveraging the accuracy of a Real-time kinematic positioning (RTK) GPS system, by allowing the multirotor to track the trajectory for an accurate ballistic launch.

In future works we plan to improve our framework by merging the obstacle avoidance and the delivery modules.

Additionally, we aim to deploy this framework for more realistic experiments in outdoor environments, considering localization uncertainties to the UAV, caused by urban canyons that affect GPS performance. On this matter, a preliminary study on delivery errors caused by wrong estimations of either the UAV position or velocity respect to an inertial frame leads to acceptable results. Indeed, state-of-the-art UAVs with common localization filters based on both Visual-Inertial Odometry and GPS systems are able to localize with a position uncertainty of centimeters. Since the release position p_r acts linearly in the Eq. 4, the collision point will be missed by the same amount, that is an acceptable error for the success of fire extinguishing tasks.

Acknowledgements This work was supported by LARICS (Laboratory for robotics and intelligent control systems) from the Faculty of Electrical Engineering and Computing of the University of Zagreb. The laboratory organized the student UAV competition held during ICUAS conference on Unmanned Aircraft systems in Dubrovnik from 21st until 24th of June 2022.

Author Contributions All authors contributed to the study conception and algorithm design. The first draft of the manuscript was written by Michael Mugnai and Massimo Teppati Losè, as well as the integration of the received peer-reviews. All authors read and approved the final manuscript.

Funding Open access funding provided by Scuola Superiore Sant'Anna within the CRUI-CARE Agreement.

Declarations

Conflicts of interest The authors have no relevant financial or non-financial interests to disclose.

References

1. Shakhatreh, H., Sawalmeh, A.H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N.S., Khreishah, A., Guizani, M.: Unmanned aerial vehicles (uavs): a survey on civil applications and key research challenges. Ieee Access **7**, 48572–48634 (2019)
2. Peskoe-Yang, L.: Paris firefighters used this remote-controlled robot to extinguish the notre dame blaze. IEEE Spectrum: Technology, Engineering, and Science News (2019)
3. Fahy, R.F., Petrillo, J.T., Molis, J.L.: Firefighter fatalities in the us-2019. Natl. Fire Prot. Assoc. pp. 1–26 (2020)
4. Lenz, C., Quenzel, J., Periyasamy, A.S., Razlaw, J., Rochow, A., Splietker, M., Schreiber, M., Schwarz, M., Süberkrüb, F., Behnke, S.: Team nimbro's ugv solution for autonomous wall building and fire fighting at mbzirc 2020, (2021). arXiv:2105.11979
5. Wang, K., Yuan, Y., Chen, M., Lou, Z., Zhu, Z., Li, R.: A study of fire drone extinguishing system in high-rise buildings. Fire **5**(3), 75 (2022)
6. Imdoukh, A., Shaker, A., Al-Toukhy, A., Kablaoui, D., El-Abd, M.: Semi-autonomous indoor firefighting uav. In: 2017 18th International Conference on Advanced Robotics (ICAR), IEEE, pp. 310–315 (2017)
7. Perez-Grau, F., Ragel, R., Caballero, F., Viguria, A., Ollero, A.: Semi-autonomous teleoperation of uavs in search and rescue scenarios. In: 2017 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, pp. 1066–1074 (2017)
8. Alon, O., Rabinovich, S., Fyodorov, C., Cauchard, J.R.: Drones in firefighting: a user-centered design perspective. In: Proceedings of the 23rd international conference on mobile human-computer interaction, pp. 1–11 (2021)
9. Brancalião, L., Gonçalves, J., Conde, M.Á., Costa, P.: Systematic mapping literature review of mobile robotics competitions. Sensors **22**(6), 2160 (2022)
10. Spurny, V., Pritzl, V., Walter, V., Petrlik, M., Baca, T., Stepan, P., Zaitlik, D., Saska, M.: Autonomous firefighting inside buildings by an unmanned aerial vehicle. IEEE Access **9**, 15872–15890 (2021)
11. Quenzel, J., Splietker, M., Pavlichenko, D., Schleich, D., Lenz, C., Schwarz, M., Schreiber, M., Beul, M., Behnke, S.: Autonomous fire fighting with a uav-ugv team at mbzirc 2020. In: 2021 International conference on unmanned aircraft systems (ICUAS), IEEE, pp. 934–941 (2021)
12. Aydin, B., Selvi, E., Tao, J., Starek, M.J.: Use of fire-extinguishing balls for a conceptual system of drone-assisted wildfire fighting. Drones **3**(1), 17 (2019)
13. Qin, H., Cui, J.Q., Li, J., Bi, Y., Lan, M., Shan, M., Liu, W., Wang, K., Lin, F., Zhang, Y., et al.: Design and implementation of an unmanned aerial vehicle for autonomous firefighting missions. In: 2016 12th IEEE International Conference on Control and Automation (ICCA), IEEE, pp. 62–67 (2016)
14. Alzugaray, I., Teixeira, L., Chli, M.: Short-term uav path-planning with monocular-inertial slam in the loop. In: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 2739–2746 (2017)
15. 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: 2021 International Conference on Unmanned Aircraft Systems (ICUAS), IEEE, pp. 330–337 (2021)
16. Loquercio, A., Kaufmann, E., Ranftl, R., Müller, M., Koltun, V., Scaramuzza, D.: Learning high-speed flight in the wild. Sci. Robot. **6**(59), 5810 (2021)
17. Singla, A., Padakandla, S., Bhatnagar, S.: Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge. IEEE Trans. Intell. Transp. Syst. **22**(1), 107–118 (2019)
18. Vanneste, S., Bellekens, B., Weyn, M.: 3dvfh+: Real-time three-dimensional obstacle avoidance using an octomap. In: MORSE 2014 Model-Driven Robot Software Engineering: Proceedings of the 1st International Workshop on Model-Driven Robot Software Engineering Co-located with International Conference on Software Technologies: Applications and Foundations (STAF 2014), York, UK. Accessed 21 July 2014/Assmann, Uwe [edit.], pp. 91–102 (2014)
19. Baumann, T.: Obstacle avoidance for drones using a 3dvfh* algorithm. Spring Term **2018**, 67 (2018)

20. Borenstein, J., Koren, Y., et al.: The vector field histogram-fast obstacle avoidance for mobile robots. IEEE Trans. Robot. Autom. **7**(3), 278–288 (1991)
21. Pham, Q.-C.: A general, fast, and robust implementation of the time-optimal path parameterization algorithm. IEEE Trans. Rob. **30**(6), 1533–1540 (2014). https://doi.org/10.1109/TRO.2014.2351113
22. Hauser, K.: Fast interpolation and time-optimization with contact. The International Journal of Robotics Research **33**(9), 1231–1250 (2014). https://doi.org/10.1177/0278364914527855
23. Pham, H., Pham, Q.-C.: A new approach to time-optimal path parameterization based on reachability analysis. IEEE Trans. Rob. **34**(3), 645–659 (2018). https://doi.org/10.1109/TRO.2018.2819195
24. Coulter, R.C.: Implementation of the pure pursuit path tracking algorithm. Technical report, Carnegie-Mellon UNIV Pittsburgh PA Robotics INST (1992)
25. Piegl, L., Tiller, W.: The NURBS Book, 2nd edn. Springer, New York, USA (1996)
26. Ulrich, I., Borenstein, J.: Vfh+: Reliable obstacle avoidance for fast mobile robots. In: Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146), vol. 2, pp. 1572–1577 (1998). IEEE
27. Andersson, J.A.E., Gillis, J., Horn, G., Rawlings, J.B., Diehl, M.: CasADi – A software framework for nonlinear optimization and optimal control. Math Program Comput (2018)
28. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math. Program. **106**, 25–57 (2006)

Michael Mugnai is a Ph.D. student in Perceptual Robotics at Scuola Superiore Sant'Anna, with a master's degree in Robotics and Automation and a bachelor's degree in Mechanical Engineering. His research mainly focuses trajectory planners for autonomous robotics, on both ground and aerial. He was a visiting student in MECO laboratory at KU Leuven, Belgium, where he worked on vineyard coverage with heavy vehicles. He took part of several autonomous competitions on the field of exploration in occluded environments.

Massimo Teppati Losè received the master's degree in Robotics and Automation Engineering and the bachelor's degree in Biomedical Engineering, in 2022 and 2019 respectively, from the University of Pisa, Italy. He is currently pursuing the Ph.D. degree in Emerging Digital Technologies, Perceptual Robotics curricula, from the Scuola Superiore Sant'Anna, Italy. He is currently within the Intelligent Automation Systems division at PERCRO Lab, from the Institute of Mechanical Intelligence of the Scuola Superiore Sant'Anna. He has participated in several national and international competitions of aerial and ground robots. His research interests include Multi-Robot Collaborative Localization and Exploration in GNSS-denied environments.

Massimo Satler received the MSc degree in Automation Engineering from the University of Pisa in 2008 and the PhD degree in Emerging Digital Technologies perceptual robotics curricula, from the Scuola Superiore Sant'Anna in 2012. He was a visiting PhD Student at the University of Twente, The Netherlands, in 2012, where he worked at the Robotics and Mechatronics Group (RAM Lab.). He is currently within the Intelligent Automation Systems division at PERCRO Lab. of the Institute of Mechanical Intelligence of the Scuola Superiore Sant'Anna. His main research interests include Field Robotics, Sensor Fusion, and AI applied to the control and perception of mobile devices.

Carlo Alberto Avizzano is Associate Professor of Robotics and Automation at Scuola Superiore Sant'Anna, Pisa. At Scuola Sant'Anna he is the coordinator of the PERCRO laboratory, the Intelligent Automation Systems group and the Department of Excellence on Robotics and artificial intelligence. His academic and research activities deal with control of cyberphysical systems, intelligent control systems, autonomous robotics, human computer interfaces and wearable systems. Field applications include: transportation systems, medical, sport and wellbeing devices, safety and security, cultural heritage, land, territory and civil infrastructure monitoring, rehabilitation and assisting devices. Carlo Alberto Avizzano has been general chair for the IEEE RoMan Conference and served many international conferences and journals in different roles. He has authored/co-authored about 170 articles in international journals and peer reviewed conferences.