IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 3, 2022, accepted March 27, 2022, date of publication April 4, 2022, date of current version April 11, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3164434

Parabolic Airdrop Trajectory Planning for Multirotor Unmanned Aerial Vehicles

ANTUN IVANOVIC, (Student Member, IEEE), AND MATKO ORSAG, (Member, IEEE)
Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: Antun Ivanovic (antun.ivanovic@fer.hr)

This work was supported by in part by the Croatian Science Foundation under Project Specularia UIP-2017-05-4042 [1], and in part by the European Union through the European Regional Development Fund—The Competitiveness and Cohesion Operational Program (KK.01.1.1.04.0041) under Project Heterogeneous Autonomous Robotic System in Viticulture and Mariculture—HEKTOR [2].

ABSTRACT In this paper we present a motion planning method for the parabolic airdrop using multirotor UAVs. The first step is to determine a set of parabolic trajectories that can reach a desired target. Next, we plan a path to the payload launch point, and based on that path we plan a collision-free trajectory that safely executes both launch and stopping motions. The described motion planning method is extensively tested in the simulation environment: by executing many trajectories through repeatability analysis; planning in a large city-like environment; and planning in a dense office environment. Furthermore, the method is tested through real-world experiments in indoor and outdoor environments. In both environments, we performed the repeatability analysis and the obstacle avoidance experiments while delivering the payload to the target. The video demonstrating our simulation and experimental analyses is available at: https://youtu.be/HJCH2vJEuuo

INDEX TERMS Unmanned aerial vehicles, motion planning, unmanned autonomous systems.

I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have been around for a significant time. Their versatility makes them interesting to the research community, industrial enterprises, small business owners, hobby enthusiasts and general public. This vast interest sparked a huge growth of the UAV market, offering new models with enhanced capabilities to consumers. The research community has always been focused on envisioning and investigating new concepts, propelling the world of UAVs forward. Consequently, UAVs can be equipped with a broad sensory apparatus, like cameras, LIDARs, etc., for observing and obtaining information about the environment. Mounting manipulators on UAVs gave them the ability to interact and change the environment, expanding operational capabilities even further.

Two main branches of these vehicles can be distinguished: fixed-wing and multirotor aircraft. Each branch has its own advantages over the other. Fixed-wing vehicles can usually cover larger distances opposed to multirotors, which makes them energy efficient. On the other hand, multirotors' Vertical Takeoff and Landing (VTOL) capability is a great advantage in dense environments, as they don't require a lot of space to get airborne. However, this capability comes with a price because multirotors use a lot of energy just to stay airborne.

The task of precision airdrop is delivering a payload to a specified target in the environment. These tasks are typically performed with fixed-wing vehicles from significant heights and by employing an active payload guidance system. In this paper, we are taking the inspiration from this task, however, we apply it to multirotor UAVs. Unlike fixed-wing UAVs, multirotors work best in cluttered environments, so we focus on delivering a payload in obstacle-populated environments which yield much shorter payload trajectories. We compare our approach to similar examples using multirotors, since fixed-wing UAVs fall in a completely different range of applications.

A lot of work has already been done in the field of precision airdrop. Work of researchers in [3], [4] considers the package deployment using parachutes while estimating wind conditions to improve the precision. Researchers in [5] provide a similar approach, but include the aerodynamic properties of the package in the release point calculation. The work in [6] presents the Joint Precision Airdrop System (JPAS) for precise military resupply, and [7] analyzes the optimal dispersion altitude of multiple packages and solves a traveling

The associate editor coordinating the review of this manuscript and approving it for publication was Jamshed Iqbal.

salesman problem with minimizing the risk for ground troops pickup of packages. Researchers in [8] take it a step further and propose a guidance system capable of steering a parafoil in an environment with sparse obstacles, while [9] uses a wind shear model to improve the landing precision of the parafoil.

Parafoils are a good solution in an obstacle-free environment and are not suitable for dense environments such as forests, cities, etc. In some cases, a ballistic free fall can be used to deploy packages. Researchers in [10] are carefully calculating the approach for the ballistic airdrop, while considering the wind speed. Later on, they extended their approach to an autonomous ballistic airdrop relying on the wind field and air resistance models in [11]. The work in [12] revolves around the precise ballistic airdrop based on the Global Positioning System (GPS).

The common denominator of the methods presented in above mentioned papers are i) using the fixed-wing aircraft for the delivery, and ii) the payload is usually released from a significant height. Although the energy efficiency of the fixed-wing aircraft enables flying to distant areas, they are not suitable for flying in cluttered environments. On the other hand, the agility of small-scale multirotor vehicles is perfect for parabolic airdrop applications in cluttered environments, which is the focus of this paper. We investigate a situation when the released package is considered to traverse a short distance, for example through a window of a building. In such a scenario, the UAV needs to execute aggressive and precise maneuvers in order to successfully deliver the package while avoiding collision with the environment. UAV control mechanisms for such aggressive and agile maneuvers are very well established. The work in [13] presents results with multiple quadrotors juggling a ball while in [14] an interception trajectory is planned for the UAV to catch a projectile. Similar approach is reported in [15], constraining the high order derivatives while planning a spline trajectory. Researchers in [16] plan aggressive motions through narrow passages and execute a loop motion to perch to an object, while the work in [17] considers aggressive search-based trajectories. Finding the collision-free feasible path and interpolating such a path with a trajectory has been reported in [18], [19], while researchers in [20] develop a local planner with the capability to avoid previously unknown obstacles. The trajectory planning algorithms mostly focus on convex optimization or mixed integer programming [21]–[23] to ensure smooth trajectories. Apart from the convex optimization and dynamic programming, the numerical integration offers a different approach to the trajectory planning. The work done in [24], [25] develops the Time Optimal Path Parameterization (TOPP), which relies on the bang-bang principle on the generalized torque. The authors have accounted for the numerical instabilities and singularities, yielding time optimal trajectories in a short computation time. The approach presented in these papers is utilized in this work because the widely accepted practical TOPP implementation reliably plans trajectories while respecting the provided velocity and acceleration constraints.

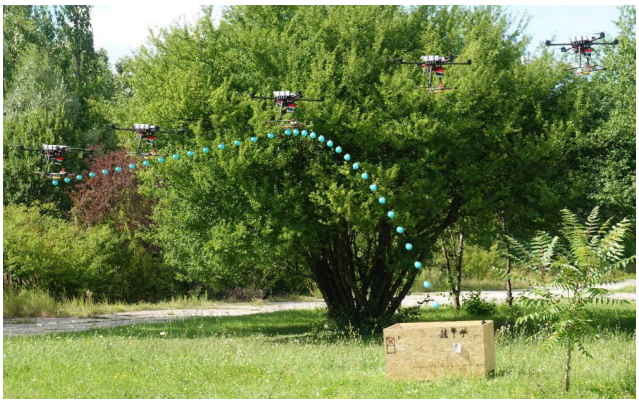

FIGURE 1. An illustration of an experiment performed outdoors. The UAV carries the ball to the release point and executes the stopping motion afterwards. After the release instance, the free-fall parabolic trajectory can be observed.

Researchers in [26] focus on indoor firefighting using UAVs that spray water on the detected fire. They concluded this approach is inefficient and the fire is not likely to be extinguished. Therefore, in future work they propose using an ejection mechanism launching fire-extinguishing capsules. This is in line with our solution, however, we base our approach on exploiting the UAV dynamics in order to launch the fire-extinguishing agent. In our previous work, we have been concentrating on the motion planning for a heterogeneous aerial-ground team performing delivery missions [27] as well the as aerial manipulator end-effector motion planning based on the dynamical model of the system [28].

As for any ballistic airdrop, the projectile has to reach a certain point and a corresponding velocity in the world frame. Due to possible proximity to obstacles at the release point, a UAV has to be capable of aggressive maneuvering in order to reach the launch point from a particular direction with a particular velocity in an obstacle-rich environment. From the trajectory planner perspective, these constraints have to be satisfied for a successful parabolic airdrop, as well as pass through a potentially large number of waypoints provided by the path planner. Although the convex optimization approaches can directly incorporate these constraints in the optimization process, their planning time tends to be significantly higher than the numerical integration due to a large number of waypoints. Therefore, the TOPP-RA numerical integration method is augmented within this paper to account for the parabolic airdrop constraints, while retaining the ability to plan in a short amount of time. The inspiration for this work is drawn from the practical use case of extinguishing a fire and the aim is to develop a trajectory planner capable of performing the parabolic airdrop.

A. CONTRIBUTIONS

Working in cluttered environments requires careful consideration to verify an obstacle-free ballistic trajectory. Therefore, the first contribution of this paper is an algorithm generating a set of potential ballistic collision-free trajectory

candidates between the released payload and the environment. The second contribution is the collision-free trajectory planning algorithm for the UAV, that builds on top of the payload trajectory, enabling the UAV to reach the launch point while avoiding collision and release the projectile with the proper velocity. We present the results of an extensive analysis, starting from a realistic simulation environment. Next, we conducted an experimental analysis starting from an indoor laboratory environment, building towards experiments in a relevant outdoor environment. The results obtained from the simulations and the experiments are compared in terms of repeatability and precision of the parabolic airdrop, with the results presented and disseminated in this work.

B. ORGANIZATION

The notation used throughout the paper is given in Table 1. In section II we derive a mathematical model of a multirotor UAV with a rigidly attached payload. This is followed by section III where we formally write the parabolic trajectory which is the base for path and trajectory planning. The proposed planning algorithm is tested and verified in a simulation environment, as shown in section IV. Following the simulation, we describe the performed experiments in section V. Next, we discuss the results obtained from the simulation and experiments in section VI. Finally, we draw conclusions in section VII.

II. MATHEMATICAL MODEL

Within this section we derive the mathematical model of a multirotor with an attached payload. The coordinate systems and transformations used in modeling are depicted in Fig. 2. We consider both kinematics and dynamics of such a system in order to satisfy the payload release conditions, which will be discussed in the following sections. The notation used in the paper can be found at the end of the paper in Table 1.

A. KINEMATICS

The inertial frame is denoted with L_{W} while the body fixed frame is L_{B}. Note that we express all vectors in the inertial frame. The gravity is observed along the negative z_{W} axis. We can define the generalized coordinates of the UAV as $\mathbf{q}_{\mathrm{B}}^{*} = \left[\mathbf{p}_{\mathrm{B}}^{T}\ \boldsymbol{\Theta}_{\mathrm{B}}^{T}\right]^{T} \in \mathbb{R}^{6}$, where $\mathbf{p}_{\mathrm{B}} = \left[x\ y\ z\right]^{T}$ represents the position of the UAV in the inertial frame and $\boldsymbol{\Theta}_{\mathrm{B}} = \left[\phi\ \theta\ \psi\right]^{T}$ represents the attitude vector. As the multirotor UAV is an underactuated system, we cannot control all six Degrees of Freedom (DoF) simultaneously. Because of that limitation we denote the reduced generalized coordinates as $\mathbf{q}_{\mathrm{B}} = \left[\mathbf{p}_{\mathrm{B}}^{T}\ \psi\right]^{T} \in \mathbb{R}^{4}$. This yields only four controllable DoF, which are in this case the position of the UAV in the inertial coordinate frame $\mathbf{p}_{\mathrm{b}}$ and the yaw angle rotation ψ. The yaw angle is measured around an intermediate axis which is parallel to the z_{w} axis and passes through the UAV body coordinate system origin L_{b}.

TABLE 1. The notation used throughout the paper.

Symbol	Description
L_a	Coordinate system
$\mathbf{q}^{*} \in \mathbb{R}^{6}$	UAV generalized coordinates
$\mathbf{q} \in \mathbb{R}^{4}$	UAV generalized coordinates without roll and pitch
$\mathbf{p}_{\mathrm{B}}$	UAV position vector
$\boldsymbol{\Theta}_{\mathrm{B}}$	UAV attitude vector
$\phi\quad\theta\quad\psi$	Roll, pitch and yaw
$\boldsymbol{T}_a^b \in \mathbb{R}^{4\times4}$	Transformation matrix between frames a and b
$F_i,\ M_i$	Propeller thrust force and moment
$k_{\mathrm{F}},\ k_{\mathrm{M}}$	Propeller thrust and drag coefficients
Ω_i	Propeller angular velocity
$m_{\mathrm{UAV}},\ m_p$	Mass of UAV and payload
$\boldsymbol{I}_{\mathrm{UAV}},\ \boldsymbol{I}_p^{\mathrm{B}}$	UAV and payload moment of inertia
$\boldsymbol{I}_s$	System moment of inertia
$\boldsymbol{r}_p$	Payload position expressed in L_{B}
$\boldsymbol{u}$	UAV control inputs
$\boldsymbol{K}$	Control mapping matrix
$\boldsymbol{R}$	UAV rotation matrix
$\boldsymbol{g}$	Gravity vector
$v_{0,p}$	Payload initial velocity magnitude
θ_p	Payload launch angle
ψ_p	Payload direction angle
$\boldsymbol{C}_p$	Parabola configuration vector
d_p	Parabola horizontal displacement
h_p	Parabola vertical displacement
τ_{L}	Launch point
τ_{T}	Target point
τ_{S}	Starting point
$\boldsymbol{v}_{\mathrm{L}}$	Launch velocity
$\boldsymbol{C}_{\mathrm{L}}$	Launch point configuration
$\mathscr{L}_p$	Set of launch point configurations
$\boldsymbol{w}_s$	Starting waypoint
$\boldsymbol{w}_{\mathrm{L}}$	Launch waypoint
$\mathscr{P}$	Planned path
$\mathscr{T}_{\mathrm{T}}$	TOPP-RA trajectory
$\mathscr{T}_{\mathrm{L}}$	Launch trajectory
$\mathscr{T}_{\mathrm{S}}$	Stopping trajectory
$\boldsymbol{x}$	Single trajectory point with position, velocity and acceleration
$\boldsymbol{C}_s$	Spline configuration vector
$a_0 \ldots a_5$	5th order spline coefficients
l_{T}	Length of trajectory between two points
r_l	Trajectory length ratio
F_d	Drag force
ρ_{air}	Air density
v	Airspeed
C_d	Drag coefficient of a sphere
$Re(v,r)$	Reynolds number
A	Projected surface area of a sphere
r	Sphere radius
μ_{air}	dynamic air viscosity

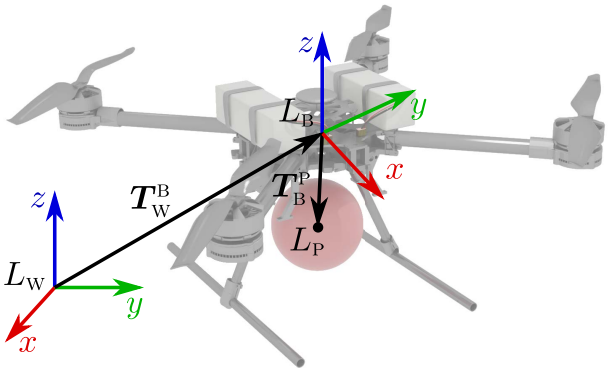

FIGURE 2. Coordinate systems and transformations of the UAV with an attached payload. L_w is the inertial coordinate system, L_b is attached to the UAV body and L_p is attached to the payload center of mass. T_w^b is the transformation matrix from the inertial coordinate system to the UAV geometric center, while T_b^p is the transformation from the UAV geometric center to the payload geometric center.

The L_P coordinate system represents the center of mass of the payload. We consider the payload to be rigidly attached below the UAV's center of mass. The $\boldsymbol{T}_W^B \in \mathbb{R}^{4\times4}$ is the transformation matrix from the inertial coordinate system to the UAV geometric center, while $\boldsymbol{T}_B^P \in \mathbb{R}^{4\times4}$ is the transformation matrix from UAV's geometric center to the payload center of mass. To obtain the position and the orientation of the payload in the inertial coordinate system, one can simply multiply these two transforms $\boldsymbol{T}_W^P = \boldsymbol{T}_W^B \cdot \boldsymbol{T}_B^P$. Given the desired payload position in the inertial frame, one can calculate the transformation of the UAV in the inertial frame with:

$$\boldsymbol{T}_W^B = \boldsymbol{T}_W^P \cdot (\boldsymbol{T}_B^P)^{-1}, \tag{1}$$

which will allow us to specify the payload launch point in the inertial frame, while being able to calculate the position and the orientation of the UAV to satisfy the desired launch point.

B. DYNAMICS

A typical multirotor consists of n_p rotors placed in a single plane, where each rotor produces the thrust force $F_i = k_F\Omega_i^2$ and moment $M_i = k_M\Omega_i^2$. Constants k_F and k_M are the propeller thrust and drag coefficients, respectively, while Ω_i is the rotor velocity. Furthermore, we can define the mass of the UAV as m_{UAV}, and the inertia matrix around the UAV's center of gravity (CoG) as $\boldsymbol{I}_{UAV} \in \mathbb{R}^{3\times3}$. As the UAV is considered to carry a payload, we define the payload mass as m_p and its inertia matrix $\boldsymbol{I}_p$. The payload is considered to be rigidly attached to the body of the UAV, with some displacement $\boldsymbol{r}_p$ from its CoG. The inertia of the system changes because of this displacement. Using the parallel axis theorem, we can write the inertia of the payload around the UAV's center of gravity:

$$\boldsymbol{I}_p^B = \boldsymbol{I}_p + m_p\left(\boldsymbol{r}_p^T \cdot \boldsymbol{r}_p \cdot \boldsymbol{E}_{3\times3} - \boldsymbol{r}_p \cdot \boldsymbol{r}_p^T\right), \tag{2}$$

where $\boldsymbol{E}_{3\times3} \in \mathbb{R}^{3\times3}$ is the identity matrix. The inertia of the whole UAV-payload system is $\boldsymbol{I}_s = \boldsymbol{I}_{UAV} + \boldsymbol{I}_p^B$, and the total mass is $m_s = m_{UAV} + m_p$.

To control the four degrees of freedom from $\boldsymbol{q}_B$, we take inspiration from the work presented in [29] and generalize the approach. The control inputs can be written as $\boldsymbol{u} = \boldsymbol{K} \cdot diag(\boldsymbol{\Omega}) \cdot \boldsymbol{\Omega}$, where $\boldsymbol{u} \in \mathbb{R}^4$ consists of roll u_1, pitch u_2, yaw u_3 and thrust u_4 inputs, $\boldsymbol{\Omega} = \left[\Omega_1, \Omega_2 \cdots \Omega_{n_p}\right]^T$ is the vector of rotors' angular velocities, and $\boldsymbol{K} \in \mathbb{R}^{4\times n_p}$ is the mapping matrix that depends on the configuration of the multirotor. Note that by multiplying $diag(\boldsymbol{\Omega}) \cdot \boldsymbol{\Omega}$, the standard quadratic relation between the control inputs and the rotors' velocities is obtained, as shown in [29]. Taking the total inertia of the system $\boldsymbol{I}_s$ and the input vector $\boldsymbol{u}$ into account, we can write dynamics equations:

$$\begin{aligned} \boldsymbol{I}_s \cdot \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times \boldsymbol{I}_s \cdot \boldsymbol{\omega} &= \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T \\ m\ddot{\boldsymbol{p}} &= \boldsymbol{R}\begin{bmatrix} 0 & 0 & u_4 \end{bmatrix}^T - m\boldsymbol{g}, \end{aligned} \tag{3}$$

where $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$ is the angular velocity vector in the world frame, $\ddot{\boldsymbol{p}}$ is the linear acceleration in the world frame, $\boldsymbol{g} = \begin{bmatrix}0 & 0 & g\end{bmatrix}^T$ is the gravity vector with the gravitational constant $g = 9.81m/s^2$, and $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix from the body frame to the inertial frame.

Having a payload attached to the body of the UAV affects the overall stability of the system. Researchers in [30] investigate the influence of a payload attached to a small multirotor vehicle. Namely, they analyze the linearized system dynamics in hovering conditions, under PID control rule, using the Routh-Hurwitz criterion. We were able to apply the methodology developed in that paper since we use the cascade PID control structure. In our case, the inner loop controls the velocity and the outer loop controls the position. More details about the control structure can be found in our previous work [31]. Following the aforementioned methodology, we obtain the boundary values of the payload mass m_p for which the system is stable. Since the UAV's intended mission is dropping the payload, this sudden change in the overall mass acts as a change of the system state. According to the hybrid systems theory, we can guarantee that a system is stable if all states of the system are stable (in our case UAV with and without payload) and the switching between states is such that the system is given enough time to stabilize between switches [32]. As the only switch between the states occurs when the payload is released, the hybrid system is stable.

III. AIRDROP TRAJECTORY PLANNING

Within this section we describe the planning procedure for the parabolic airdrop trajectory. We consider the payload to be rigidly attached to the body of the UAV. Furthermore, we assume the payload can be detached instantaneously when triggered. The detached projectile is considered to follow a parabolic trajectory to the provided target in the environment. The planner goes through three stages before the collision-free trajectory is generated: based on the provided target,

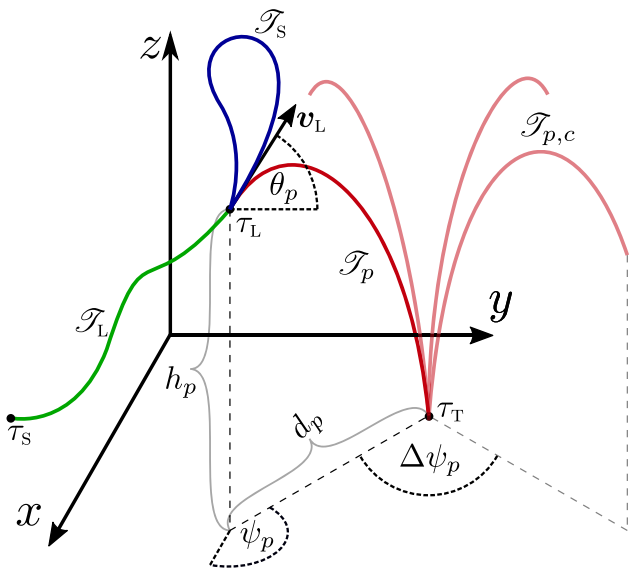

FIGURE 3. An example of parabolic projectile trajectory $\mathscr{T}_p$ chosen among candidates $\mathscr{T}_{p,c}$. The launch trajectory $\mathscr{T}_L$ and the stopping trajectory $\mathscr{T}_S$ are executed by the UAV.

a parabolic projectile trajectory candidate is calculated defining the launch point; next, an obstacle free path is planned to the launch point; and finally, a collision-free trajectory is planned based on the previously obtained path.

A. PARABOLIC FREE-FALL TRAJECTORY

When a certain projectile motion is observed, it can be described as a parabolic trajectory. While free falling, the air resistance force acts on the projectile, and the final trajectory can be described as ballistic, rather than parabolic. However, in this paper the speed of the projectile is low and the air resistance can safely be neglected, which is explained in detail in Section III-F, so the final trajectory can be observed as parabolic without any significant impact on the target point precision. The parabolic trajectory can be characterized with an initial velocity vector and a launch position, which allows obtaining the projectile position with respect to time. Within this section, the goal is to solve an inverse problem: to determine a set of suitable launch points with some initial velocities based on the user-supplied target point. An example can be observed in a manifold of such trajectories, Fig. 3, depicted with bold red. To start, we derive the parabolic trajectory equations as a manifold in three dimensions:

$$\begin{aligned} x_p(t) &= x_L + v_{0,p} \cdot cos(\theta_p)cos(\psi_p) \cdot t \\ y_p(t) &= y_L + v_{0,p} \cdot cos(\theta_p)sin(\psi_p) \cdot t \\ z_p(t) &= z_L + v_{0,p} \cdot sin(\theta_p) \cdot t - \frac{g \cdot t^2}{2}, \end{aligned} \tag{4}$$

where $\tau_L \equiv (x_L, y_L, z_L)$ is the launch point in the world frame, $v_{0,p}$ is the initial velocity magnitude of the projectile, θ_p is the launch angle, ψ_p is the direction angle, g denotes gravity magnitude along the z_W axis and t is time. After the release, only the gravitational force acts on the projectile, giving it the characteristic parabolic trajectory shape.

In the airdrop use case, we consider the target point $\tau_T \equiv (x_T, y_T, z_T)$ as an input. Based on the target, we search for a suitable launch point τ_L candidate within the manifold (4), as shown on Fig. 3. First we can define the parabolic trajectory configuration vector as:

$$\boldsymbol{C}_p = \begin{bmatrix} d_p & h_p & v_{0,p} & \theta_p & \psi_p \end{bmatrix}^T, \tag{5}$$

where d_p and h_p are horizontal and vertical displacements. When performing a parabolic airdrop, the UAV must reach the launch point with certain velocity so that the payload would execute a parabolic motion. Given the aforementioned configuration vector $\boldsymbol{C}_p$ and the target point τ_T, we can write coordinates of the launch point as follows:

$$\begin{aligned} x_L &= x_T - d_p \cdot cos(\theta_p)cos(\psi_p) \\ y_L &= y_T - d_p \cdot cos(\theta_p)sin(\psi_p) \\ z_L &= z_T + h_p, \end{aligned} \tag{6}$$

and velocity vector:

$$\boldsymbol{v}_L = \begin{bmatrix} v_{0,p} \cdot cos(\theta_p)cos(\psi_p) \\ v_{0,p} \cdot cos(\theta_p)sin(\psi_p) \\ v_{0,p} \cdot sin(\theta_p) \end{bmatrix}. \tag{7}$$

For simplicity and brevity we define the launch point configuration vector as $\boldsymbol{C}_L = \begin{bmatrix} \tau_L^T & \boldsymbol{v}_L^T \end{bmatrix}^T$ which consists of a point in space the UAV has to reach for launching the projectile, alongside a velocity the UAV must ensure for the planned launch. Note that the launch velocity $\boldsymbol{v}_L$ is closely tied with the launch angle θ_p. Namely, θ_p is the angle between the horizontal and vertical components of the launch velocity. This way, we can achieve a certain launch angle through the UAV control by reaching the specified velocity and releasing the payload at that time.

At this point, the unknown parameters in the equation (4) are time $t = T_p$ when the projectile reaches the target point and the height displacement $h_p = z_L - z_p(t = T_p)$. Given some distance from the target d_p, speed $v_{0,p}$ and launch angle θ_p one can calculate unknown parameters with:

$$\begin{aligned} T_p &= \frac{d_p}{v_{0,p} \cdot cos(\theta_p)} \\ h_p &= v_{0,p} \cdot sin(\theta_p) \cdot T_p - \frac{gT_p^2}{2}, \end{aligned} \tag{8}$$

which allows deriving the launch point τ_L using equation (4).

Recalling the parabolic trajectory configuration vector $\boldsymbol{C}_P$ from equation (5), we can restrict its members to some intervals. Namely, $d_p \in (d_p^{min}, d_p^{max})$, $v_{0,p} \in (v_{0,p}^{min}, v_{0,p}^{max})$ and $\theta_p \in (\theta_p^{min}, \theta_p^{max})$ are user defined intervals. The direction angle interval is constrained to be within $\psi_p \in (0, 2\pi)$ which means the parabola candidates are considered in all directions. Using equation (8), and based on the aforementioned constraints, the interval for h_p can be obtained. We can further discretize these intervals with Δd_p, $\Delta v_{0,p}$, $\Delta\theta_p$ and $\Delta\psi_p$ as steps for each of the search space variables. By doing so, we obtain

a discrete search space with a finite number of parabolic trajectory candidates $\mathscr{T}_{p,c}$. Each candidate has its own launch point configuration vector $\boldsymbol{C}_\mathrm{L}$ and all possible configurations can be written as a set:

$$\mathscr{L}_p = \left\{ C_{\mathrm{L},c} \mid \begin{matrix} d_p \in (d_p^{min}, d_p^{max}), & v_{0,p} \in (v_{0,p}^{min}, v_{0,p}^{max}) \\ \theta_p \in (\theta_p^{min}, \theta_p^{max}), & \psi_p \in (0, 2\pi) \end{matrix} \right\} \quad (9)$$

B. PATH PLANNING

The main task of the path planner is to find a piecewise straight line obstacle-free path given the desired waypoints. We consider the environment to be known, either given through a known map or obtained by an exploration or mapping algorithm, i.e. as proposed in [33]. In order to find an obstacle-free path in the environment, we employ the well-known RRT* algorithm [34], although other path planners can be used as well. Even though the UAV indeed has six degrees of freedom (DoF), it is an underactuated system which means we cannot plan for all six DoF simultaneously. Therefore, we reduce the planning space to four degrees of freedom as $\boldsymbol{q}_\mathrm{B} = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T \in \mathbb{R}^4$.

As shown on Fig. 3, the UAV has to move from its starting point $\tau_\mathrm{S} \equiv (x_\mathrm{S}, y_\mathrm{S}, z_\mathrm{S})$, which is the current UAV position, to the launch point τ_L. Since the orientation is not relevant for a successful launch, we set it to the parabola orientation $\psi_\mathrm{L} = \psi_p$. The start waypoint $\boldsymbol{w}_\mathrm{S} = \begin{bmatrix} x_\mathrm{S} & y_\mathrm{S} & z_\mathrm{S} & \psi_\mathrm{S} \end{bmatrix}^T \in \mathbb{R}^4$ and end waypoint $\boldsymbol{w}_\mathrm{L} = \begin{bmatrix} x_\mathrm{L} & y_\mathrm{L} & z_\mathrm{L} & \psi_\mathrm{L} \end{bmatrix}^T \in \mathbb{R}^4$ are the input to the path planning algorithm. The output is obstacle-free piecewise straight path between waypoints:

$$\mathscr{P} = \left\{ \boldsymbol{p}_i \mid \boldsymbol{p}_i \in \mathbb{R}^4, i \in (0, 1, \ldots, m-1) \right\}, \quad (10)$$

where $m \geq 2$ is the number of points in the path, and $\boldsymbol{p}_i = \begin{bmatrix} x_i & y_i & z_i & \psi_i \end{bmatrix}^T \in \mathbb{R}^4$ denotes the i-th point on the path. The launch configuration $\boldsymbol{C}_\mathrm{L}$ requires that both position and velocity are satisfied at the launch point. However, the RRT* does not account for dynamics and can only ensure launch position is reached. To satisfy the velocity condition, we plan a trajectory based on the path $\mathscr{P}$, obtained by the planner described in the following section.

C. TOPP-RA TRAJECTORY

To plan dynamically feasible trajectories, we use Time Optimal Path Parameterization by Reachability Analysis (TOPP-RA) approach developed within [25]. The approach works on an n-dimensional problem, given the velocity and acceleration constraints of each DoF. Since it is based on a numerical integration approach, the original practical implementation runs very fast while providing time optimal trajectories. The input to the trajectory planning is the path $\mathscr{P}$ defined in equation (10). The output is a trajectory:

$$\mathscr{T}_\mathrm{T} = \left\{ \boldsymbol{x}(t) \mid \boldsymbol{x}(t) \in \mathbb{R}^{3\cdot 4}, t \in (0, t_{end}) \right\}, \quad (11)$$

where $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q}_\mathrm{B}^T & \dot{\boldsymbol{q}}_\mathrm{B}^T & \ddot{\boldsymbol{q}}_\mathrm{B}^T \end{bmatrix}^T$ is one trajectory point with position, velocity and acceleration in the generalized coordinates of the UAV, t is time, and t_{end} the trajectory duration.

The trajectory $\mathscr{T}_\mathrm{T}$ is a stop-to-stop trajectory at this point. However, one of the requirements from Section III-A is the velocity at the launch point. Although the TOPP-RA approach allows for non-zero velocities at both start and end point, the widely accepted practical implementation struggles to find a feasible solution in such cases. Through our empirical validation, we discovered that run time of the algorithm increases with non-zero velocities. Furthermore, the solution is usually found if provided velocities are close to zero, however, when an arbitrary velocity was set, the implementation failed to find a solution in most cases. The reasons why the aforementioned planner fails goes beyond the scope of this paper.

Although the TOPP-RA fails when arbitrary non-zero velocity and acceleration are set at the start or at the end, it is still a reliable planner. TOPP-RA plans time optimal trajectories while respecting the given constraints, and does this in a very short time period for both small and large environments. Throughout the extensive testing in simulation and experimental environments, the planner never failed to produce a feasible stop-to-stop trajectory. Furthermore, in our analysis of the planner so far, it never failed on high-dimensional problems, i.e. 22 DoF. In such cases, the planning time only slightly increased when compared to the UAV-only planning with four DoF. One disadvantage of the TOPP-RA when compared to the convex optimization methods is the non-continuity of the high-order derivatives, such as jerk or snap. Methods from [27], [29] work well with a small number of points in the path ($n < 15$) and with a limited number of DoF. As we use large environments in following sections, the path planning produces a large number of waypoints which imposes a significant planning time for the aforementioned methods. This kind of reliability and a fast planning time for a large number of waypoints prompts us to still use the TOPP-RA for the initial trajectory planning, and achieving non-zero velocity and acceleration at the launch point is described in the following section.

D. SPLINE INTERPOLATION

The aforementioned behavior prompted us to augment the stop-to-stop trajectory generated by TOPP-RA with a 5^{th} order spline. As this spline has six free coefficients, we are able to define the position, velocity and acceleration at both start and the end of the spline. Since the original trajectory also contains position, velocity and acceleration, this allows us to replan any segment of the original trajectory with such a spline. We start with the 5^{th} order spline general equation:

$$p(t) = \sum_{i=0}^{5} a_i t^i, \quad (12)$$

where a_i are free coefficients of the spline. We can also define a spline configuration vector:

$$\boldsymbol{C}_s = \begin{bmatrix} p_s & v_s & a_s & p_e & v_e & a_e \end{bmatrix}^T,$$

where subscript s denotes the start point and subscript e denotes the end point. Free coefficients a_i can be uniquely calculated with:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} p_s & 0 & 0 & 0 & 0 & 0 \\ 0 & v_s & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{a_s}{2} & 0 & 0 & 0 \\ \frac{-10p_0}{T^3} & \frac{10p_e}{T^3} & \frac{-6v_s}{T^2} & \frac{-4v_e}{T^2} & \frac{-3a_s}{2T} & \frac{a_e}{2T} \\ \frac{15p_0}{T^4} & \frac{-15p_e}{T^4} & \frac{8v_s}{T^3} & \frac{7v_e}{T^3} & \frac{3a_s}{2T^2} & \frac{-a_e}{T^2} \\ \frac{-6p_0}{T^4} & \frac{6p_e}{T^4} & \frac{-3v_s}{T^3} & \frac{-3v_e}{T^3} & \frac{-a_s}{2T^2} & \frac{a_e}{T^2} \end{bmatrix} \cdot \boldsymbol{C}_s, \tag{13}$$

where T denotes the spline duration. We also impose dynamic constraints on the spline as the maximum velocity magnitude v_{max} and the maximum acceleration magnitude a_{max}. Since the 5^{th} order spline also has duration as an unknown parameter, we perform an initial guess of the duration with $T_0 = |p_e - p_s| / v_{max}$. Next, we use the equations (12) and (13) to find maximum velocity magnitude $v_{max,s}$ and acceleration magnitude $a_{max,s}$ on this particular trajectory. We optimize the spline duration T in an iterative manner with:

$$\begin{aligned} s &= \max\left\{\frac{v_{max,s}}{v_{max}}, \sqrt{\frac{a_{max,s}}{a_{max}}}\right\} \\ T_{i+1} &= T_i \cdot (1 + sgn(s-1) \cdot \alpha), \end{aligned} \tag{14}$$

where α is the convergence factor and i is the current iteration starting from $i = 0$. The optimization is over when the derivative ratio factor is within user defined bounds $-\epsilon < s - 1 < \epsilon$, or the spline duration T does not exceed a user defined threshold. In our experience, the time threshold condition has never been triggered and the optimization always converged within the bound ϵ. This procedure ensures the spline meets at least one dynamical constraint, either velocity or acceleration.

To account for multiple DoFs, we have to extend this procedure to an n-dimensional problem. As all DoFs have to be time synchronized, the duration is simply set to the maximum duration of all DoFs, $T_i = \max\{T_{i,1}, T_{i,2} \cdots T_{i,n}\}$. The derivative ratio factor is the maximum of all DoFs' factors, $s = \max\{s_1, s_2 \cdots s_n\}$. This allows us to produce an n-dimensional spline that is within the dynamic constraints of all DoFs. We employ the described method for planning both launch and stopping trajectories.

1) LAUNCH TRAJECTORY

As the initial trajectory $\mathscr{T}_\mathrm{T}$ from equation (11) plans stop-to-stop motion, we need to ensure the required velocity at the launch point. Moving backwards along the initial trajectory, we can replace a portion of the initial trajectory with a spline trajectory that satisfies the launch velocity. Fig. 4 depicts how the launch trajectory is chosen.

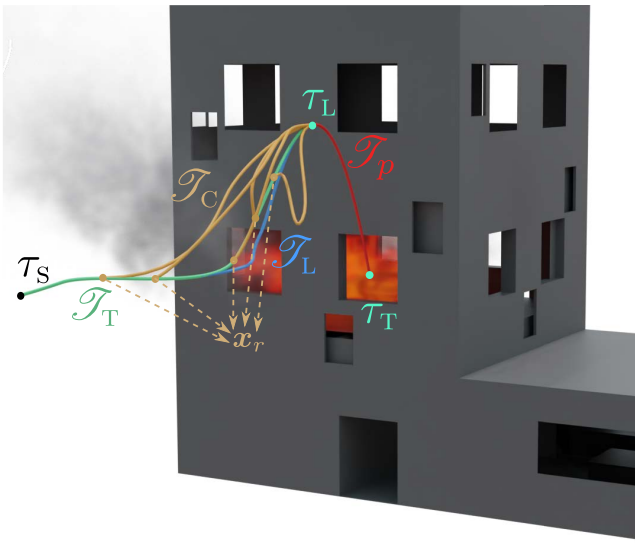

FIGURE 4. An illustrative example of replacing a portion of the initial trajectory $\mathscr{T}_\mathrm{T}$ with a launch spline trajectory $\mathscr{T}_\mathrm{L}$ chosen between candidates $\mathscr{T}_\mathbf{C}$. Points denoted as x_r represent the candidate spline starting points. After supplying the target point τ_t, the launch point τ_l is determined based on the payload parabolic trajectory $\mathscr{T}_\boldsymbol{p}$. The trajectory is then planned from the starting point τ_s which is the current position of the UAV.

Moving backwards along the initial trajectory $\mathscr{T}_\mathrm{T}$, we calculate its cartesian length using a line integral formula:

$$l_\mathrm{T} = \oint_{T_1}^{T_2} \sqrt{x^2(t) + y^2(t) + z^2(t)}dt, \tag{15}$$

where we integrate between T_1 and T_2. Every Δl_T meters, we take a point $\boldsymbol{x}_r$ on the initial trajectory and plan a launch trajectory candidate $\mathscr{T}_c$ from that point to the launch point $\boldsymbol{x}_\mathrm{L} = \begin{bmatrix}\boldsymbol{q}_\mathrm{L}^T & \dot{\boldsymbol{q}}_\mathrm{L}^T & \ddot{\boldsymbol{q}}_\mathrm{L}^T\end{bmatrix}^T$ where $\boldsymbol{q}_\mathrm{L} = \begin{bmatrix}\tau_\mathrm{L}^T & \psi_p\end{bmatrix}^T$, $\dot{\boldsymbol{q}}_\mathrm{L} = \begin{bmatrix}\boldsymbol{v}_\mathrm{L}^T & 0\end{bmatrix}^T$ and $\ddot{\boldsymbol{q}}_\mathrm{L} = \boldsymbol{0}_{4\times1}$. Note that in this particular case, T_2 is the trajectory end time and T_1 is the time when we reach the point $\boldsymbol{x}_r$. Using the procedure from Section III-D, we plan a candidate launch trajectory from $\boldsymbol{x}_r$ to $\boldsymbol{x}_\mathrm{L}$, which is immediately checked for collision. If the trajectory is collision-free, the candidate becomes a part of the launch candidates set $\mathscr{S}_\mathrm{L}$. Integrating backwards continues until the user specified spline length limit L, or the end of the initial trajectory is reached.

Let l_c be the length of each launch trajectory candidate $\mathscr{T}_c$, obtained through equation (15), and l_i be the length of the initial trajectory portion $\mathscr{T}_\mathrm{T}$ that the candidate trajectory replaces. The criterion for selecting the most appropriate launch trajectory is a length ratio:

$$r_l = l_c / l_i \tag{16}$$

The candidate with the ratio closest to one is chosen as the launch spline that is incorporated into the initial trajectory. A careful reader should note that this ratio is not monotonically decreasing or increasing because it depends on the selected launch point configuration $\boldsymbol{C}_\mathrm{L}$. The resulting trajectory is the launch trajectory $\mathscr{T}_\mathrm{L}$. If there is no feasible

launch trajectory, the launch point candidate is discarded and the process is repeated with a new candidate.

2) STOPPING TRAJECTORY

As the UAV reaches the launch point with some non-zero velocity, it will execute some stopping motion. If neglected, this motion could potentially hit an obstacle, which is not a desired behavior. To account for that, we plan a stopping trajectory $\mathscr{T}_\mathrm{S}$ between the launch point $\boldsymbol{x}_\mathrm{L}$ and the resting point $\boldsymbol{x}_\mathrm{R} = \left[\boldsymbol{q}_\mathrm{L}^T\ \mathbf{0}_{1\times4}\ \mathbf{0}_{1\times4}\right]^T$, using the spline trajectory described in the previous section. Unlike the launch trajectory, we plan the stopping trajectory for each axis separately. This is possible because we do not require all degrees of freedom reaching the resting point simultaneously. In turn, this slightly different approach yields a more aggressive trajectory since all degrees of freedom will reach a dynamic constraint. Thus, the final trajectory path length will be shorter which shortens the stopping motion.

One can observe that the resting point position is the launch position, while velocity and acceleration at the end are set to null vectors to ensure hovering. After the trajectory is planned, it is checked for collision and the launch point is dismissed if the stopping trajectory is not collision-free.

Algorithm 1: Path and Trajectory Planning Overview

```
PlanAirdropTrajectory (𝓛_p, q_B):
    inputs : Launch configuration set 𝓛_p,
             UAV pose q_B
    output: Trajectory 𝒯
    forall C_L ∈ 𝓛_p do
        /* Plan path, equation (10)      */
        𝒫 = planPath(C_L, q_B);
        /* Plan trajectory, equation (11)
           */
        𝒯_T = planToppraTrajectory(𝒫);
        /* Section III-D1                */
        𝒯_L = planLaunchTrajectory(𝒯_T, v_L);
        /* Section III-D2                */
        𝒯_S = planStoppingTrajectory(𝒯_L);
        𝒯 = concatenateTrajectories(𝒯_L, 𝒯_S);
        if isCollisionFree(𝒯) then
            return 𝒯;
        else
            continue searching;
        end
    end
end
```

E. PLANNING OVERVIEW

To summarize, we give a brief overview of the whole planning procedure described within previous subsections. First we obtain a finite set launch point candidates $\mathscr{L}_p$, which provides points in the world frame together with the velocity for the parabolic airdrop. Iterating through $\mathscr{L}_p$, we check whether the parabolic trajectory for the current candidate is collision-free, if so we plan a collision-free path to that particular point. The trajectory is planned in three phases described in previous text. First, a stop-to-stop TOPP-RA trajectory $\mathscr{T}_\mathrm{T}$ is planned based on path $\mathscr{P}$ obtained in section III-B. Next, a portion of this trajectory is replaced with the launch trajectory $\mathscr{T}_\mathrm{L}$ in order to satisfy velocity and acceleration constraints at the launch point. Finally, a stopping trajectory $\mathscr{T}_\mathrm{S}$ is concatenated to the launch trajectory to account for the UAV stopping motion. Naturally, all the aforementioned trajectories are collision-free and are discarded otherwise. The aforementioned search is performed sequentially, iterating through $\mathscr{L}_p$. As soon as the first collision-free trajectory $\mathscr{T}$ is found, the search is finished and other candidates are not evaluated. The full planning procedure is captured within Algorithm 1.

1) IMPLEMENTATION NOTE

Throughout this and previous sections we refer to the collision-free path and trajectory. In this paper, the *OctoMap* [35] representation of the environment is used. The *OctoMap* consists of voxels that can be occupied or free, and it is essentially a spatial discretization of the environment with various resolutions. It offers a simple and fast interface to check the voxel state. The UAV is represented with a fixed size bounding box when checking for collisions. The dimensions of the bounding box are larger than the UAV itself to account for safety margins in the planning procedure. The bounding box is then spatially discretized and each point obtained in such a way is checked for the collision. If any point is indeed colliding with the environment, the given UAV configuration is not feasible. The collision checking is performed within the RRT* path planning algorithm to produce collision-free piecewise-straight line paths, and it is also performed on the planned trajectory since it can deviate from the underlying path.

2) COMPUTATIONAL COMPLEXITY DISCUSSION

As we already mentioned and summarized in Algorithm 1, there are several aspects of the planning procedure, each with its own computational complexity. The procedure runs an exhaustive search iterating through $\boldsymbol{C}_\mathrm{l}$ and it terminates after the first collision-free trajectory is found. In each iteration, four algorithms with different computational complexities are executed:

1) RRT* path planning. According to [36], the computational complexity of the RRT* algorithm is $O(n \log n)$, where n denotes the number of samples for a fixed environment. For more open and sparsely populated environments, such as the city example in Section IV-B, the number of samples is relatively small. For more complex environments, such as the office example in Section IV-C, the number of samples is greater due to narrow passages.
2) TOPP-RA trajectory planning. In [25], researchers report the computational complexity of $O(mN)$, where

m denotes the number of inequalities that have to be solved at each of N discretization points. As the underlying path gets longer, the number of discretization points is higher so longer trajectories tend to have higher computational time.

3) Stopping trajectory. The spline planning algorithm is discussed in Section III-D and is an iterative subgradient method. This class of optimization methods is similar to the steepest descent methods, however, it can operate on a non-differentiable objective function as the one used in this paper. Time required to solve the problem depends on the initial and final dynamical conditions. For the stopping spline, it will depend on the launch point configuration. The computational complexity of the subgradient method is an open research problem in computer science, and therefore goes beyond the scope of this paper.
4) Launch trajectory. The same procedure as for the stopping spline is used to plan the launch spline. The difference lies in calling the spline planning method multiple times as candidate points $\boldsymbol{x}_r$ are determined on the TOPP-RA trajectory. Indirectly, the method depends on the environment configuration since a more complex environment will produce a more challenging TOPP-RA trajectory, and TOPP-RA trajectory dictates the initial conditions.

The computation time for each part of the planning algorithm is given in Section VI.

F. NEGLECTING AIR RESISTANCE

To treat the payload trajectory as parabolic rather than ballistic, we provide an analysis on how much air resistance influences the trajectory. In this paper, we consider releasing a spherical payload. The drag force of a sphere is given with:

$$F_d = \frac{1}{2}\rho_{air} v^2 C_d\,(Re\,(v, r))\,A, \tag{17}$$

where $\rho_{air} = 1.1839\ kg/m^3$ is the air density, v is the projectile speed, C_d is the drag coefficient of the sphere that depends on the Reynolds number Re, r is the radius of the sphere, and $A = r^2\pi$ is the sphere projection area perpendicular to the velocity direction.

The Reynolds number depends on the speed and the radius of the sphere, and can be expressed as:

$$Re(v, r) = \frac{\rho_{air} v r}{3\mu_{air}}, \tag{18}$$

where $\mu_{air} = 1.837 \cdot 10^{-5} Ns/m^2$ is the dynamic air viscosity. As the drag coefficient C_d depends on the Reynolds number, we use the equation 7 of the work in [37] to obtain the drag coefficient.

Based on system and task specific limitations, we consider the following constraints: the initial horizontal velocity magnitude of the released object is $v_h < 5m/s$; the maximum height between the release point and the target is $\Delta z < 10m$; the radius of the object is within interval $0.04m < r < 0.25m$; and the mass of the object is within the interval $0.3kg < m_p < 3kg$. Furthermore, if the object is released horizontally, the maximum speed will be achieved at the target. The maximum speed can be obtained with $v_{max} = \sqrt{v_h^2 + 2g\Delta z}$, where g denotes gravitational acceleration. Plugging all the above assumptions in equations (17) and (18), we obtain the interval for the Reynolds number of $1278 < Re < 7987$ and finally for the maximum drag force $0.0007N < F_d < 0.0147N$.

As the drag force of the sphere depends on the speed, it will increase as the object accelerates. To be on the safe side, we can assume the worst case scenario in which the force along the whole path is constant and equals the maximum drag force at the impact. Taking the mass of the payload into account and relying on the fact that the path traversed under constant acceleration equals $\Delta s = 0.5 \cdot \frac{F_d}{m_p} t^2$, we obtain the maximum deviation of the object with respect to the target of $\Delta s_{max} = 0.005m$. The obtained maximum deviation is negligible, especially considering the fact that the assumptions that led to the maximum deviation are favoring the worst case scenario. Therefore, we can safely neglect the air resistance when accounting for the ballistic free fall trajectory.

IV. SIMULATION

To examine and validate the proposed parabolic trajectory planning approach, we created a realistic simulation environment in the Gazebo simulator within the Robotic Operating System(ROS). We model the UAV as a single rigid body with rotational joints attached to each of the UAV's arms. Rotational joints are simulating the propeller dynamics through *rotors_simulator* [38], a widely used public package. The projectile model is a simple ball with mass and radius as parameters. To pick up and release the projectile, we use the *storm_gazebo_ros_magnet* [39] package that simulates a permanent dipole magnet. In order to release the projectile, we adapted the package to function as an electromagnet with a ROS topic for toggling it on and off. The full implementation can be found at https://github.com/larics/storm_gazebo_ros_magnet/tree/melodic_electromagnet_dev.

We validate this approach in several distinct environments in order to test all aspects of the planning algorithm.

A. PARABOLIC TRAJECTORY ANALYSIS

In this subsection we consider planning a trajectory in an obstacle free environment. This allows us to concentrate on the planned and the executed parabolic trajectory of the projectile. The ball radius is set to $r_b = 0.1m$ and the target position is $\tau_T \equiv (0, 0, 0.1)$. The target z axis coordinate is set to the ball radius because the measurement of the ball's position is obtained at its center. To analyze this behavior, we planned trajectories for $n = 150$ parabola configuration vectors $\boldsymbol{C}_p$. These configurations were obtained by all combinations of the height displacement $h_p \in \{1.0, 1.5, \ldots, 4.5\}\,m$, the initial velocity magnitude $v_{0,p} \in \{1.5, 2.0, \ldots, 3.5\}\,m/s$

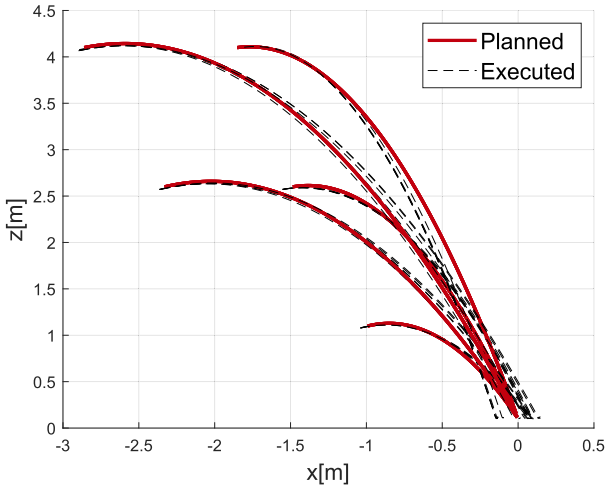

FIGURE 5. An example of various planned parabolic trajectories $\mathscr{T}_p$ and executed $\mathscr{T}_{p,e}$.

and the launch angle $\theta_p \in \{0, 7, 14, 21\}°$. The distance from target d_p is calculated using equation (8), and the parabola yaw angle is kept at zero for simplicity. Each trajectory was then executed in 10 trials to obtain the average Root Mean Square Error (RMSE) and distance to the target values. Some examples of such trajectories can be observed in Fig. 5. Note that to better visualize the experiment, we constrained the planner to the $x - z$ plane. The overall success rate in an obstacle-free environment was $1104/1500$. Note that the success here is determined the same as in Section IV-C, all trials with the impact point within $d_t = 0.375m$ are considered to be successful.

To calculate the average RMSE between the planned and executed parabolic trajectory, we use the Hausdorff distance from each point on the planned trajectory to the executed trajectory. Using this method, the obtained error averages at $RMSE_p = 0.2046m$. The other criterion we considered was the distance from the target. The distance from the target averages at $d_{avg} = 0.2717m$ with the maximum of $d_{max} = 0.9065m$ and the median value of $d_{med} = 0.2481m$. This is a relatively small error which allows us to consistently perform a very precise parabolic airdrop.

Apart from analyzing the RMSE of the projectile trajectory, we also analyzed the executed trajectory performance. Throughout all the experiments, the velocity and acceleration constraints were kept the same, given within Table 2. The table reflects constraints for all three stages of the trajectory planning: the approach trajectory $\mathscr{T}_\mathrm{T}$ planned with TOPP-RA; the launch trajectory spline $\mathscr{T}_\mathrm{L}$; and the stopping trajectory spline $\mathscr{T}_\mathrm{S}$. Consequently, we analyze the RMSE of all three trajectory stages separately. We calculated the average RMSE for all conducted airdrops at $RMSE_\mathrm{T} = 0.1299m$, $RMSE_\mathrm{L} = 0.1123m$, $RMSE_\mathrm{S} = 0.0795m$. The RMSE difference between the three stages of the trajectory occurs due to different planning approaches and constraints imposed upon the trajectory. Note that the stopping trajectory has slightly higher constraints in order to stop the UAV more aggressively. The obtained RMSE indicates the UAV follows the trajectory in a precise manner, which in turn ensures the precise parabolic airdrop.

TABLE 2. Velocity and acceleration constraints for all three segments of the planned trajectory. $\mathscr{T}_\mathrm{T}$ denotes the approach trajectory, planned through TOPP-RA. $\mathscr{T}_\mathrm{L}$ denotes the launch spline and $\mathscr{T}_\mathrm{S}$ denotes the stopping spline. All values are in standard SI units: $v[m/s]$, $a[m/s^2]$, $\omega[rad/s]$ and $\dot{\omega}[rad/s^2]$.

	v_x	v_y	v_z	ω_z	a_x	a_y	a_z	$\dot{\omega}_z$
$\mathscr{T}_\mathrm{T}$	2.0	2.0	1.5	1.0	1.2	1.2	0.8	1.0
$\mathscr{T}_\mathrm{L}$	5.0	5.0	3.0	1.0	2.5	2.5	1.0	1.0
$\mathscr{T}_\mathrm{S}$	8.0	8.0	3.0	1.0	3.0	3.0	1.5	1.0

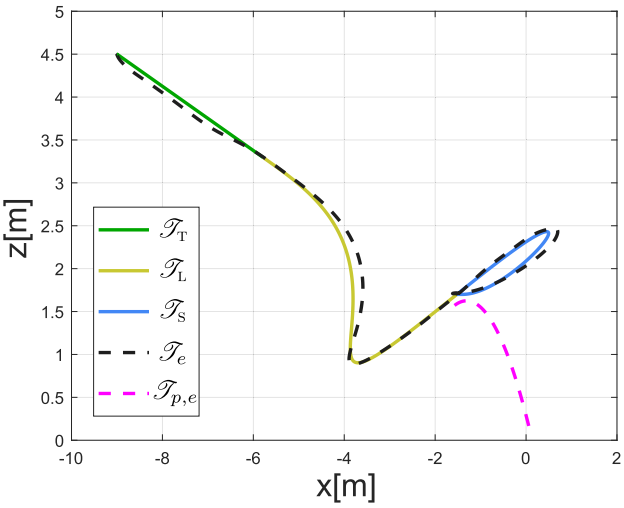

FIGURE 6. Planned and executed trajectories of the UAV for a parabolic airdrop task. The projectile trajectory, given in magenta, is depicted for clarity.

B. LARGE-SCALE ENVIRONMENT

After the consistent parabolic airdrop was achieved, we turned to planning in a large scale city-like environment. The choice of such an environment is driven by the practical use case: delivering and deploying a fire extinguishing ball to a building on fire. To clarify, a fire extinguishing ball contains a dry powder which disperses in contact with the flame. Since we use *OctoMap* [35] as a map representation, we assume a known map of the city. In a real world scenario such a map can be obtained through cadastral urban maps or built through mapping procedures, however, this goes beyond the scope of this paper. The RRT* planner uses the *OctoMap* as an environment map and plans an obstacle free path as described in Section III-B. An example of the trajectory planned in the city environment is shown on Fig. 7. The city environment is scaled down for clarity, but from the perspective of the planner it still retains the same level of complexity as a full scale environment. The scaled down environment is roughly the size $100m \times 100m \times 40m$.

The start point of the UAV was set to the far side of the city in order to test the planner's ability to find a feasible trajectory

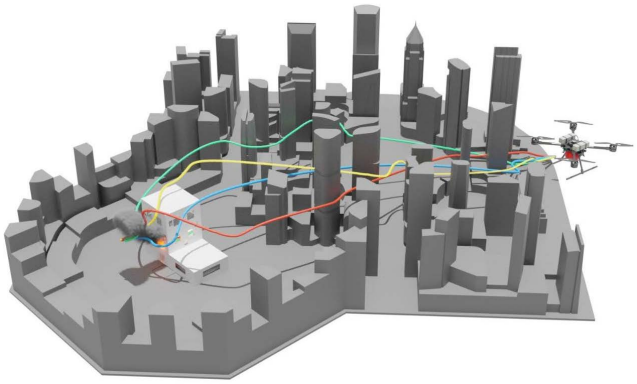

FIGURE 7. Several examples of executed trajectories in a large-scale city environment. The UAV navigates through the city, reaches the launch point and releases the ball into a building of interest.

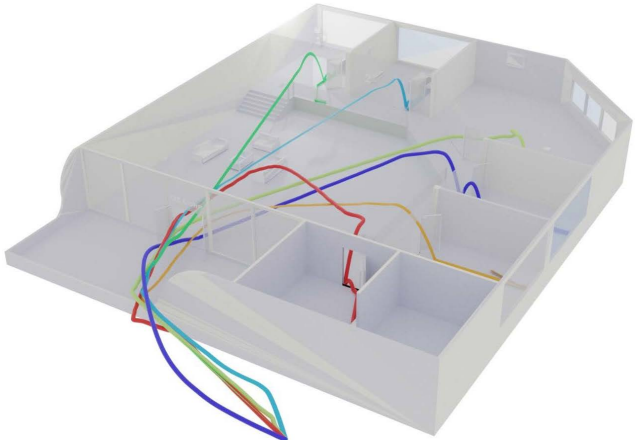

FIGURE 8. Executed ball trajectories for launching the ball into each room of the office environment.

navigating through the city. We performed 40 trials aiming to deliver the ball into the building. For consistency, we chose to deliver through the same window. The average planned trajectory length for the city environment is $l_{\mathscr{T}} = 167.79m$. The average RMSE for the city environment is very similar to the one obtained in Section IV-A, and averages at $RMSE_{\mathrm{T}} = 0.1086m$, $RMSE_{\mathrm{L}} = 0.1402m$ and $RMSE_{\mathrm{S}} = 0.0614m$. The system was able to deliver the ball through the window in 34/40 instances. However, the executed parabolic trajectory has a slightly poorer performance opposed to Section IV-A. The parabola error averaged at $RMSE_p = 0.1557m$, average distance to target is $d_{avg} = 0.2896m$ and maximum distance to target is $d_{max} = 0.3275$ and the median of $d_{med} = 0.2917m$. This kind of performance is expected since the RRT* algorithm plans a different path for each experiment, and the resulting launch trajectory is therefore different in each instance. Nevertheless, the high precision of the parabolic airdrop is evident from the high rate of successful deliveries.

C. DENSE INDOOR ENVIRONMENT

To push the limits of the method, we tested it in a dense indoor environment. Namely, we chose an office space layout with several rooms and a common place, depicted on Fig. 8. As in the city example, we assume a known map of the environment. Having such an environment also puts high requirements on the maneuverability and control of the UAV, since it has to navigate through doors and abruptly stop if the projectile was released close to an obstacle, i.e. parabolic airdrop through a door. The planner's task is to find a feasible parabolic airdrop candidate such that the UAV does not crash into obstacles while launching the projectile in a precise manner. The planner was tested in a series of experiments where the ball was delivered into each room of the office for 20 times. Note that the RRT* obtains a different path each time it is called.

Since the planned stopping trajectory was consistently not feasible due to crashing into the wall above the door of an office, we needed to take a slightly different approach for the launch and stopping trajectories. Namely, we introduce a non-zero acceleration a_{L} at the launch point τ_{L}. This acceleration is a scalar that is spanned in the x-y plane opposite to the planned direction of the ball and is given by $\ddot{\boldsymbol{q}}_{\mathrm{L}} = \left[-a_{\mathrm{L}}cos(\psi_p)\ -a_{\mathrm{L}}sin(\psi_p)\ 0\ 0\right]^{\mathrm{T}}$. Setting any acceleration at the launch point does not affect the planned parabolic trajectory because at the instance of release the only force acting on the ball is the gravity. For the office environment the launch acceleration was set to $a_{\mathrm{L}} = 1.5\ m/s^2$. This method enabled the UAV to start decelerating even before it releases the ball and the resulting stopping trajectory had no contact with the wall. Furthermore, the acceleration constraint for the stopping trajectory was slightly increased to $a_x = 4.0m/s^2$ and $a_y = 4.0m/s^2$, in order to avoid contact with the environment. The average planned trajectory length for the office environment is $l_{\mathscr{T}} = 38.35m$.

The obtained average RMSE of the executed trajectories is similar as in Section IV-A, a total of 120 trajectories (20 trials per room in the office) were performed. The resulting performance is: $RMSE_{\mathrm{T}} = 0.0894m$, $RMSE_{\mathrm{L}} = 0.1522m$, $RMSE_{\mathrm{S}} = 0.0728m$. The obtained average parabola error is $RMSE_p = 0.4167m$, the average distance to target is $d_{avg} = 0.4723m$ and the maximum distance to the target is $d_{max} = 0.9936m$, and the median of $d_{med} = 0.4389m$. We attribute this decrease in precision to the tight environment and the increase in dynamical constraints. On the other hand, in 87/120 cases the ball ended up in an office as planned. We would like to point out that in an average case of $d_{avg} = 0.4723m$ the numbers suggest that the system misses the office. However, this is not entirely true because the distance to target when the miss occurs is larger than the average. The median at $d_{med} = 0.4389m$ provides a deeper insight on the performance which complements the greater success rate than the average suggests.

To further test the method, we performed a series of experiments aiming for a bucket placed in one room of the office, totaling in 40 more trials. The radius of the placed bucket is $r_{bucket} = 37.5cm$ and the ball radius was set to $r_{ball} = 0.1m$. With such parameters the system was able to successfully throw the ball in the bucket 26/40 times.

TABLE 3. Average execution times of different planning procedure steps. All times are expressed in seconds. Note that RRT* step does not occur in an empty environment, thus, the planning time is zero.

Step \ Env	Empty	City	Office
RRT*	0	7.944	33.165
TOPP-RA	0.022	0.127	0.054
Stopping	0.066	0.123	0.091
Launch	0.749	2.611	1.969

The trajectory tracking errors averaged at $RMSE_{\mathrm{T}} = 0.0872m$, $RMSE_{\mathrm{L}} = 0.1405m$, $RMSE_{\mathrm{S}} = 0.0646m$. The obtained average parabola error is $RMSE_p = 0.2643m$, the average distance to target is $d_{avg} = 0.3076m$ and the maximum distance to the target is $d_{max} = 0.7914m$, and the median of $d_{med} = 0.2837m$.

D. ALGORITHM RUNTIME

Within section III-E, a discussion on computational complexity is given. As the planning procedure has been tested in the simulation, the planning time of each part is recorded and is shown in Table 3. The RRT* planning step does not occur in case of an empty environment, which yields the zero planning time. As the office environment is more complex than the city environment, the RRT* planning time is longer due to a greater number of steps. On the contrary, the planning time of TOPP-RA is lower in the office environment than the city. This is the direct consequence of the longer path in the city environment, which increases the number of discretization points when planning the TOPP-RA trajectory. The stopping spline planning time varies only a little between the environments. This is attributed to the fact that the planning time depends only on the initial and final conditions, which are not greatly different between environments. Lastly, the launch spline planning time is longer in office and city environments since it replaces a portion of the TOPP-RA trajectory which dictates the initial conditions. In both office and city environments, path planning induces complex trajectory shapes, which affect initial dynamic conditions of launch splines, making it challenging to optimize launch splines.

V. EXPERIMENTAL RESULTS

After analyzing the system's performance in the simulation environment, we started with the extensive experimental verification of the proposed method. The experiments were conducted using two UAVs with different characteristics in both indoor and outdoor environments. The goal in both cases was to drop a ball into a box, and we performed repeatability experiments without obstacles and a series of experiments with obstacles.

Both UAVs were equipped with the devised magnetic gripper. To release the projectile at the specific instant, we opted for an electromagnetic principle gripper with *Groove Electromagnet*. This particular electromagnet is compatible with *Arduino* boards and has the peak attraction force of $10N$, operates on $U = 5V$ voltage and $I = 400mA$ current while active. To increase the payload capability and secure the projectile in flight, we attached two electromagnets to a flat board and interfaced them with the *Arduino Nano* board. To communicate with the UAV's onboard computer, we utilized the *rosserial_arduino* library which enabled us to toggle the magnetic gripper via a ROS topic.

A. INDOOR LABORATORY ENVIRONMENT

In the indoor environment we use the *AscTec NEO* hexacopter. The dimensions of the UAV are $0.45m \times 0.45m \times 0.3m$ and the mass is $m = 2.68kg$, with maximum payload of $m_p = 1kg$. It is equipped with the *AscTec Trinity* flight controller and the *Intel NUC* onboard computer. The communication between the flight controller and the onboard computer is carried out through a serial port. Furthermore, the flight controller runs the low-level attitude control law and the onboard computer is running a high level model predictive based position control described in [40]. The NEO hexacopter is endowed with the aforementioned magnetic gripper, which is mounted $10.5cm$ below the geometric center of the UAV. The target for the indoor experiments was a box of dimensions $0.33m \times 0.28m \times 0.23m$ that was fixed to the ground.

The laboratory is equipped with the *Optitrack* motion capture system. This system is used to provide position and orientation feedback to the UAV. The static map of the environment can be observed on Fig. 9. Since the real world UAV and the laboratory environment differ from simulation, we employed a different set of constraints, as shown in Table 4.

TABLE 4. Velocity and acceleration constraints for all three segments of the planned trajectory for the *AscTec NEO* UAV. $\mathscr{T}_{\mathrm{T}}$ denotes the approach trajectory, planned through TOPP-RA. $\mathscr{T}_{\mathrm{L}}$ denotes the launch spline and $\mathscr{T}_{\mathrm{S}}$ denotes the stopping spline. All values are in standard SI units: $v[m/s]$, $a[m/s^2]$, $\omega[rad/s]$ and $\dot{\omega}[rad/s^2]$.

	v_x	v_y	v_z	ω_z	a_x	a_y	a_z	$\dot{\omega}_z$
$\mathscr{T}_{\mathrm{T}}$	2.0	2.0	1.2	1.0	0.8	0.8	0.5	0.8
$\mathscr{T}_{\mathrm{L}}$	4.0	4.0	3.0	2.0	1.0	1.0	1.0	1.0
$\mathscr{T}_{\mathrm{S}}$	5.0	5.0	3.0	2.0	2.0	2.0	1.5	1.0

1) REPEATABILITY

Within the first set of experiments we tested the parabolic airdrop over $n = 20$ straight-line trajectories with various parabola configuration vectors $\boldsymbol{C}_p$. The main point of these experiments was to inspect the repeatability of the *AscTec NEO*. The system was able to deliver the ball into the box on the floor in 16/20 instances. The trajectory tracking errors averaged at $RMSE_{\mathrm{T}} = 0.1720m$, $RMSE_{\mathrm{L}} = 0.2909m$ and $RMSE_{\mathrm{S}} = 0.1465m$. Unlike in the simulation environment, we were unable to measure the projectile position after the release instance. Nevertheless, the parabolic trajectory can be reconstructed with the position and velocity at the moment of release by employing equation (4). The duration of the parabolic trajectory can be approximated when the height of the projectile reaches the target height. Note that this reconstruction depends on the measured state of the UAV

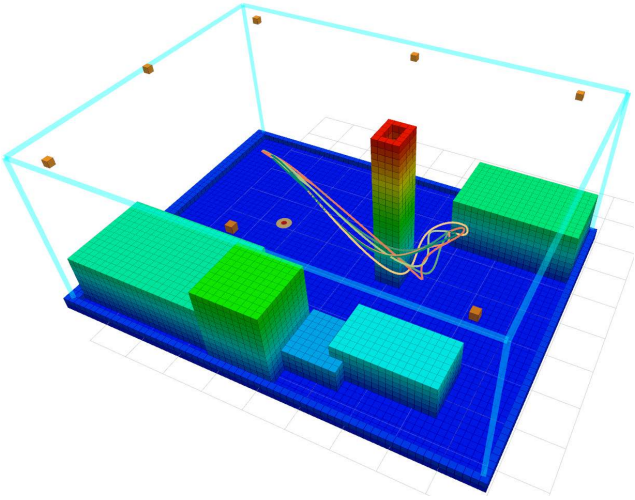

FIGURE 9. An *OctoMap* representation of the indoor environment with several planned trajectories for obstacle avoidance. The orange cubes represent positions of *Optitrack* cameras. Note that the obstacle in the middle was removed for repeatability experiments.

and can be prone to errors. On the other hand, we report that the reconstructed parabolic trajectory outcome(hit or miss) is consistent with the observed one. The error of the projectile parabolic trajectory averaged at $RMSE_p = 0.1494m$, the average distance to target is $d_{avg} = 0.1885m$, the maximum distance to target is $d_{max} = 0.3357m$ and the median of $d_{med} = 0.1880m$.

2) OBSTACLE AVOIDANCE

To test the planner's ability to avoid obstacles, we set an obstacle in the middle of our indoor lab, Fig. 9. Using the planning framework described in Section III, the objective was to navigate around the obstacle and drop the ball at the designated target. The system performed $n = 12$ trajectories with the identical starting point and various parabola configuration vectors $\boldsymbol{C}_p$. Several examples of such trajectories are depicted on Fig. 9. Note that all trajectories navigate around the left side of the obstacle. This behavior is intentional due to safety, since the operator is located on the right side of the obstacle.

The system managed to hit the box in $11/12$ instances and the overall trajectory tracking errors averaged at $RMSE_{\mathrm{T}} = 0.0751m$, $RMSE_{\mathrm{L}} = 0.1975m$ and $RMSE_{\mathrm{S}} = 0.1632m$. We reconstructed the projectile parabolic trajectory with the same method described in the previous section. The parabolic trajectory error averaged at $RMSE_p = 0.2080m$, the average distance to target is $d_{avg} = 0.2273m$, the maximum distance to target is $d_{max} = 0.3286m$ and the median of $d_{med} = 0.2410m$. These results are very similar to the ones obtained in the repeatability analysis, which indicates that the system is able to consistently track the planned trajectories.

B. OUTDOOR ENVIRONMENT

For the outdoor environment, we use a large-scale carbon-fiber quadcopter, custom built by the company *Kopterworx*. The dimensions of the quadcopter are $1.2m \times 1.2m \times 0.45m$ and the mass is $m = 9.5kg$ including all electronics and batteries. The propulsion system is composed of four *T-motor P60 KV170* motors with folding 22.4×8.0 propellers capable of producing maximum thrust of $68N$. The vehicle is equipped with a *Pixhawk 2.1* flight control unit running the *Ardupilot* flight stack, and the *SpektreWorks Kore v1.3.1* power board. Similar to the *AscTec NEO*, the onboard computer is also an *Intel NUC* that communicates with the flight controller through a serial interface. Furthermore, we attached a *Velodyne Puck LITE* LiDAR and *LPMS CU2* IMU used for mapping and localization.

The onboard computer runs Linux *Ubuntu 18.04* with ROS installed and runs all the necessary software. The control structure is a standard PID cascade loop with inner loop controlling the velocity and outer loop controlling the position. The output of the high-level controller consists of attitude roll and pitch angles, yaw rate, and thrust, which are sent to the flight controller. We use the *Cartographer* Simultaneous Localization and Mapping (SLAM) algorithm for obtaining the position and orientation, which are fused with the IMU data through Kalman filter to estimate both position and velocity used for feedback. Apart from the feedback, the *Cartographer* submap clouds are used to generate the OctoMap occupancy grid of the environment. In our previous work [31], we have compared three different SLAM methods on feedback quality and trajectory tracking and have chosen the *Cartographer* for this paper. The detailed hardware and software description of our system can be found in the previously cited paper. Note that we do not use the GPS for outdoor experiments as we want to ensure our system works in GPS-denied environments.

The outdoor environment is roughly the size of $50m \times 80m \times 15m$, depicted on Fig. 10. We limit the maximum height of the UAV to $8m$ because we want to force the path planning algorithm to steer around the obstacles, and not above them. As we do not have the map of the outdoor environment, we have to build it because the planning algorithm depends on it. Within this work, we opted for manual flight to explore the environment. On the other hand, it is possible to employ environment exploration methods, such as the one presented in [41], but this goes beyond the scope of this paper. The target is a wooden crate of size $1.3m \times 1.0m \times 0.7m$ placed on the ground. As the *Kopterworx* UAV differs from the *AscTec NEO*, we impose a different set of constraints given within Table 5.

1) REPEATABILITY

Similar as in Section V-A1, we tested the system's repeatability on $n = 56$ straight-line trajectories with various parabola configuration vectors $\boldsymbol{C}_p$. An illustrative example of the outdoor experiment can be seen on Fig. 1. The system performed successful parabolic airdrops in $39/56$ instances. The overall trajectory tracking errors averaged at $RMSE_{\mathrm{T}} = 0.1901m$, $RMSE_{\mathrm{L}} = 0.2595m$ and $RMSE_{\mathrm{S}} = 0.1531m$. Using the method described in Section V-A1, the reconstructed

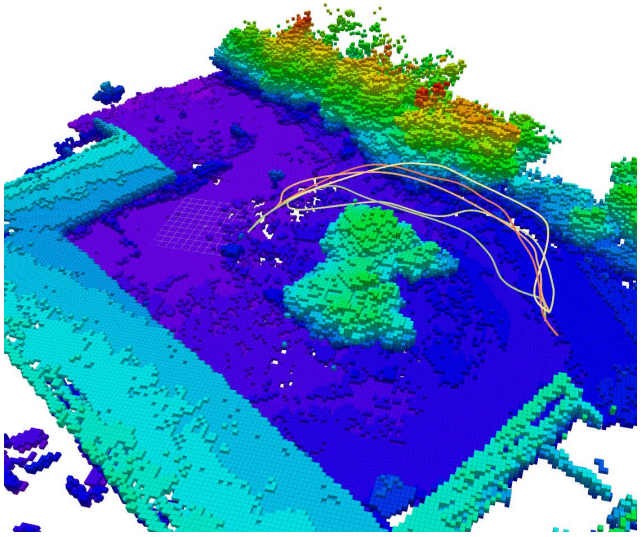

FIGURE 10. An *OctoMap* representation of the outdoor environment with several planned trajectories for the obstacle avoidance. This map was built by manually flying through the environment.

TABLE 5. Velocity and acceleration constraints for all three segments of the planned trajectory for the *Kopterworx* UAV. $\mathscr{T}_\mathrm{T}$ denotes the approach trajectory, planned through the TOPP-RA. $\mathscr{T}_\mathrm{L}$ denotes the launch spline and $\mathscr{T}_\mathrm{S}$ denotes the stopping spline. All values are in standard SI units: $v[m/s]$, $a[m/s^2]$, $\omega[rad/s]$ and $\dot{\omega}[rad/s^2]$.

	v_x	v_y	v_z	ω_z	a_x	a_y	a_z	$\dot{\omega}_z$
$\mathscr{T}_\mathrm{T}$	1.5	1.5	0.75	1.0	0.5	0.5	0.5	0.8
$\mathscr{T}_\mathrm{L}$	4.0	4.0	3.0	2.0	0.8	0.8	0.6	1.0
$\mathscr{T}_\mathrm{S}$	4.0	4.0	4.0	2.0	0.8	0.8	0.8	1.0

parabolic trajectory performance averaged at $RMSE_p = 0.5124m$, average distance to target is $d_{avg} = 0.6191m$ and maximum distance to target is $d_{max} = 1.8303m$ and the median of $d_{med} = 0.5256m$. Due to a series of unpredictable disturbances such as wind gusts, LiDAR beam reflections, etc., as well as due to the fact that only onboard sensors have been used for localization and mapping, the quality of obtained results is slightly decreased compared to indoor experiments.

2) OBSTACLE AVOIDANCE

The obstacle avoidance experiment, together with the environment map, can be seen on Fig. 10. In this experiment, the UAV navigated around trees in the middle of the environment, with the mission to deliver the ball into the box. We performed the obstacle avoidance experiment for $n = 6$ trajectories and the successful delivery was achieved in 3/6 trajectories. The trajectory tracking performance averaged at $RMSE_\mathrm{T} = 0.3642m$, $RMSE_\mathrm{L} = 0.4650m$ and $RMSE_\mathrm{S} = 0.3561m$. Using the method described in Section V-A1, the reconstructed parabolic trajectory performance averaged at $RMSE_p = 0.8844m$, the average distance to target is $d_{avg} = 1.0962m$, the maximum distance to target is $d_{max} = 1.6499m$ and the median of $d_{med} = 1.0833m$. The main difference between the obstacle avoidance and repeatability experiments is the randomness introduced by the RRT* path planning algorithm. All of these factors affected the overall performance of the system, however, the system managed to successfully deliver the ball in multiple instances.

VI. DISCUSSION

In both simulation and experimental verification, the same tests were performed to analyze the system's performance. The combined results can be observed in Table 6, and the statistical box-and-whiskers plot is shown on Fig. 11. The presented data shows a lot of similarities with the indoor experiments, confirming the realistic character of the simulation environment. The only part where the simulation underperforms is the success rate of delivering the ball to the bucket in the office environment. This was the most challenging simulation task because the RRT* algorithm had to plan the path through the environment and into the room where we placed the bucket. The resulting path occasionally featured some sharp turns near the end of the trajectory which directly impacts the launch trajectory. Nevertheless, the system performed very well in the simulation environment.

Furthermore, the difference in the performance between the simulation and the real world can also be attributed to different types of vehicles and feedback used in certain tasks. We would like to point out that the target position was in both cases measured by a human operator, and this measurement error is embedded in the final result. The *Optitrack* system provides a very precise feedback opposed to the *Cartographer* SLAM feedback. These factors mainly influence the difference in the performance between indoor and outdoor environments. The *AscTec NEO* was used in the indoor environment. The vehicle itself can perform quite agile and aggressive maneuvers through the on-board attitude controller and the position MPC. The trajectory tracking RMSE for the indoor scenario is similar to ones reported in the literature. Researchers in [42] report RMSE of 17.53*cm* for a circular trajectory and 11.27*cm* for a lemniscate trajectory. In [43] a RMSE of 16.8*cm* is reported for a trajectory without jerk and snap tracking. In both cases experiments were conducted indoors with the *Optitrack* motion capture system as feedback. The results indicate precise trajectory tracking which is the main factor for high success rate.

The outdoor experiments have the lowest success rate even though the target was the largest among all performed tests. Fig. 11 shows how the outdoor system has higher uncertainty than all others. However, there are several factors one has to take into account. This is the most challenging environment of all because the localization is done only with the onboard sensors. Furthermore, the map of the environment is a-priori unknown and must be built online. This map is then used in the planning procedure, as well as for determining the position of the target. All these factors contribute to the uncertainty and thus the success rate. One can observe a slightly higher drop in success rate for outdoor obstacle avoidance. This behavior is somewhat expected since this is the most challenging task imposed on the system. The main difference

TABLE 6. This table contains data from all conducted simulation and real-world experiments in one place. We also show the success rate in percentages for easier comparison. Apart from the success rate, the measurement unit for all other fields is meter *m*.

	Environment	Success [%]	$RMSE_{\mathrm{T}}$	$RMSE_{\mathrm{L}}$	$RMSE_{\mathrm{S}}$	$RMSE_{\mathrm{P}}$	d_{avg}	d_{max}	d_{med}
Simulation	Repeatability	73.6	0.1299	0.1123	0.0795	0.2046	0.2717	0.9065	0.2481
	City	85.0	0.1086	0.1402	0.0614	0.1557	0.2896	0.3275	0.2917
	Office	72.5	0.0894	0.1522	0.0728	0.4167	0.4723	0.9936	0.4389
	Office bucket	65.0	0.0872	0.1405	0.0646	0.2643	0.3076	0.7914	0.2837
Real world	Indoor	80.0	0.1720	0.2909	0.1465	0.1494	0.1885	0.3357	0.1880
	Indoor obstacle	91.7	0.0751	0.1975	0.1632	0.2080	0.2273	0.3286	0.2410
	Outdoor	69.6	0.1901	0.2595	0.1531	0.5124	0.6191	1.8303	0.5256
	Outdoor obstacle	50.0	0.3642	0.4650	0.3561	0.8844	1.0962	1.6499	1.0833

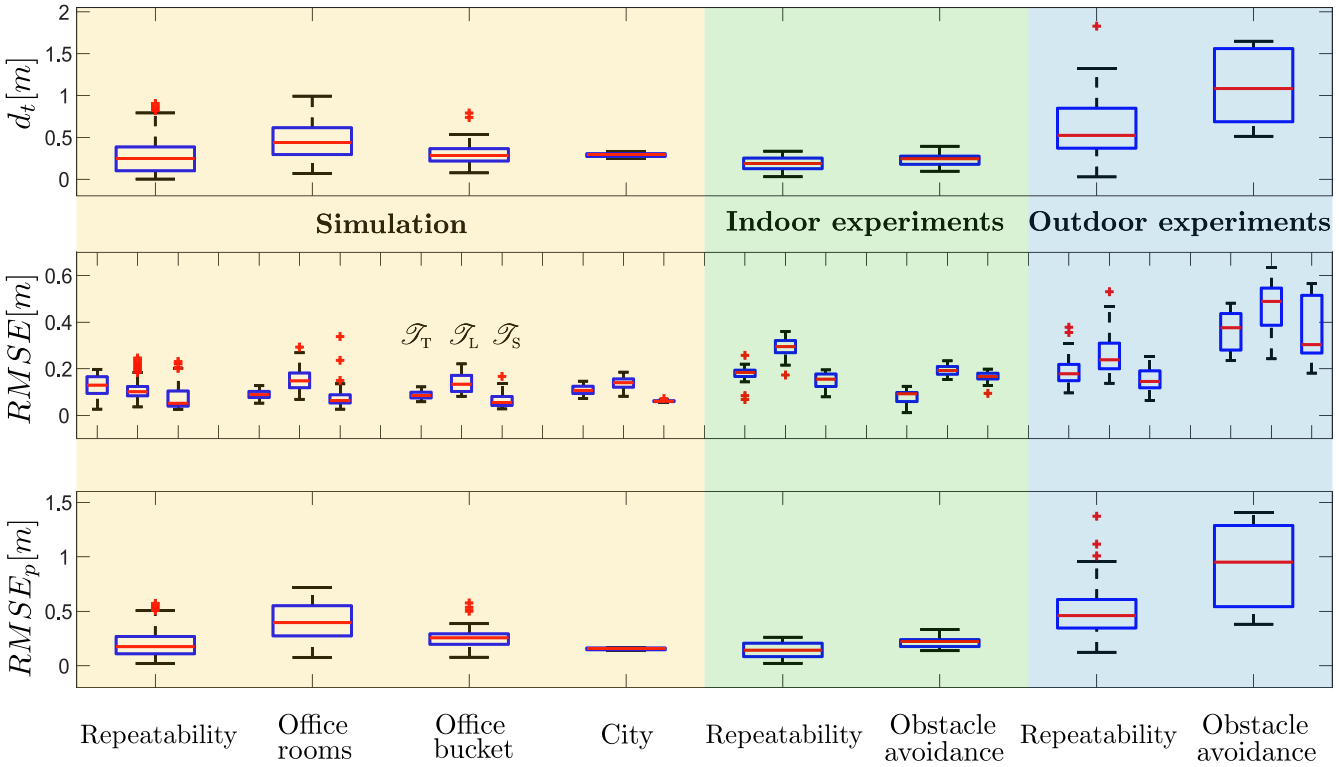

FIGURE 11. Box-plot of the data obtained through the simulation and the experimental analysis. The top plot shows the distance from the target. The middle plot shows the RMSE of trajectories for all the conducted tests. Note that we show three stages of the trajectory planning. For each test, the left bar is the TOPP-RA trajectory, the middle bar is the launch spline and the right bar is the stopping trajectory. The bottom plot shows the RMSE between the planned and the executed parabolic trajectories.

when compared to the outdoor repeatability analysis is the loop closure along the relatively longer trajectory which the Cartographer SLAM performs during the flight. These loop closures are shifting the map of the environment, and therefore the target box, which decreases the success rate of the airdrops. One can argue that the system can take advantage of hovering when performing the drop, especially in case of outdoor experiments where no obstacles were around the target. However, the aim of these experiments is to obtain real-world data to assess planner and overall system performance, keeping in mind the initial intention to deliver the fire extinguishing agent through window or door. Nevertheless, the system managed to perform the parabolic airdrop in most of the experiments.

A. SOURCES OF UNCERTAINTY

We would like to point out that there are several more significant sources of uncertainty that affect the executed parabola in both simulation and experiments. First, there is feedback noise which directly affects the position and the velocity at the launch point. Second, higher velocity and acceleration specified at the launch point will result in a more challenging trajectory with a slightly higher error when reaching the desired launch configuration. Third, the planner assumes zero pitch angle at the launch point. However, this will not be the case while executing the trajectory. Due to the payload displacement from the UAV center of mass $\boldsymbol{T}_{\mathrm{B}}^{\mathrm{P}}$, the non-zero pitch angle while executing the trajectory is introducing the error in the payload position. Fourth, the planner also assumes

zero angular rates at the launch point which will not be the case due to disturbances and localization errors. Angular rates result in some tangential velocity, thus increasing the uncertainty of the payload velocity. Fifth(simulation only), due to the coupling between the ball and the UAV induced by the magnetic force of the simulated magnet, the release position and the release velocity of the ball differ from the planned ones. Small errors at the release point consequently result in errors at the impact point which then differs from the target point.

All of the aforementioned reasons will result in some error in the launch point configuration because they affect initial conditions of the release, and ultimately decrease precision towards the target point. Through our extensive simulation and experimental analyses, we observed the cumulative effect of these uncertainties through presented performance indicators. Even though uncertainties have some effect on the overall results, it is still possible to achieve a successful parabolic airdrop even in outdoor environments.

VII. CONCLUSION

Within this paper we have developed a motion planning method for the parabolic airdrop using multirotor UAVs. Based on the supplied target point, the algorithm searches for a suitable launch point and plans an obstacle free trajectory with information when to release the payload. The method itself is not computationally expensive which makes it suitable for the on-board computer implementation. Therefore, the method was extensively verified in both simulation and experimental environments. The verification process revealed the high precision and success rate of our approach. The comparison between the classic precision airdrop and our approach would be unfair due to different types of vehicles, difference in the release altitude, and the potential use of steering devices after the release. Therefore, we compared the trajectory tracking RMS of our approach with available state-of-the-art approaches to the trajectory tracking of rotorcraft vehicles.

These results show the potential of applying our approach in the real-world scenarios, such as deploying an extinguishing ball into a fire or delivering packages. Such real-world applications come with a set of challenges. Namely, the system can encounter various disturbances, SLAM inaccuracies due to agile maneuvers and potential smoke, which consequently lead to a map inaccuracy. These uncertainties can lead to a higher target miss rate, however, we embedded some of them into our analysis within this paper. Although the target point is currently measured and supplied by the operator, a target detection method can be applied depending on the target type. This widens the use of this method even further as the system has the potential to autonomously detect the target and deploy the payload. It is also worth noting that the parabolic airdrop planning pipeline presented in this paper does not account for dynamic obstacles, which are likely to be present in a real-world fire extinguishing scenario. However, since TOPP-RA has a short execution time it is possible to plan a stop-to-stop trajectory to avoid an obstacle and use the developed spline methodology to account for initial velocity and acceleration. This is beyond the scope of this paper, but it is considered for future work.

To further augment the functionality of this method, our future work includes specifying the rotational velocity at the launch point. Given some displacement of the payload from the UAV body, the rotational velocity will result in a linear tangential velocity, which increases the overall launch velocity. We also plan to explore some sort of a spring loaded ejection mechanism that can increase the launch velocity even further. Both of these augmentations have the potential to significantly increase the required distance from the target, making the airdrop safer because there will be no need for such aggressive maneuvers near obstacles.

The video demonstration of this work can be found in [44].

REFERENCES

[1] M. Polic, A. Ivanovic, B. Maric, B. Arbanas, J. Tabak, and M. Orsag, "Structured ecological cultivation with autonomous robots in indoor agriculture," in *Proc. 16th Int. Conf. Telecommun. (ConTEL)*, Jun. 2021, pp. 189–195.

[2] J. Goričanec, N. Kapetanović, I. Vatavuk, I. Hrabar, G. Vasiljević, G. Gledec, D. Stuhne, S. Bogdan, M. Orsag, T. Petrović, N. Mišković, Z. Kovačić, A. Kurtela, J. Bolotin, V. Kožul, N. Glavić, N. Antolović, M. Anić, B. Kozina, and M. Cukon, "Heterogeneous autonomous robotic system in viticulture and mariculture—Project overview," in *Proc. 16th Int. Conf. Telecommun. (ConTEL)*, Jun. 2021, pp. 181–188.

[3] M. Ward and M. Costello, "Adaptive glide slope control for autonomous airdrop systems," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 2557–2562.

[4] M. R. Cacan, E. Scheuermann, M. Ward, M. Costello, and N. Slegers, "Autonomous airdrop systems employing ground wind measurements for improved landing accuracy," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 6, pp. 3060–3070, Dec. 2015.

[5] J. T. VanderMey, D. B. Doman, and A. R. Gerlach, "Release point determination and dispersion reduction for ballistic airdrops," *J. Guid., Control, Dyn.*, vol. 38, no. 11, pp. 2227–2235, Nov. 2015.

[6] M. Joshua and A. N. Eaton, "Point of impact: Delivering mission essential supplies to the warfighter through the joint precision airdrop system," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Apr. 2013, pp. 783–790.

[7] A. R. Gerlach, S. G. Manyam, and D. B. Doman, "Precision airdrop transition altitude optimization via the one-in-a-set traveling salesman problem," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2016, pp. 3498–3502.

[8] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale real-time visual-inertial localization," *Robot. Sci. Syst.*, vol. 1, no. 37, Jul. 2015.

[9] R. Rakesh and R. Harikumar, "Autonomous airdrop system using small-scale parafoil," in *Proc. Int. Conf. Comput. Commun. Informat. (ICCCI)*, Jan. 2019, pp. 1–6.

[10] S. H. Mathisen, V. Grindheim, and T. A. Johansen, "Approach methods for autonomous precision aerial drop from a small unmanned aerial vehicle," in *Proc. 20th IFAC World Congr.*, vol. 50, 2017, pp. 3566–3573.

[11] S. G. Mathisen, F. S. Leira, H. H. Helgesen, K. Gryte, and T. A. Johansen, "Autonomous ballistic airdrop of objects from a small fixed-wing unmanned aerial vehicle," *Auto. Robots*, vol. 44, no. 5, pp. 859–875, Jan. 2020.

[12] R. Mardiyanto, M. Pujiantara, H. Suryoatmojo, R. Dikairono, and A. N. Irfansyah, "Development of unmanned aerial vehicle (UAV) for dropping object accurately based on global positioning system," in *Proc. Int. Seminar Intell. Technol. Appl. (ISITIA)*, Aug. 2019, pp. 86–90.

[13] M. Müller, S. Lupashin, and R. D'Andrea, "Quadrocopter ball juggling," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2011, pp. 5113–5120.

[14] M. W. Mueller, M. Hehn, and R. D'Andrea, "A computationally efficient motion primitive for quadrocopter trajectory generation," *IEEE Trans. Robot.*, vol. 31, no. 6, pp. 1294–1310, Dec. 2015.

[15] A. Boeuf, J. Cortés, R. Alami, and T. Siméon, "Planning agile motions for quadrotors in constrained environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 218–223.
[16] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 664–674, 2012.
[17] S. Liu, K. Mohta, N. Atanasov, and V. Kumar, "Search-based motion planning for aggressive flight in SE(3)," *IEEE Robot. Autom. Lett.*, vol. 3, no. 3, pp. 2439–2446, Jul. 2018.
[18] B. MacAllister, J. Butzke, A. Kushleyev, H. Pandey, and M. Likhachev, "Path planning for non-circular micro aerial vehicles in constrained environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 3933–3940.
[19] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics research*. Cham, Switzerland: Springer, 2016, pp. 649–666.
[20] H. Oleynikova, M. Burri, Z. Taylor, J. Nieto, R. Siegwart, and E. Galceran, "Continuous-time trajectory optimization for online uav replanning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 5332–5339.
[21] F. Augugliaro, A. P. Schoellig, and R. D'Andrea, "Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 1917–1922.
[22] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1469–1475.
[23] M. M. de Almeida and M. Akella, "New numerically stable solutions for minimum-snap quadcopter aggressive maneuvers," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 1322–1327.
[24] Q.-C. Pham, "A general, fast, and robust implementation of the time-optimal path parameterization algorithm," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1533–1540, Dec. 2014.
[25] H. Pham and Q.-C. Pham, "A new approach to time-optimal path parameterization based on reachability analysis," *IEEE Trans. Robot.*, vol. 34, no. 3, pp. 645–659, Jun. 2018.
[26] V. Spurny, V. Pritzl, V. Walter, M. Petrlik, T. Baca, P. Stepan, D. Zaitlik, and M. Saska, "Autonomous firefighting inside buildings by an unmanned aerial vehicle," *IEEE Access*, vol. 9, pp. 15872–15890, 2021.
[27] B. Arbanas, A. Ivanovic, M. Car, M. Orsag, T. Petrovic, and S. Bogdan, "Decentralized planning and control for UAV–UGV cooperative teams," *Auton. Robots*, vol. 42, no. 8, pp. 1601–1618, Dec. 2018.
[28] A. Ivanovic, M. Car, M. Orsag, and S. Bogdan, "Exploiting null space in aerial manipulation through model-in-the-loop motion planning," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Sep. 2020, pp. 686–693.
[29] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 2520–2525.
[30] P. E. I. Pounds, D. R. Bersak, and A. M. Dollar, "Stability of small-scale UAV helicopters and quadrotors with added payload mass under PID control," *Auton. Robots*, vol. 33, nos. 1–2, pp. 129–142, Feb. 2012.
[31] R. Milijas, L. Markovic, A. Ivanovic, F. Petric, and S. Bogdan, "A comparison of LiDAR-based SLAM systems for control of unmanned aerial vehicles," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, Jun. 2021, pp. 1148–1154.
[32] M. Orsag, C. Korpela, S. Bogdan, and P. Oh, "Dexterous aerial robots—Mobile manipulation using unmanned aerial systems," *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1453–1466, Oct. 2017.
[33] A. Batinovic, A. Ivanovic, T. Petrovic, and S. Bogdan, "A shadowcasting-based next-best-view planner for autonomous 3D exploration," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 2969–2976, Apr. 2022.
[34] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.
[35] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robot.*, vol. 34, no. 3, pp. 189–206, Feb. 2013.
[36] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.
[37] M. D. Mikhailov and A. P. S. Freire, "The drag coefficient of a sphere: An approximation using Shanks transform," *Powder Technol.*, vol. 237, pp. 432–435, Mar. 2013.
[38] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference* (Rotor—A Modular Gazebo MAV Simulator Framework), vol. 1. Cham, Switzerland: Springer, 2016, pp. 595–625.
[39] A. Z. Taddese, P. R. Slawinski, K. L. Obstein, and P. Valdastri, "Closed loop control of a tethered magnetic capsule endoscope," in *Robotics: Science and Systems* (Robotics: Science and Systems Foundation). 2016. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9361658/references
[40] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," in *Studies in Computational Intelligence*. Cham, Switzerland: Springer, 2017, pp. 3–39.
[41] A. Batinovic, T. Petrovic, A. Ivanovic, F. Petric, and S. Bogdan, "A multi-resolution frontier-based planner for autonomous 3D exploration," *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 4528–4535, Jul. 2021.
[42] M. Faessler, A. Franchi, and D. Scaramuzza, "Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories," *IEEE Robot. Autom. Lett.*, vol. 3, no. 2, pp. 620–626, Apr. 2018.
[43] E. Tal and S. Karaman, "Accurate tracking of aggressive quadrotor trajectories using incremental nonlinear dynamic inversion and differential flatness," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 1203–1218, May 2021.
[44] LARICS. *Laboratory for Robotics and Intelligent Control Systems Youtube Channel*. Accessed: Feb. 25, 2022. [Online]. Available: https://www.youtube.com/playlist?list=PLC0C6uwoEQ8YADPAL1_J4S-CawI8Muwz% T

ANTUN IVANOVIC (Student Member, IEEE) received the B.S.E.E. and M.S.E.E. degrees from the University of Zagreb (UNIZG), Croatia, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree with the Laboratory for Robotics and Intelligent Control Systems, Faculty of Electrical Engineering and Computing (FER).

He is employed as a Researcher within the FER, UNIZG. In 2015, he joined LARICS. In 2018, he was a Visiting Researcher with the United States Military Academy West Point, USA, where he collaborated on work related to aerial–ground cooperative manipulation. As a Ph.D. student, he participated in the European Robotics Challenge (EuRoC), the Mohamed Bin Zayed International Robotics Challenge (MBZIRC2020), and the MORUS Project (Unmanned Systems for Maritime Security and Environmental). He is currently working on several EU or national funded projects, including Specularia, ENCORE, and AeRoTwin. His research interests include robotics, unmanned aerial vehicles, aerial manipulation, and motion planning. During his undergraduate studies, he received the Rector's Award for the work entitled "Augmented human–machine interface for aerial manipulators."

MATKO ORSAG (Member, IEEE) is an Associate Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb (FER-UNIZG). He has been involved as a Researcher in various projects financed by the government and industry. In 2011/2012, he worked as a Visiting Researcher with Drexel University, Philadelphia, USA, as a recipient of the Fulbright Exchange Grant. As a Researcher, he participated in several national and international research projects in the field of robotics, control, and automation. Currently, he is working as a Researcher in several EU projects: AeRoTwin, ENCORE, and RoboCom++, as well as the NATO Project MORUS. He is the Principal Investigator of the project SPECULARIA, funded by the Croatian Science Foundation; and the project ENDORSE, funded through the H2020 framework. He has authored and coauthored over 60 scientific and professional papers, including journals and conference papers, as well as a monograph and a book chapter in the field of unmanned aerial systems and robotics. He is a member of the euRobotics Aerial Robotics Topic Group. He is currently the Co-Chair of the Croatian Section of IEEE RAS. He is serving as an Associate Editor for *Automatika—Journal for Control, Measurement, Electronics, Computing, and Communications*.