Optimal control of multiple drones for obstacle avoidance ⋆

Boglárka Sütő * **Alexandru Codrean** * **Zsófia Lendek** *

* *Technical University of Cluj-Napoca, Memorandumului 28, Cluj-Napoca, 400114, Romania (e-mail:suto.boglarka@yahoo.com, alexandru.codrean@aut.utcluj.ro, zsofia.lendek@aut.utcluj.ro)*

Abstract: The control and path planning of multiple drones in a 3D space with obstacle avoidance is a complex challenge, subject to ongoing research. In this paper we propose an optimal control approach, with a baseline controller and filter running on the drones, and a prediction based optimization algorithm running remotely, as a supervisory controller. The supervisor is responsible for calculating the minimal deviation from the trajectory given by the baseline controllers, such that the obstacle is avoided. The approach is tested in simulations with nonlinear drone models in a realistic setting with noise and transmission delays.

Keywords: Optimal control, networked control, drones, obstacle avoidance.

1. INTRODUCTION

Small unmanned aerial vehicles (sUAVs) have become increasingly popular in multiple areas of applications, ranging from education to research, commercial and military. The quadcopter (quadrotor) is the most common type of sUAV used by inexperienced users, teachers, engineers and scientists, because it is inexpensive, light-weight, has easy to understand flight principles/high maneuverability, and simple mechanical design.

Despite its advantages, real life applications impose complex challenges, from the point of view of the control and estimation algorithms that need to be designed, but also with respect to the hardware and software performance that has to be ensured. First, from a classical control perspective, the quadcopter is an underactuated mechanical system (4 actuators with 6 degrees of freedom), nonlinear and inherently unstable, with input and state constraints. So designing a control law for stabilization and tracking is not a trivial task (Emran and Najjaran, 2018). Moreover, many applications require some degree of robustness to model uncertainty and external disturbances (Nascimento and Saska, 2019). Second, not all the states of the drone are measurable and the information from the onboard sensors is not always reliable, as it is usually affected by noise and bias. To alleviate this, Kalman filters are generally used, along with camera based systems (inside) or GPS localization (outside). The third issue that needs to be addressed, besides control and estimation, is path planning (Marshall et al., 2021), such that a sequence of waypoints is generated from the drone's current position to the target (goal) position. Traditionally this task is executed separately from the control loop, and involves different collision avoidance strategies, global or local, online or offline, depending on whether the environment and obstacles are known or not. The interrelation with the control loop and re-planning of the path when detecting unanticipated objects are still subject of intensive research for the 3D case, and represents a challenging task in practice (Huang et al., 2019).

A practical issue that is rarely addressed in the literature is timing, or how fast can the algorithms run given the hardware limitations of the sUAV. The computational capabilities are reduced in comparison with a standalone PC, while the relatively fast drone dynamics imposes a small sampling period. Most nonlinear control algorithms proposed in the literature are designed and analysed in continuous-time, while the implementation is in discrete time (Marshall et al., 2021). To mitigate unwanted discretization effects, many practitioners split the control structure into a lower level control on the drone (simpler, running at higher sampling frequency), and a higher level control running on a remote PC (more complex, running at a lower sampling frequency). This approach however introduces synchronisation issues, and the stability of the overall system is affected by wireless network communication delays and packet loss.

In this context this paper proposes an optimal control and planning approach for multiple drones, with obstacle avoidance. Our starting point is the work of Glotfelter et al. (2017), who developed a framework for the use of nonsmooth barrier functions, exemplified on multi-agent systems (modeled as simple integrators) in a 2D space. Here we consider the 3D case, agents with complex dynamics, and a more realistic setting involving communication delay and noise. Due to the reduced computational capabilities of the drones, we first design a baseline LQR controller and a Kalman filter that can run on a drone in real-time. Then we develop a prediction-based optimisation algorithm that runs off-board, acting as a supervisory controller. Whenever an obstacle is detected, the supervisor computes the minimal deviation from the

⋆ This work was supported by a grant of the Romanian Ministry of Education and Research, CNCS - UEFISCDI, project nr. PN-III-P1-1.1-TE-2019-1956, contract nr. TE185/2021, and SeaClear support project number PN-III-P3-3.6-H2020-2020-006, within PNCDI III.

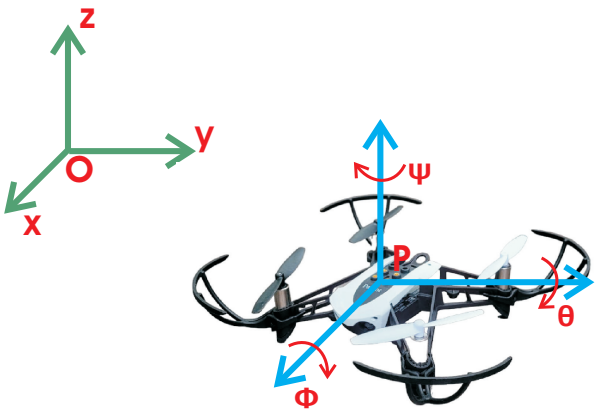

Fig. 1. Parrot Mambo Drone

trajectory given by the baseline controllers, such that the obstacle is avoided, and this signal is transmitted to the drones. The time delays due to the network transmissions are compensated by performing the optimization based on predicted values. The efficiency of our control approach is shown through simulations on the nonlinear model of the Parrot Mambo drone.

The structure of the paper is as follows: Section 2 describes the quadcopter and presents the baseline control and filtering. Section 3 details the optimization algorithm used for collision avoidance and Section 4 illustrates the simulation results. Finally, Section 5 draws some conclusions and mentions directions for future work.

2. QUADCOPTER DESCRIPTION AND BASELINE CONTROL

In this paper we use the Parrot Mambo drone, see Figure 1, which is a lightweight quadcopter, and offers the possibility of direct access to the sensing and actuating hardware. First, we describe the drone model. Next, as a baseline, a linear tracking controller and a linear Kalman filter are designed. These algorithms are simple enough such that they can run on a real drone, despite its reduced computation capabilities. To anticipate the sampling rate restrictions, everything is designed in discrete time.

2.1 Parrot Mambo

The Parrot Mambo is a small drone of 0.18×0.18 meters and weights only 0.063 kg. The four propellers are actuated by DC motors. The drone is equipped with an accelerometer that measures the linear accelerations $\ddot{x}$, $\ddot{y}$ and $\ddot{z}$; a gyroscope for the Euler angle rates $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$; ultrasound sensor to measure the vertical distance from the ground z; a vertical camera, a barometer, and a temperature sensor. An estimation of the velocities $\dot{x}$ and $\dot{y}$ is determined based on camera images, a combination of an optical flow and corner detection algorithms (Derbanne, 2013), and some geometric transformations and corrections (Zeng, 2021). The drone can receive and transmit signals wireless via Bluetooth. The software interface with the sensors, actuators and network communication is made possible due to a firmware specifically developed for Matlab (Matlab, 2022).

2.2 Mathematical model

Next, we describe the mathematical model of the drone. Its pose in the coordinate frame $O(x, y, z)$, see Figure 1, is given by

$$P = [\xi^T \eta^T]^T = [x \quad y \quad z \quad \phi \quad \theta \quad \psi]^T \tag{1}$$

where the position is given by x, y, and z and the orientation by the Euler angles ϕ (roll), θ (pitch), ψ (yaw). The dynamic model [1] of the drone can be developed using the Euler-Lagrange formalism - see (Máthé, 2016) - and is given by [2]

$$\begin{cases} \ddot{x} = [c(\phi)s(\theta)c(\psi) + s(\phi)s(\psi)]\dfrac{U_{coll}}{m} \\ \ddot{y} = [c(\phi)s(\theta)s(\psi) - s(\phi)c(\psi)]\dfrac{U_{coll}}{m} \\ \ddot{z} = -g + c(\phi)c(\theta)\dfrac{U_{coll}}{m} \\ \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = J^{-1}(\eta)\left(\begin{bmatrix} U_\phi \\ U_\theta \\ U_\psi \end{bmatrix} - C(\eta, \dot{\eta})\dot{\eta}\right) \end{cases} \tag{2}$$

with the Jacobian

$$J(\eta) = \begin{bmatrix} I_x & 0 & -I_x s(\theta) \\ 0 & I_y c^2(\phi) + I_z s^2(\phi) & J_{23} \\ -I_x s(\theta) & (I_y - I_z)c(\phi)s(\phi)c(\theta) & J_{33} \end{bmatrix} \tag{3}$$

and the Coriolis matrix

$$C(\eta, \dot{\eta}) = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \tag{4}$$

where

$$\begin{aligned}
J_{23} &= (I_y - I_z)c(\phi)s(\phi)c(\theta) \\
J_{33} &= I_x s^2(\theta) + I_y s^2(\phi)c^2(\theta) + I_z c^2(\phi)c^2(\theta) \\
c_{11} &= 0 \\
c_{12} &= (I_y - I_z)(\dot{\theta}c(\phi)s(\phi) + \dot{\psi}s^2(\phi)c(\theta)) + \\
&\quad (I_z - I_y)\dot{\psi}c^2(\phi)c(\theta) - I_x\dot{\psi}c(\theta) \\
c_{13} &= (I_z - I_y)\dot{\psi}c^2(\theta)s(\phi)c(\phi) \\
c_{21} &= (I_z - I_y)(\dot{\theta}s(\phi)c(\phi) + \dot{\psi}s^2(\phi)c(\theta)) + \\
&\quad (I_y - I_z)\dot{\psi}c^2(\phi)c(\theta) + I_x\dot{\psi}c(\theta) \\
c_{22} &= (I_z - I_y)\dot{\phi}c(\phi)s(\phi) \\
c_{23} &= -I_x\dot{\psi}s(\theta)c(\theta) + I_y\dot{\psi}s^2(\phi)s(\theta)c(\theta) + \\
&\quad I_z\dot{\psi}c^2(\phi)s(\theta)c(\theta) \\
c_{31} &= (I_y - I_z)\dot{\psi}c^2(\theta)s(\phi)c(\phi) - I_x\dot{\theta}c(\theta) \\
c_{32} &= (I_z - I_y)(\dot{\theta}c(\phi)s(\phi)s(\theta) + \dot{\phi}s^2(\phi)c(\theta)) + \\
&\quad (I_y - I_z)\dot{\phi}c^2(\phi)c(\theta) + I_x\dot{\psi}s(\theta)c(\theta) - I_y\dot{\psi}s^2(\phi)s(\theta)c(\theta) \\
&\quad - I_z\dot{\psi}c^2(\phi)s(\theta)c(\theta) \\
c_{33} &= (I_y - I_z)\dot{\phi}c^2(\theta)s(\phi)c(\phi) - I_y\dot{\theta}s^2(\phi)s(\theta)c(\theta) - \\
&\quad I_z\dot{\theta}c^2(\phi)s(\theta)c(\theta) + I_x\dot{\theta}s(\theta)c(\theta).
\end{aligned}$$

The control inputs are the torques on the three rotational axes U_ϕ, U_θ, U_ψ, and the collective force U_{coll}. The

[1] Throughout this paper we use the dot convention for the time derivative ($\dot{\mathbf{x}} := \frac{d\mathbf{x}}{dt}$) and the time dependence is omitted ($\mathbf{x} := \mathbf{x}(t)$) whenever the meaning is obvious.

[2] $s(\cdot)$ and $c(\cdot)$ are shorthand notations for $sin(\cdot)$ and $cos(\cdot)$

parameters - moments of inertia $\{I_x, I_y, I_z\}$, mass m, gravitational acceleration g - are known and given in Table 1.

Table 1. Drone parameters

Parameter	Notation	Value	Units
Mass	m	0.063	kg
X-axis inertia moment	I_x	$0.5829 \cdot 10^{-4}$	kg m^2
Y-axis inertia moment	I_y	$0.7169 \cdot 10^{-4}$	kg m^2
Z-axis inertia moment	I_z	$1.000 \cdot 10^{-4}$	kg m^2
gravitational acceleration	g	9.8	m/s^2

Model (2) can be rewritten as:

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) \tag{5}$$

where the state is $\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]^T$ and the input $\mathbf{u} = [U_{coll}, U_\phi, U_\theta, U_\psi]^T$.

Throughout this paper we assume that the drone model and parameters are known, the linear and angular positions $\{x, y, z, \phi, \theta, \psi\}$, along with their corresponding velocities, can be measured, and the drone can be controlled directly in torque. This is a realistic assumption for indoor flights thanks to the availability of camera systems like OptiTrack (Furtado et al., 2019).

Let the equilibrium point in hovering mode be $\mathbf{x}^e = [x^e, y^e, z^e, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T$, with inputs $\mathbf{u}^e = [U^e_{coll}, 0, 0, 0]^T$. The linearized model is of the form

$$\dot{\mathbf{x}}_l = A\mathbf{x}_l + B\mathbf{u}_l \tag{6}$$

where $\mathbf{x}_l = \mathbf{x} - \mathbf{x}^e$ and $\mathbf{u}_l = \mathbf{u} - \mathbf{u}^e$, and the expressions for the matrices A and B are omitted due to space restrictions. The simplified model can also be written as:

$$\begin{cases} \ddot{x} = \theta g \\ \ddot{y} = -\phi g \\ \ddot{z} = \dfrac{\Delta U_{coll}}{m} \end{cases} \qquad \begin{cases} \ddot{\phi} = \dfrac{U_\phi}{I_x} \\ \ddot{\theta} = \dfrac{U_\theta}{I_y} \\ \ddot{\psi} = \dfrac{U_\psi}{I_z} \end{cases} \tag{7}$$

where $\Delta U_{coll} = U_{coll} - m\,g$. In near-hovering mode, a linear control has been shown to be sufficient in practice (Nascimento and Saska, 2019).

Finally, for the purpose of controller and observer design, we discretize (5) and (6) using Euler's forward method, leading to

$$\mathbf{x}(k+1) = \mathbf{x}(k) + T_s \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) \tag{8}$$

and, in the linear case

$$\mathbf{x}_l(k+1) = \mathbf{x}_l(k) + T_s(A\mathbf{x}_l(k) + B\mathbf{u}_l(k)), \tag{9}$$

where T_s is the sampling period and k refers to the current sample.

2.3 Linear control and filtering

For tracking a given reference position $[r_x\, r_y\, r_z]^T$, we consider the linear state-feedback control law

$$\mathbf{u} = -K\mathbf{e}_x = -K(\mathbf{x} - \begin{bmatrix} r_x \\ r_y \\ r_z \\ 0_{9\times 1} \end{bmatrix}). \tag{10}$$

As a baseline, we design an LQR (Franklin et al., 1998), assuming that the drone is hovering at a constant altitude and keeps the same orientation during tracking. This controller will be running as the default one on the quadcopter.

To design the LQR, we use the linear model (7), and the weight matrices $R_e = \text{diag}([1/15\,1000\,1000\,100])$ and $Q_e = \text{diag}([0.1\,0.1\,10\,0.01\,0.01\,0.01\,0.01\,0.01\,0.01\,1\,0.1,\ 0.1\,0.1])$. For tracking, relatively larger weights are used in Q_e for the position. Also, the weights of z and $\dot{z}$ are lager than the rest (10 and 1, respectively), for the controller to provide sufficiently large thrust for the drone to remain in hovering. For reducing the oscillations in hovering, the weights on the angular rates are 10 times larger than those on the Euler angles.

The control weights are chosen to avoid large input values that would destabilize the drone, and also avoid saturation. The weight corresponding to ΔU_{coll} is considerably smaller than the rest, in order to permit a large enough thrust when needed, particularly for take off.

Although we assume that all the states are measured, since the measurements are not precise and are affected by noise, plus there are model uncertainties, to use these values for control, filtering is necessary. The most commonly used filter is the linear Kalman filter (Grewal and Andrews, 2015), which we will employ in what follows.

The linear discrete-time model (9) is rewritten as

$$\begin{aligned} \mathbf{x}_l(k+1) &= A_d\mathbf{x}_l(k) + B_d\mathbf{u}_l(k) + \mathbf{w}(k) \\ \mathbf{y}_l(k) &= C_d\mathbf{x}_l(k) + \mathbf{v}(k). \end{aligned} \tag{11}$$

The process $\mathbf{w}$ and measurement noise $\mathbf{v}$ are assumed to be independent white noise, with known covariances $E(\mathbf{w}\mathbf{w}^T) = Q$ and $E(\mathbf{v}\mathbf{v}^T) = R$.

The Kalman filter equations we implement are (Welch and Bishop, 2006):

- time update equations

$$\hat{\mathbf{x}}^p_l(k) = A_d\hat{\mathbf{x}}_l(k-1) + B_d\mathbf{u}_l(k-1) + \mathbf{w}(k-1), \tag{12}$$

$$\mathbf{P}^p(k) = A_d\mathbf{P}(k-1)A_d^T + Q, \tag{13}$$

- measurement update equations

$$\mathbf{K}(k) = \mathbf{P}^p(k)C_d^T(C_d\mathbf{P}^p(k)C_d^T + R)^{-1}, \tag{14}$$

$$\mathbf{P}(k) = (I - \mathbf{K}(k)C_d)\mathbf{P}^p(k), \tag{15}$$

$$\hat{\mathbf{x}}_l(k) = \hat{\mathbf{x}}^p_l(k) + \mathbf{K}(k)(\mathbf{y}_l(k) - C_d\hat{\mathbf{x}}^p_l(k)) \tag{16}$$

where the superscript 'p' denotes prediction, $\mathbf{P}$ is the covariance of the estimation error, $\hat{\mathbf{x}}$ the estimated state, and $\mathbf{K}$ the Kalman gain.

2.4 Drone network communication

Wireless communication can be used for sending the control input, reference trajectory, waypoints, etc., thus, we are actually dealing with a networked control system (Hespanha et al., 2007; Tipsuwan and Chow, 2003). For the Parrot Mambo we use the UDP transmission protocol (Stefan et al., 2010), with the assumption that there is no packet loss, and the bandwidth is sufficient for transmitting multiple signals for control purposes.

One of the crucial parameters affecting network transmission is the sampling period. The sampling period on the

Parrot Mambo is by default 5[ms]. However, our experiments revealed that this sampling period is much too small for network transmissions, as it leads to a large amount of packet loss and a transmission delay of over 1s. On the other hand, by increasing the sampling period too much we can miss important transient behavior of the real drone, and the closed loop control performance can deteriorate significantly. We have determined empirically that the smallest sampling period for which our data integrity assumption holds is $T_s = 20$ms.

Based on experiments with the real drone, the transmission delay remains rather constant during an experiment, but can differ from one experiment to the other due to the initialization of network transmission. Round trip delay values varied between 100[ms] and 200[ms] in different experiments. This is consistent with the results of Scola et al. (2021), where a round trip delay of 170[ms] was identified experimentally. Although in real life applications the time delay may vary because the drone moves and obstacles may dynamically appear or disappear, the values determined in our experiments can be used as average values for compensation.

3. OPTIMAL CONTROL AND PLANNING WITH OBSTACLE AVOIDANCE

Up until this point, we considered a single drone, with a given control law, that ensures tracking a given reference, and that can be run on the drone. In what follows, we treat the problem of obstacle avoidance in the case of multiple drones. We consider the scenario, similar to (Glotfelter et al., 2017), where the starting position and the end position of the drones are known and nominal controllers have already been designed. Obstacles are detected and should be avoided during flight with a minimal deviation from the trajectory given by the nominal controllers. A challenge that appears here is due to the reduced computational capabilities of the drones, which makes optimization performed on the drones impossible. Therefore, in the considered setting, all information regarding the states of the drones, detected obstacles, etc., are transmitted to a server, which acts as a supervisory controller. If deviation from the nominal control is necessary, an optimized control input that e.g., ensures avoidance of the obstacle, is computed and transmitted to the drones. If the nominal control is satisfactory, then nothing is transmitted. An issue that we consider is the presence of transmission delays: we assume a 100[ms] transmission delay, corresponding to 5 sampling periods. To compensate for this delay, the optimization is performed based on predicted values. Furthermore, due to the delay in transmitting the information from the drones to the server, the update of the drones' position is also performed with a delay.

In what follows, we explain the basic approach, then prediction-based optimization, first without and then in the presence of delay and measurement noise, and finally way-point generation.

3.1 Approach and setting

The approach we use is an adaptation and extension of the approach of Glotfelter et al. (2017), whose main theoretical results, that represent the starting point of this work, will be summarized next.

Consider the control-affine continuous-time dynamics:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}) + G(\mathbf{x})\mathbf{u} \tag{17}$$

where $\mathbf{x} \in \mathbb{R}^{n_x}$ denotes the states of the system; $\mathbf{u} \in \mathbb{R}^{n_u}$ is the control input, given by any pre-designed control law; $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$, and $G : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x \times n_u}$.

Let $h_i : \mathcal{D} \subset \mathbb{R}^{n_x} \to \mathbb{R}$, $i = 1, 2, \ldots, n_c$ continuously differentiable candidate nonsmooth barrier functions (NBF) – in our case, corresponding to constraints regarding obstacle avoidance –, with locally Lipschitz derivatives, and $h^{\min} : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ defined as $h^{\min} = \min_i\{h_i(\mathbf{x})\}$. Furthermore, let us assume that there exists a locally Lipschitz, extended class-$\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ such that $\dot{h}^{\min}(\mathbf{x}) \geq -\alpha(h^{\min}(\mathbf{x}))$.

Consider $\mathbf{u}^* : \mathcal{D} \subset \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$, solution of the optimization problem

$$\begin{aligned} &\mathbf{u}^*(\mathbf{x}) = \text{argmin}_{\mathbf{u}\in\mathbb{R}^m}(\mathbf{u}^T H(\mathbf{x})\mathbf{u} + b(\mathbf{x})^T\mathbf{u}) \\ &\text{s.t. } \nabla h_i(\mathbf{x})^T(f(\mathbf{x}) + G(\mathbf{x})\mathbf{u}) + \alpha(h_i(\mathbf{x})) > 0, \\ &i = 1, 2, \ldots, n_c \end{aligned} \tag{18}$$

where $H : \mathcal{D} \subset \mathbb{R}^{n_x} \to \mathbb{R}^{n_u \times n_u}$ is locally Lipschitz, symmetric, positive definite, and $b : \mathcal{D} \in \mathbb{R}^{n_x} \to \mathbb{R}^{n_u}$ is locally Lipschitz. If $h^{\min}$ is a candidate NBF and there exist feasible solutions of the optimization problem (18), then $\mathbf{u}^*$ is locally Lipschitz and $h^{\min}$ is a valid NBF for the closed-loop system using the controller $\mathbf{u}^*$. For more details, please see (Glotfelter et al., 2017).

The results above have been exemplified by Glotfelter et al. (2017) on a team of 8 robots with planar, single-integrator dynamics, which have to drive from their initial condition to some desired points, while avoiding obstacles, defined as discs with given centers and radii. In this paper, we consider agents with more complex dynamics in a realistic setting with communication delay and noise, as will be described next.

Consider n drones in 3D space, each with a predefined control law that ensures driving the drones from their initial to their final positions, and m ball-shaped obstacles in this space. Our goal is to modify the control input $\mathbf{u}$ as little as possible such that the robots avoid obstacles and each other. Assuming that, to ensure a safe flight, the minimum distance between the positions of two drones ξ_i and ξ_j should be at least r_{ij}, and between the position of a drone ξ_i and the position of an obstacle ξ_j^o is at least r_j^o, the constraints are defined as:

$$\begin{aligned} &h_{ij}^d(\xi) = ||\xi_i - \xi_j||^2 - r_{ij}^2 \geq 0, i, j = 1,\ 2, \ldots, n \\ &h_{ij}^o(\xi) = ||\xi_i - \xi_j^o||^2 - r_j^{o2} \geq 0, \\ &\quad i = 1, 2, \ldots, n, j = 1, 2, \ldots, m \end{aligned} \tag{19}$$

Similar to (Glotfelter et al., 2017), we stack the dynamics (5) of the n drones, denote $\overline{\mathbf{x}} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_n^T]^T$, $\overline{\mathbf{u}} = [\mathbf{u}_1^T\ \mathbf{u}_2^T, \ldots, \mathbf{u}_n^T]^T$, and use a Boolean composition of the $n_c = n(n-1)/2 + nm$ constraints, i.e., $h^{\min}(\overline{\mathbf{x}}) = \wedge_{i,j=\overline{1,n},i<j} h_{ij}(\overline{\mathbf{x}}) \wedge_{i=\overline{1,n},j=\overline{1,m}} h_{ij}^o(\overline{\mathbf{x}})$, this being the intersection of all constraints, thus arriving to the optimization problem:

$$\begin{aligned}&\overline{\mathbf{u}}^*(\mathbf{x}) = \text{argmin}_{\overline{\mathbf{u}}\in\mathbb{R}^m}(\overline{\mathbf{u}}^T\overline{\mathbf{u}} - \overline{\mathbf{u}}^{\text{nom}T}\overline{\mathbf{u}})\\&\text{s.t. } \nabla h_i(\mathbf{x})^T\overline{\mathbf{f}}(\overline{\mathbf{x}},\overline{\mathbf{u}}) + \alpha(h_i(\overline{\mathbf{x}})) > 0,\\&i = 1,\,2,\,\ldots,\,n_c\end{aligned} \tag{20}$$

where the superscript *nom* denotes the nominal value of the input, given by the predesigned control law, and the overline denotes the stacked variables/ functions.

Solving this optimization problem is not straightforward, due to two reasons: 1) the constraints only involve the positions, not the full state vector and 2) for implementation on actual Parrot Mambo drones, a discrete-time solution is required. Thus, in what follows we propose a discrete-time prediction-based algorithm.

3.2 Prediction based optimization

In order to compute the optimal control input, we first rewrite the objective function and the constraints as an optimization problem for the velocities of the n drones, $\dot{\overline{\xi}} = [\dot{\xi}_1^T,\,\dot{\xi}_2^T,\,\ldots,\,\dot{\xi}_n^T]^T$, not directly the inputs, obtaining

$$\begin{aligned}&\dot{\overline{\xi}}^{\text{opt}} = \text{argmin}_{\dot{\overline{\xi}}}\|\dot{\overline{\xi}}^{\text{nom}} - \dot{\overline{\xi}}\|^2\\&\text{s.t. } \nabla h_i(\overline{\xi})^T\dot{\overline{\xi}} + \alpha(h_i(\overline{\xi})) \geq 0\\&i = 1,\,2,\,\ldots,\,n_c\end{aligned} \tag{21}$$

For this particular case, each constraint is expressed as:

- in case of two drones i and j

$$h_{ij}(\overline{\xi}) = [\xi_i^T\,\xi_j^T]R_{ij}\begin{bmatrix}\xi_i\\\xi_j\end{bmatrix} + \omega_{ij}^T\begin{bmatrix}\xi_i\\\xi_j\end{bmatrix} + \delta_{ij} \tag{22}$$

  with $R_{ij} = \begin{pmatrix}I_3 & -I_3\\-I_3 & I_3\end{pmatrix}$, $\omega_{ij} = 0$, $\delta_{ij} = -r_{ij}^2$

- in case of a drone i and an obstacle j

$$h_{ij}^o(\overline{\xi}) = \xi_i^T R_i^o\xi_i + \omega_{ij}^{oT}\xi_i + \delta_{ij}^o \tag{23}$$

  with $R_i^o = I_3$, $\omega_{ij}^o = -2\xi_j^o$, $\delta_{ij}^o = -r_j^{o2}$.

Given the current state $\overline{\mathbf{x}}$ for all the drones, the positions ξ_i^o and radii r_i^o of the obstacles, and if there is a feasible solution, the quadratic optimization problem (21) can easily be solved to obtain the optimal velocities $\dot{\xi}_i$, $i = 1,\,2,\,\ldots,\,n$. Note however, that this does not solve the problem of optimizing the input to attain the optimal velocities. Therefore, we use a prediction-based optimization, i.e., we will optimize the future velocities, as will be described next.

Since for controlling the real drone a discrete-time control input is required, in what follows, we use model (9). Note that the maximal relative order of the system is 4, and, under the assumption that at time k the current states $\overline{\mathbf{x}}(k)$ and inputs $\overline{\mathbf{u}}^{\text{nom}}(k)$ are available, a 4-step ahead prediction of the positions x and y and a 2-step ahead prediction on z can be performed. Using the predicted values, the optimization problem becomes:

$$\begin{aligned}&\dot{\overline{\xi}}^{\text{opt}}(k) = \text{argmin}_{\dot{\overline{\xi}}}\|\dot{\overline{\xi}}^{\text{pred}}(k) - \dot{\overline{\xi}}(k)\|^2\\&\text{s.t. } \nabla h_i(\overline{\xi}^p(k))^T\dot{\overline{\xi}}(k) + \alpha(h_i(\overline{\xi}(k))) \geq 0\\&i = 1,\,2,\,\ldots,\,n_c\end{aligned} \tag{24}$$

where, with a slight abuse of notation, we used $\dot{\overline{\xi}}(k)$ to denote the linear velocities at sample k and the superscript *pred* denotes predicted values.

Once the optimal velocities become available, based on the discrete-time linear model and the predicted values, the input to be applied is computed. As an example, consider U_θ for the first drone. Based on the current state $\mathbf{x}_1(k)$ and current nominal input $\mathbf{u}_1^{\text{nom}}(k)$, the position $x^{\text{pred}}(k+4)$, the velocity $\dot{x}^{\text{pred}}(k+3)$, etc., can be pre-computed. At this point the optimal velocity $\dot{x}^{\text{opt}}(k+3)$ can be computed as the solution of the optimization problem. Using the discrete-time linear model (9), finite differences, and the predicted values at different sampling instants, the input $U_\theta^{\text{opt}}(k)$ can be recovered.

3.3 Optimization in the presence of noise and delay

An issue that appears in the experimental application is the presence of measurement noise and model mismatch, which in this paper we consider as state noise. Assuming zero mean white noises with covariances Q and R, affecting the states and the outputs, the Kalman filter, as described in Section 2 can be used to obtain a better estimate of the states. Note that the prediction-based optimization remains the same even in the presence of noise: the prediction – and optimization – is made based on the current states only, this being consistent with the zero mean white noise.

Another problem is represented by the computational capacity of the drones. Although the drones can handle simple computations, e.g., computing the control input, their processing power is not enough to solve complex optimization problems. Therefore, the input optimization has to be performed off-board, on a stronger processing unit. This implies transmission of the current states and of the optimal input. As discussed in Section 2, the delay is of $100 - 200$[ms], corresponding to up to 5 sampling periods, and can be considered constant during an experiment. The delay affects both the input optimization – the transmitted input will be available with a delay –, and the filtering/ estimation of the states – which can be performed with a delay.

To handle this issue we propose the following algorithm. Assuming a constant delay of τ samples, an updated estimate of the states at sample k is available off-board for sample $k-\tau$, i.e., $\hat{\overline{\mathbf{x}}}(k-\tau)$. Based on the nonlinear model (2) and the nominal or optimized inputs $\overline{\mathbf{u}}(k-\tau)$, $\overline{\mathbf{u}}(k-\tau+1)$, $\ldots$, $\overline{\mathbf{u}}(k+\tau-1)$ the states $\overline{\mathbf{x}}^{\text{pred}}(k-\tau+1)$, $\overline{\mathbf{x}}^{\text{pred}}(k-\tau+2)$, $\ldots$, $\overline{\mathbf{x}}^{\text{pred}}(k+\tau)$ can be predicted. Note that if a given input $\overline{\mathbf{u}}(d)$ at sample d has been optimized and transmitted, then it has been applied; on the other hand, if nothing has been transmitted, the nominal control input computed on the drone – which may differ from the one computed off-board – will be applied. Based on the predicted state $\overline{\mathbf{x}}^{\text{pred}}(k+\tau)$ and the corresponding nominal input, the control input can be optimized as in Section 3.2; afterwards, it can be transmitted, if required.

Consider now that at sample k a measurement is received, a measurement that actually refers to the state at time $k-\tau+1$. At this point, the Kalman filter can be used to update the prediction $\overline{\mathbf{x}}^{\text{pred}}(k-\tau+1)$, i.e., obtain $\hat{\overline{\mathbf{x}}}(k-\tau+1)$. In turn, the states $\overline{\mathbf{x}}^{\text{pred}}(k-\tau+2)$, $\overline{\mathbf{x}}^{\text{pred}}(k-\tau+3)$, $\ldots$, $\overline{\mathbf{x}}^{\text{pred}}(k+\tau+1)$ and the corresponding inputs that have not been transmitted to the drone can be re-predicted.

3.4 Waypoint generation

Although the goal is to track a given reference point, instead of using it directly in the control law, waypoints are generated along the shortest path from the drone to the goal. Next to potentially reducing the overshoot due to obstacle avoidance, intermediate goal points have the advantage of limiting the control input, and thus avoiding saturation. Waypoints are regenerated every I iterations during flight, and the next points are placed at a distance p from the current positions on the direction to the final point.

4. SIMULATION RESULTS

This section presents the simulation results. The system has been implemented in discrete time in Matlab version R2022a. The scenario we consider concerns 2 robots and 2 obstacles. The initial positions [3] of the robots are $\xi_1(0) = (-2\ 0.5\ 0.5)^T$ and $\xi_2(0) = (-2\ -0.5\ -0.5)^T$ and the goal points are $\xi_{1g} = (2\ -0.5\ -0.5)^T$ and $\xi_{2g} = (2\ 0.5\ 0.5)^T$, respectively. The minimum safe distance between the two robots is $r_{12} = r_{21} = 0.6$. The obstacles' positions are $\xi_1^o = (0.3\ -0.25\ 0.25)^T$ and $\xi_2^o = (0.3\ 0.25\ -0.5)^T$, and the minimum distance to be maintained from them is $r_1^o = r_2^o = 0.3$. The initial, goal, and obstacle positions are chosen such that the nominal trajectories (without obstacle avoidance) intersect and go through the obstacles. Drones are using by default the linear controller designed in Section 2. It is important to note that although the optimization is performed using the linear model of the drones, the input, independent of the employed controller, is applied to the nonlinear model. Furthermore, for prediction during delay compensation also the nonlinear model is used. Waypoints are recomputed after every 30 samples and positioned at a distance of 0.75.

Each robot detects objects (obstacles and drones) in a range of 0.6. To simulate on-line detection, the distance between robots and obstacles are checked every sampling period. Once an object is detected, optimization is activated, i.e., the optimization problem (24) is solved. We chose $\alpha(h_i) = 100h_i^3$, a possibility suggested by Glotfelter et al. (2017). Whether all constraints satisfy $\dot{h}_i(\overline{\mathbf{x}}) \geq -\alpha(h_i(\overline{\mathbf{x}}))$ is verified a posteriori.

First, we consider a scenario without noise and delay. It is assumed that all states are perfectly known and the computed inputs are transmitted instantly. The results for this scenario are presented in Figure 2. As can be seen, the drones avoid the obstacles and arrive at their goal position in 390 samples.

Second, we consider the case when all states are measured, but the measurements are corrupted by zero mean white noise, with covariances $Q = 10^{-4}I$, $R = 10^{-4}I$. Still, we assume that the inputs can be instantly transmitted, i.e., there is no delay. The Kalman filter described in Section 2 is employed to filter the states. Due to the presence of noise, the drones are stopped when the norm of the difference between their current state and their goal position is less then 0.3. Based on 20 tests, the average number of samples required to get to the goal position was 795.2, with standard deviation 273.58. A trajectory is shown in Figure 3.

[3] In what follows, all values are given in meters.

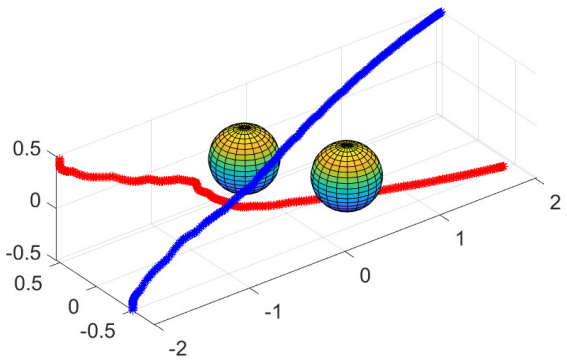

Fig. 2. Obstacle avoidance without noise and delay.

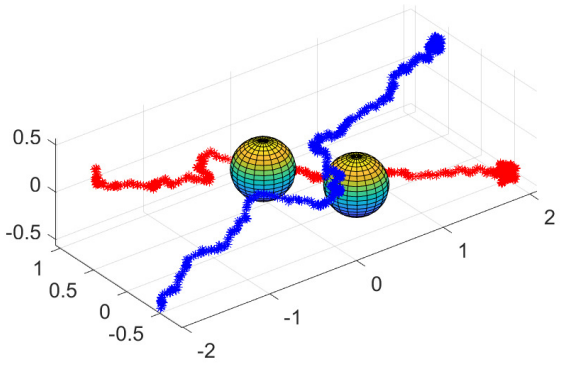

Fig. 3. Obstacle avoidance in the presence of noise.

Finally, we consider the realistic scenario when next to the presence of noise there is a 5-sample delay due to network transmission, both when transmitting the inputs and receiving the measurements. Thus, the Kalman filter is used to update the past estimates of the states; states and inputs are predicted up to 5 samples to compensate for the delay, and based on the prediction, the input is optimized and transmitted, if necessary. Similarly to the previous case, 20 tests have been performed, the drones having been stopped when the difference between the states of the drones and the goal positions is less then 0.3. The average number of samples necessary to reach the final position was 756.6, with a standard deviation of 326.9. Although the average number of steps is about the same, the standard deviation is understandably higher than in the no-delay case, as the optimization is based on predictions with larger uncertainty. A trajectory for this case is shown in Figure 4, illustrating the extra samples due to the uncertainty.

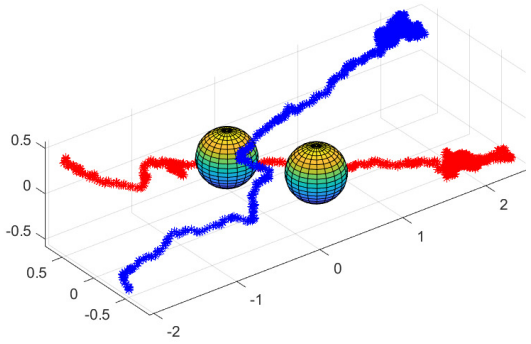

Fig. 4. Obstacle avoidance in the presence of noise and transmission delay.

5. CONCLUSIONS AND FUTURE WORK

This paper presented an optimal control approach for multiple drones, for obstacle avoidance in a 3D space. The limited computation capabilities of the drones was taken into account by splitting the control structure in two: a baseline linear controller and filter running on the drone, and a remote supervisory controller responsible for planning and obstacle avoidance. The supervisory controller consists in a prediction based optimization algorithm which calculates the minimal deviation from the trajectory given by the baseline controllers, such that collisions are avoided. Multiple simulations using the nonlinear model of a Parrot Mambo, realistic noise and network transmission delays, illustrate that our approach works well, the drones arriving at the goal points, while avoiding each other and the obstacles. Our future work will focus on testing this approach in experiments on the Parrot Mambo drones, where some of the states will be measured with OptiTrack cameras, while others will be estimated using extended Kalman filters.

REFERENCES

Derbanne, T. (2013). *Method of evaluating the horizontal speed of a drone, in particular a drone capable of performing hovering flight under autopilot.* US Patent 8.498.447.

Emran, B. and Najjaran, H. (2018). A review of quadrotor: An underactuated mechanical system. *Annual Reviews in Control*, 46, 165–180.

Franklin, G., Powell, J., and Workman, M. (1998). *Digital Control of Dynamic Systems.* Adisson-Wesley.

Furtado, J.S., Liu, H.H., Lai, G., Lacheray, H., and Desouza-Coelho, J. (2019). Comparative analysis of Optitrack motion capture systems. In *Advances in Motion Sensing and Control for Robotic Applications*, 15–31. Springer.

Glotfelter, P., Cortés, J., and Egerstedt, M. (2017). Nonsmooth barrier functions with applications to multi-robot systems. *IEEE Control Systems Letters*, 1(2), 310–315.

Grewal, M.S. and Andrews, A.P. (2015). *Kalman Filtering: Theory and Practice Using MATLAB.* John Wiley & Sons.

Hespanha, J., Naghshtabrizi, P., and Yonggang, X. (2007). A survey of recent results in networked control systems. *Proceedings of the IEEE*, 95(1), 138–162.

Huang, S., Teo, R., and Tan, K. (2019). Collision avoidance of multi unmanned aerial vehicles: A review. *Annual Reviews in Control*, 48, 147–164.

Marshall, J., Sun, W., and L'Afflitto, A. (2021). A survey of guidance, navigation, and control systems for autonomous multi-rotor small unmanned aerial systems. *Annual Reviews in Control*, 52, 390–427.

Máthé, A.K. (2016). *Nonlinear Control for Commercial Drones in Autonomous Railway Maintenance.* PhD Thesis, Technical University of Cluj-Napoca.

Matlab (2022). *Simulink Support Package for Parrot Minidrones.* Matlab.

Nascimento, T. and Saska, M. (2019). Position and attitude control of multi-rotor aerial vehicles: A survey. *Annual Reviews in Control*, 48, 129–146.

Scola, I.R., Reyes, G.G., Carrillo, L.G., Hespanha, J.P., and Burlion, L. (2021). A robust control strategy with perturbation estimation for the Parrot Mambo platform. *IEEE Transactions on Control Systems Technology*, 29(4), 1389–1404.

Stefan, O., Dragomir, T., Codrean, A., and Silea, I. (2010). Issues of identifying, estimating and using delay times in telecontrol systems based on TCP/IP networks. *IFAC Proceedings*, 43(23), 143–148.

Tipsuwan, Y. and Chow, M.Y. (2003). Control methodologies in networked control systems. *Control Engineering Practice*, 11(10), 1099–1111.

Welch, G. and Bishop, G. (2006). *An Introduction to the Kalman Filter.* TR 95-041, Department of Computer Science University of North Carolina at Chapel Hill.

Zeng, X. (2021). *Implementing Tracking Error Control for Quadrotor UAV.* Master Thesis, Eindhoven University of Technology.