Experimental Validation of Sensitivity-Aware Trajectory Planning for a Quadrotor UAV Under Parametric Uncertainty

Ali Srour, Salvatore Marcellini, Tommaso Belvedere, Marco Cognetti, Antonio Franchi, Paolo Robuffo Giordano

Experimental Validation of Sensitivity-Aware Trajectory Planning for a Quadrotor UAV Under Parametric Uncertainty

Ali Srour[1], Salvatore Marcellini[2], Tommaso Belvedere[3], Marco Cognetti[4], Antonio Franchi[3,5], Paolo Robuffo Giordano[1]

***Abstract*— In this work, we provide an experimental validation of the recent concepts of *closed-loop state and input sensitivity* in the context of robust flight control for a quadrotor (UAV) equipped with the popular PX4[1] controller. Our objective is to experimentally assess how the optimization of the reference trajectory w.r.t. these sensitivity metrics can improve the closed-loop system performance against model uncertainties commonly affecting the quadrotor systems. To accomplish this, we present a series of experiments designed to validate our optimization approach on two distinct trajectories, with the primary aim of assessing its precision in guiding the quadrotor through the center of a window at relatively high speeds. This approach provides some interesting insights for increasing the closed-loop robustness of the robot state and inputs against physical parametric uncertainties that may degrade the system's performance.**

I. INTRODUCTION

Aerial robotics has experienced a growing interest propelled by advancements in research, resulting in numerous practical uses. A primary concern with aerial robots is to improve system autonomy and ensure safety during motion tasks. Nevertheless, attaining high levels of autonomy for flying robots remains an enduring challenge due to the need to operate in unpredictable and uncertain real-world conditions. Despite efforts to create precise models, real-world complexities introduce several uncertainties among which inaccuracies in the model parameters, which can potentially disrupt the system behavior during task execution. The pursuit of greater autonomy, precision, and safety in aerial robotics requires bridging the gap between theoretical models and practical real-world conditions. Adaptive [1]–[3] or robust control [4], [5] are typical methods to deal with parametric uncertainty, either through online parameter estimation or by introducing trade-offs between performance and robustness vs. parametric uncertainty. Another approach involves employing Model Predictive Control, as discussed in [6], [7], which predicts the system behavior using a dynamic model. However, the MPC effectiveness depends on the model, and thus parameters, accuracy. Uncertainties in these parameters can significantly impact both the controller performance and the robot behavior. Another possibility is to plan feedforward trajectories with minimal *state sensitivity*, as in [8]–[10]. However, these approaches operate in an open-loop fashion and thus do not consider the presence and strengths/weaknesses of the motion controller that is eventually implemented on the robot.

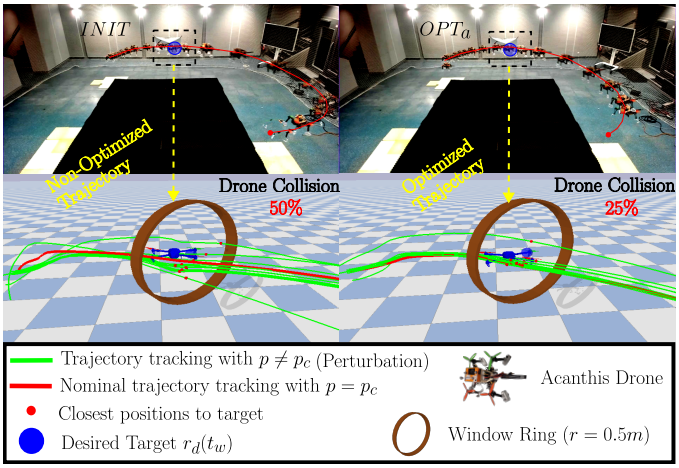

Fig. 1. Drone trajectory tracking under parametric uncertainties (green) to pass through a target $\boldsymbol{s}_d(t_w)$ (blue sphere). Non-optimized ($INIT$, left) and optimized (OPT_a, right) trajectories are compared. The closest positions reached to the desired target after each experiment (small red sphere) are visualized in PyBullet where the drone passes through a circular window at a relatively high speed. A video of the experiments is available at https://youtu.be/QFnrQ_O2BiU.

Recent strategies for addressing the parameter uncertainty problem have been introduced in [11]–[15] where metrics based on *closed-loop state/input sensitivity* assess how uncertainties in robot model affect robots behavior during closed-loop control. To enhance robustness vs. parametric uncertainties, one can plan a feedforward desired trajectory that minimizes these sensitivity metrics, creating a motion plan that is inherently robust. These concepts have been further developed in [16] for optimal initialization of an energy tank in the context of passivity-based control, and in [17] where an extension has been proposed to evaluate the impact of the controller choice, gain tuning and shape of reference trajectory for minimizing the parametric sen-

[1]A. Srour and P. Robuffo Giordano are with CNRS, Univ Rennes, Inria, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France. email: {ali.srour,prg}@irisa.fr

[2]S. Marcellini is with the PRISMA Lab, Department of Electrical Engineering and Information Technology, University of Naples Federico II, Via Claudio 21, Naples, 80125, Italy. email: salvatore.marcellini@unina.it

[3]T. Belvedere and A. Franchi are with the Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00185 Rome, Italy. email: belvedere@diag.uniroma1.it ; antonio.franchi@uniroma1.it

[4]M. Cognetti is with LAAS-CNRS, Université de Toulouse, CNRS, UPS, Toulouse, France LAAS-CNRS, email: mcognetti@laas.fr

[5]A. Franchi is also with the Robotics and Mechatronics Department, Electrical Engineering, Mathematics, Computer Science (EEMCS) Faculty, University of Twente, 7500 AE Enschede, The Netherlands. email: a.franchi@utwente.nl

This work was supported by the project ANR-20-CE33-0003 "CAMP", the Horizon EU research and innovation programme [grant agreement No. 101120732] AUTOASSESS, and the Chaire de Professeur Junior grant no. ANR-22-CPJ1-0064-01.

[1]https://px4.io/

sitivity for quadrotor flight control. Similarly to some of these previous works, [18] proposes an offline optimization scheme in which ellipsoidal tubes on the *state* are used to provide approximate robustness of state constraints against disturbances. Notably, the propagation of the uncertainty ellipsoids exploited in [14]–[17] and used in this work takes a different expression due to being oriented towards parametric uncertainties instead of using a stochastic modeling of disturbances.

While these methods have shown promise in simulated case studies, only in [16] the notion of *closed-loop state/input sensitivity* was applied to real-world experiments with a robotic manipulator. Notably, there have been no real implementations using quadrotors executing a motion task. A major contribution of this work is to provide for the first time a quadrotor-focused experimental assessment and validation of a sensitivity-based trajectory generation. The employed quadrotor is controlled by the widely-used open-source autopilot PX4. While PX4 is used by numerous research groups, companies, and enthusiasts for its valuable features, there are several instances where its control performance can fall short of meeting user requirements. Users often attempt to enhance control by customizing the autopilot, sometimes opting for a model-based approach. However, such customization is not straightforward and demands a strong grasp of firmware architecture and proficient programming skills. Our objective is then to employ the *closed-loop state/input sensitivity* metric (and derived quantities) on a drone controlled by a PX4 default controller to assess how a proper reference trajectory planning, designed to be robust against parametric uncertainties, can enhance the performance of a standard drone.

The rest of the paper is structured as follows: In Sect. II, we recall the main notions of *closed-loop state/input sensitivity*. In Sect. III, we present the reader with the optimization problem proposed with the details of the quadrotor model and tracking controller used. Sect. IV discusses the simulation and the experimental results on the controller performance, and Sect. V concludes the paper.

II. PRELIMINARIES

In this section, we recall for the reader's convenience the main notions related to the *closed-loop state/input sensitivity* introduced in [11], [12] and recently exploited in [13]–[17]. We consider a robot model

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^{n_q}$ is the state, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ the control inputs and $\boldsymbol{p} \in \mathbb{R}^{n_p}$ a vector of (possibly uncertain) model parameters. Letting $\boldsymbol{s}(\boldsymbol{x}) \in \mathbb{R}^{n_s}$ be a quantity of interest (e.g., the position of the robot center), we consider the tracking task of a desired trajectory $\boldsymbol{s}_d(\boldsymbol{a}, t)$ defined for $t \in [t_0, t_f]$ and function of a finite set of trajectory parameters $\boldsymbol{a} \in \mathbb{R}^{n_a}$. The tracking task is realized by a controller having the following generic form

$$\begin{cases} \dot{\boldsymbol{\xi}} = \boldsymbol{g}(\boldsymbol{\xi}, \boldsymbol{x}, \boldsymbol{a}, \boldsymbol{p}_c, \boldsymbol{k}_c, t) \\ \boldsymbol{u} = \boldsymbol{h}(\boldsymbol{\xi}, \boldsymbol{x}, \boldsymbol{a}, \boldsymbol{p}_c, \boldsymbol{k}_c, t) \end{cases}, \tag{2}$$

which is evaluated at a nominal value $\boldsymbol{p}_c$ for the uncertain parameters $\boldsymbol{p}$. Vector $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ represents the possible internal controller states (e.g., an integral action), and $\boldsymbol{k}_c \in \mathbb{R}^{n_k}$ is the vector of control gains.

Following [11], [12], we define the state sensitivity for the closed-loop system (1–2) as

$$\boldsymbol{\Pi}(t) = \left.\frac{\partial \boldsymbol{x}(t)}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \tag{3}$$

and the input sensitivity as

$$\boldsymbol{\Theta}(t) = \left.\frac{\partial \boldsymbol{u}(t)}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_c}. \tag{4}$$

Matrix $\boldsymbol{\Pi}(t)$ quantifies how variations of the parameters $\boldsymbol{p}$ around a nominal value $\boldsymbol{p}_c$ will affect the evolution of the state $\boldsymbol{x}$ (in closed-loop). Analogously, matrix $\boldsymbol{\Theta}(t)$ relates variations of $\boldsymbol{p}$ to variations of the inputs $\boldsymbol{u}$. Although a closed-form expression for matrices $\boldsymbol{\Pi}(t)$ and $\boldsymbol{\Theta}(t)$ is in general not available, it is possible to easily obtain these two quantities via forward integration of a differential equation along the system trajectories [11], [12].

Matrices $\boldsymbol{\Pi}(t)$ and $\boldsymbol{\Theta}(t)$ can then be used for several purposes in the context of trajectory optimization, for instance for obtaining suitable sensitivity metrics to be optimized or an estimation of the envelope of 'perturbed trajectories' for the states/inputs. To this end, assume that each parameter p_i can vary in a given range δp_i centered at a nominal p_{c_i}

$$p_i \in [p_{c_i} - \delta p_i,\, p_{c_i} + \delta p_i] \tag{5}$$

and define the diagonal weight matrix $\boldsymbol{W} = \mathrm{diag}(\delta p_i^2)$. Letting $\Delta\boldsymbol{p} = \boldsymbol{p} - \boldsymbol{p}_c$, an ellipsoid in parameter space centered at $\boldsymbol{p}_c$ and with semi-axes δp_i has equation

$$\Delta\boldsymbol{p}^T \boldsymbol{W}^{-1} \Delta\boldsymbol{p} = 1. \tag{6}$$

Following the derivations in [14], from (6) and (3–4) one can obtain the corresponding ellipsoids in state space

$$\Delta\boldsymbol{x}^T (\boldsymbol{\Pi}\boldsymbol{W}\boldsymbol{\Pi}^T)^{-1} \Delta\boldsymbol{x} = 1 \tag{7}$$

and in input space

$$\Delta\boldsymbol{u}^T (\boldsymbol{\Theta}\boldsymbol{W}\boldsymbol{\Theta}^T)^{-1} \Delta\boldsymbol{u} = 1, \tag{8}$$

where $\Delta\boldsymbol{x} = \boldsymbol{x} - \boldsymbol{x}_{nom}$ and $\boldsymbol{x}_{nom}$ is the state evolution of (1–2) in the unperturbed case $\boldsymbol{p} = \boldsymbol{p}_c$, and analogously for $\Delta\boldsymbol{u} = \boldsymbol{u} - \boldsymbol{u}_{nom}$.

The state and input space ellipsoids can be exploited for defining a "weighted sensitivity norm" by considering the eigenvalues λ_i of the kernel matrix $\boldsymbol{\Pi}\boldsymbol{W}\boldsymbol{\Pi}^T$ in (7). In particular, in [14] and in this work we consider the following matrix norm

$$\|\boldsymbol{\Pi}\|_W = \max(\lambda_i(\boldsymbol{\Pi}\boldsymbol{W}\boldsymbol{\Pi}^T)) \tag{9}$$

which represents the largest (worst-case) deviation of the state $\boldsymbol{x}$ assuming a parametric uncertainty as in (5). Furthermore, one can also exploit (7–8) for obtaining the tubes of perturbed trajectories for the individual components of the states and the inputs. By referring to [14] for all details, for each direction of interest in the input space, one can obtain the "tube radius" $r_i(t)$ such that

$$u_{nom,i}(t) - r_i(t) \le u_i(t) \le u_{nom,i}(t) + r_i(t). \tag{10}$$

where $u_{nom,i}(t)$ is the behavior of the input $u_i(t)$ in the unperturbed case $\boldsymbol{p} = \boldsymbol{p}_c$. Equation (10) bounds from above/below the envelope of perturbed inputs when the parameter uncertainty is bounded as in (5), and an analogous upper/lower bound can also be obtained for the generic state component $x_i(t)$.

III. OPTIMIZATION PROBLEM

In this work, we consider the following trajectory optimization problem

$$\begin{aligned} \boldsymbol{a}^* = \arg\min_{\boldsymbol{a}} \|\boldsymbol{\Pi}(t_w)\|_W \\ s.t. \quad \boldsymbol{M}\boldsymbol{a} = \boldsymbol{b} \\ U_{min,i} \leq u_{nom,i}(t) - r_i(t) \; \forall i \, \forall t \in [t_0,\, t_f] \\ u_{nom,i}(t) + r_i(t) \leq U_{max,i} \; \forall i \, \forall t \in [t_0,\, t_f]. \end{aligned} \tag{11}$$

We seek the optimal value $\boldsymbol{a}^*$ of the shape parameter $\boldsymbol{a}$ of the reference trajectory $\boldsymbol{s}_d(\boldsymbol{a}\,, t)$ for minimizing the weighted norm (9) at a specific time t_w (e.g., for passing through a desired point like the center of a window). Minimization of this cost will increase the robustness (thus reducing the sensitivity) of the closed-loop system against parameter uncertainties at t_w. The constraints consist of given initial/final conditions for $\boldsymbol{s}_d(\boldsymbol{a}\,, t)$, represented by the linear constraints $\boldsymbol{M}\boldsymbol{a} = \boldsymbol{b}$, and constraints that bound the envelope of perturbed inputs within actuation limits $U_{min,i} \leq U_{max,i}$, ensuring that the tracking of the optimized reference trajectory will be feasible for any value of the uncertain parameters $\boldsymbol{p}$ in the range (5). Note that these constraints leverage the "input tubes" as described in (10). Of course, other objective functions can be considered such as the integral of the sensitivity norm along the trajectory to enhance tracking accuracy during motion. Furthermore, one can easily include extra constraints like obstacle avoidance by exploiting the tubes on the states, as shown in [15]. We note that this optimization problem is non-convex (as any other trajectory optimization problem of this type), and therefore one can only guarantee convergence to a local minimum for a given initial guess.

A. Quadrotor Model

Concerning the quadrotor dynamics, we adopted the model from [14] that considers a displaced center of mass (CoM). Let $\boldsymbol{\mathcal{F}}_{\mathcal{W}} = \{\boldsymbol{O}_{\mathcal{W}}, \boldsymbol{x}_{\mathcal{W}}, \boldsymbol{y}_{\mathcal{W}}, \boldsymbol{z}_{\mathcal{W}}\}$ denote the world inertial frame, and $\boldsymbol{\mathcal{F}}_{\mathcal{B}} = \{\boldsymbol{O}_{\mathcal{B}}, \boldsymbol{x}_{\mathcal{B}}, \boldsymbol{y}_{\mathcal{B}}, \boldsymbol{z}_{\mathcal{B}}\}$ represent the body frame attached to the drone geometric center. We define: $\boldsymbol{r} = (x, y, z) \in \mathbb{R}^3$ as the drone position, $\boldsymbol{v} = (v_x, v_y, v_z) \in \mathbb{R}^3$ as its linear velocity in the world frame $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, $\boldsymbol{q} = (q_w, q_x, q_y, q_z) \in \mathbb{S}^3$ as the unit-norm quaternion representing the orientation of $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$ relative to $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, and $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z) \in \mathbb{R}^3$ as the angular velocity of $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$ with respect to $\boldsymbol{\mathcal{F}}_{\mathcal{W}}$, expressed in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$. The quadrotor state vector is $\boldsymbol{x} = (\boldsymbol{r}, \boldsymbol{v}, \boldsymbol{q}, \boldsymbol{\omega}) \in \mathbb{R}^6 \times \mathbb{S}^3 \times \mathbb{R}^3$.

Let w_i be the squared velocity of the i-th propeller and define the quadrotor control input $\boldsymbol{u} = (w_1, \ldots, w_4)$. An allocation matrix is used to relate the inputs $\boldsymbol{u}$ (i.e., the squared propeller speeds) to the thrust/torques $(f, \boldsymbol{\tau})$

$$\begin{bmatrix} f \\ \boldsymbol{\tau} \end{bmatrix} = k_f \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & l & 0 & -l \\ -l & 0 & l & 0 \\ k_m & -k_m & k_m & -k_m \end{bmatrix} \boldsymbol{u} = \boldsymbol{T}\boldsymbol{u} \tag{12}$$

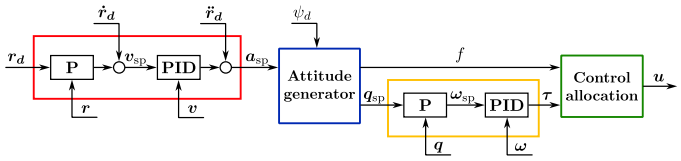

Fig. 2. Block diagram of the PX4 controller, composed of four stages: the position controller (red), the attitude generator (blue), the attitude controller (yellow), and the control allocation (green).

where l is the arm length of the quadrotor arm, and k_f and k_m are the thrust and the drag aerodynamic coefficients of the propellers [19], [20]. We consider that the quadrotor center of mass $\boldsymbol{G}_{\mathcal{B}}$ is displaced w.r.t. the geometric center $\boldsymbol{O}_{\mathcal{B}}$ by a displacement denoted as $\boldsymbol{g}_C = (g_x, g_y, g_z)$ in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$. The parameters considered uncertain are then the set[2] $\boldsymbol{p} = (k_f,\, g_x,\, g_y,\, g_z,\, m) \in \mathbb{R}^5$. The total force $\boldsymbol{f}_{tot}$ acting on the quadrotor in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$ includes the propeller thrust f, gravitational effects, and an additional fictitious force due to displacement $\boldsymbol{g}_C$:

$$\boldsymbol{f}_{tot} = f\boldsymbol{z}_W - mg\mathbf{R}(\boldsymbol{q})^T\boldsymbol{z}_W - m[\boldsymbol{\omega}]_\times[\boldsymbol{\omega}]_\times\boldsymbol{g}_C.$$

For the total torque $\boldsymbol{\tau}_{tot}$ (expressed in $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$), we have:

$$\boldsymbol{\tau}_{tot} = \boldsymbol{\tau} - mg[\mathbf{g}_C]_\times\mathbf{R}(\boldsymbol{q})^T\boldsymbol{z}_W - [\boldsymbol{\omega}]_\times\mathbf{J}\boldsymbol{\omega},$$

where $\boldsymbol{\tau}$ represents the propeller torque as in (12), $\mathbf{R}(\boldsymbol{q}) \in SO(3)$ is the rotational matrix associated with the quaternion $\boldsymbol{q}$ and $\mathbf{J}$ stands for the body frame inertia matrix about the geometric center $\boldsymbol{\mathcal{F}}_{\mathcal{B}}$. By defining the spatial inertia matrix

$$\boldsymbol{M} = \begin{bmatrix} m\boldsymbol{I_3} & -m[\mathbf{g}_C]_\times \\ m[\mathbf{g}_C]_\times & \mathbf{J} \end{bmatrix}$$

one can finally obtain the following relation:

$$\begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\alpha} \end{bmatrix} = \boldsymbol{M}^{-1} \begin{bmatrix} \boldsymbol{f}_{tot} \\ \boldsymbol{\tau}_{tot} \end{bmatrix}$$

which leads to the quadrotor model with displaced center of mass

$$\begin{cases} \dot{\boldsymbol{r}} = \mathbf{R}(\boldsymbol{q})\boldsymbol{v} \\ \dot{\boldsymbol{v}} = \boldsymbol{\nu}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \\ \dot{\boldsymbol{q}} = \frac{1}{2}\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \otimes \boldsymbol{q} \\ \dot{\boldsymbol{\omega}} = \boldsymbol{\alpha}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) \end{cases}. \tag{13}$$

B. PX4 Controller

It is well known that the position $\boldsymbol{r}$ and the yaw angle ψ are flat outputs for the quadrotor model (13), as explained in, e.g., [21]. Based on this fact, the PX4 controller is designed to track a desired position $\boldsymbol{r_d}(t)$ and a yaw angle $\psi_d(t)$ trajectories by generating suitable propeller speeds $\boldsymbol{u}$. More in detail, the controller has a cascaded structure consisting of four stages (as illustrated in Fig. 2): *(i)* the *position controller* in which the desired position $\boldsymbol{r_d}$ is transformed into an equivalent acceleration setpoint $\boldsymbol{a}_{\text{sp}}$, *(ii)* the *attitude generator* responsible for producing both a combined thrust f and an attitude setpoint $\boldsymbol{q}_{\text{sp}}$ derived from the desired yaw angle $\boldsymbol{\psi_d}$ and the previously computed acceleration setpoint,

[2]We did not include k_m since uncertainties in this parameter have a negligible impact compared to the other parameters as shown in, e.g., [17].

(iii) the *attitude controller* determining the desired body torques and, finally, *(iv)* the *control allocation* where thrust and body torques are transformed in propeller speeds via the inverse of the nominal allocation matrix (12).

Following the available documentation and open-source code[3], we have implemented an equivalent continuous time controller and integrated it into our sensitivity-aware planning scheme. Note that, in the PX4 controller, the inertial and the body frames are expressed using the North-East-Down (NED) convention, so trivial conversions are needed if one chooses to express the dynamic model in different frames.

The position controller in the PX4 consists of a P and a PID action on the position and velocity errors, respectively. The latter introduces two 3-dimensional internal states $\boldsymbol{\xi}_v$ and $\boldsymbol{\xi}_a$ to realize the integral and the low-pass-filtered derivative actions. Denoting with Ω_p the low-pass filter cutoff pulsation and with $\boldsymbol{P}_r$, $\boldsymbol{P}_v$, $\boldsymbol{I}_v$, $\boldsymbol{D}_v$ the diagonal matrix gains, the position controller equations can be expressed as

$$\begin{aligned}\dot{\boldsymbol{\xi}}_v &= \dot{\boldsymbol{r}}_{\boldsymbol{d}} - \boldsymbol{v}\\ \dot{\boldsymbol{\xi}}_a &= -\Omega_p \boldsymbol{\xi}_a - \Omega_p^2 \boldsymbol{v}\\ \boldsymbol{a}_{\text{sp}} &= \ddot{\boldsymbol{r}}_{\boldsymbol{d}} + \boldsymbol{P}_v(\dot{\boldsymbol{r}}_{\boldsymbol{d}} - \boldsymbol{v} + \boldsymbol{P}_r(\boldsymbol{r}_{\boldsymbol{d}} - \boldsymbol{r}))\\ &\quad - \boldsymbol{D}_v(\boldsymbol{\xi}_a + \Omega_p \boldsymbol{v}) + \boldsymbol{I}_v \boldsymbol{\xi}_v.\end{aligned}$$

The attitude generator receives the acceleration setpoint $\boldsymbol{a}_{\text{sp}} = (\ddot{x}_{\text{sp}}, \ddot{y}_{\text{sp}}, \ddot{z}_{\text{sp}})$ and computes the collective thrust f. Let $\boldsymbol{b}_z = \frac{(-\ddot{x}_{\text{sp}}, -\ddot{y}_{\text{sp}}, g)}{\|(-\ddot{x}_{\text{sp}}, -\ddot{y}_{\text{sp}}, g)\|}$ and $\boldsymbol{t}_{\text{sp}} = \frac{\boldsymbol{b}_z}{b_{z,z}}\left(\ddot{z}_{\text{sp}}\frac{h_t}{g} - h_t\right)$, where $h_t \in (0,\, 1)$ denotes the nominal normalized hovering thrust, so that the computed thrust is $f = -\|\boldsymbol{t}_{\text{sp}}\|$. The attitude setpoint $\boldsymbol{q}_d$ is then computed by aligning the z-axis of the body frame to the desired thrust $\boldsymbol{t}_{\text{sp}}$ and rotating of an angle ψ_d around such axis [22]. To generate the body torques, let the quaternion error be $\boldsymbol{q}_{\text{e}} = \boldsymbol{q}^{-1} \otimes \boldsymbol{q}_{\text{sp}}$. The angular rate setpoint is then computed proportionally to the error as $\boldsymbol{\omega}_{\text{sp}} = 2\boldsymbol{P}_q \,\text{sign}(q_{\text{e},w})\boldsymbol{q}_{\text{e},xyz}$, with $\boldsymbol{P}_q$ denoting the proportional gains and $\boldsymbol{q}_{\text{e}} = [q_{\text{e},w},\, \boldsymbol{q}_{\text{e},xyz}]^T$. Finally, the equations of the PID controller tracking the attitude rate $\boldsymbol{\omega}_{\text{sp}}$ are

$$\begin{aligned}\dot{\boldsymbol{\xi}}_\omega &= \boldsymbol{\omega}_{\text{sp}} - \boldsymbol{\omega}\\ \dot{\boldsymbol{\xi}}_\alpha &= -\Omega_a \boldsymbol{\xi}_\alpha - \Omega_a^2 \boldsymbol{\omega}\\ \boldsymbol{\tau} &= \boldsymbol{P}_\omega(\boldsymbol{\omega}_{\text{sp}} - \boldsymbol{\omega}) - \boldsymbol{D}_\omega(\boldsymbol{\xi}_\alpha + \Omega_a \boldsymbol{\omega}) + \boldsymbol{I}_\omega \boldsymbol{\xi}_\omega,\end{aligned}$$

where $\boldsymbol{\xi}_\omega$ and $\boldsymbol{\xi}_\alpha$ are the 3-dimensional internal states of the controller, Ω_a is the cutoff pulsation of the low-pass filter, and $\boldsymbol{P}_\omega$, $\boldsymbol{I}_\omega$, $\boldsymbol{D}_\omega$ are the diagonal matrix gains.

Lastly, the control input $\boldsymbol{u}$ is reconstructed from the allocation matrix as

$$\boldsymbol{u} = \boldsymbol{T}^{-1}\begin{bmatrix} f \\ \boldsymbol{\tau}\end{bmatrix}$$

where, of course, the matrix $\boldsymbol{T}$ from (12) is evaluated on the *nominal* values of the parameters $\boldsymbol{p}_c$ (which may differ from the actual $\boldsymbol{p}$ because of inaccuracies in the quadrotor model).

IV. EXPERIMENTAL ANALYSIS

A series of experiments and simulations were conducted to assess the effectiveness of the proposed trajectory generation obtained by solving (11). Two distinct trajectories $\boldsymbol{s}_d(\boldsymbol{a},\, t) = (\boldsymbol{r}_d(\boldsymbol{a}\,, t),\, \psi_d(\boldsymbol{a}\,, t))$, denoted as Traj1 and Traj2 in the following, were used to guide the drone through a circular window at a relatively high speed ($2.0 \leq v \leq 2.5$ m/s) (see Fig. 3) while complying with the initial/final state constraints (rest-to-rest motions) and input saturations as in (11). Note that the actual window was not introduced during the real experiments for safety reasons, but it was used in PyBullet[4] with the real flight data for visualization.

We start by generating a first trajectory – that will be referred to as $INIT$ in the following – constructed using piecewise Bezier curves, as discussed in [11], [12], [17], [23], by performing a preliminary trajectory optimization that minimizes the *snap* of $\boldsymbol{r}_d(\boldsymbol{a}\,, t)$ over the whole time interval with the aim of obtaining a smooth trajectory with minimal curvature changes and satisfying nominal constraints over the state and inputs. Thus, all initial non-optimized trajectories $INIT$ use the minimum snap cost function defined as

$$\begin{aligned}\boldsymbol{a} = \arg\min_{\boldsymbol{a}} &\left(\int_{t_0}^{t_f} \|\frac{d^4 r_d(\boldsymbol{a}, \tau)}{dt^4}\|^2 \mathrm{d}\tau\right)\\ &s.t. \quad \boldsymbol{M}\boldsymbol{a} = \boldsymbol{b}\\ U_{min,i} \leq u_{nom,i}(t) &\leq U_{max,i}\ \forall i\, \forall t \in [t_0,\, t_f].\end{aligned} \tag{14}$$

Our framework then modifies this trajectory by solving (11) with $INIT$ as an initial guess. The resulting trajectory is denoted as OPT_a. More in detail, our framework is implemented in Python and utilizes the COBYLA [24] nonlinear optimizer from the nlopt toolbox by employing the symbolic toolbox for symbolic system representation and making use of the Jitcode [25] framework for in-time compilation of ordinary differential equations. Within this framework, optimizing trajectories with the PX4 controller typically requires 4 to 6 minutes per trajectory.

Our main goal is to evaluate how close the drone gets to a specific target location, labeled as $\boldsymbol{r}_d(t_w)$, which is right at the center of a predefined window. We want to find out if the drone can safely go through the window, even when in presence of uncertainties in its parameters. Additionally, we want to know how close the drone can get to the center of the window. Traj1, having a duration of 7 seconds, requires a relatively low initial acceleration to reach the target. Conversely, Traj2, which has a duration of 5 seconds, requires a more significant initial acceleration to reach the target. Both trajectories ensure a minimum speed of 2 m/s when they reach the target.

A. Simulation Results

After obtaining trajectories using our sensitivity Python framework, dynamic simulations have been carried out in Gazebo thanks to the software in the loop (SITL) feature of PX4, which allows to simulate the autopilot and communicate through a ROS 2 application. This application reads the desired trajectory from a csv file, and it sends the series of setpoints (position, velocity, and acceleration) to the drone.

Starting from the default model of the Iris quadrotor, we performed a simulation running with the nominal parameters. Then, we carried out $N_p = 11$ simulations using uncertain parameters $\boldsymbol{p}$ generated by uniformly sampling the inner

[3] https://github.com/PX4/PX4-Autopilot

[4] https://pybullet.org/

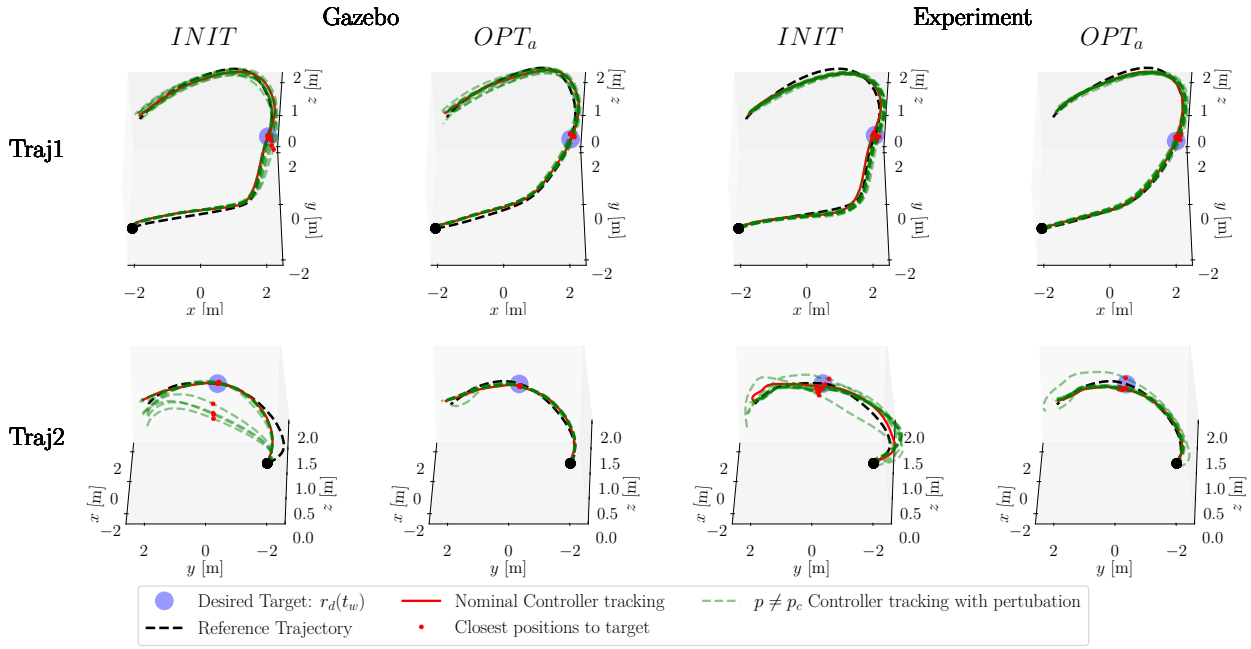

Fig. 3. Trajectory tracking results by PX4 controller for two trajectory types (1st row: Traj1, 2nd row: Traj2) in simulation (first two columns) and experiments (last two columns). All trajectories start from the black sphere at $(-2, -2, 1)$. Cases include $INIT$ and OPT_a. Perturbed runs consist of N_p green trajectories, while the nominal trajectory tracking (i.e. when $\boldsymbol{p} = \boldsymbol{p}_c$) is denoted in red, all sharing an origin which is the black sphere. Parameters $\boldsymbol{p}$ were randomly drawn from (6) in Gazebo simulation where it was implemented by modifying the Iris URDF model. However, the experiments were carried out as in (Fig: 7) where actual modifications were done on the drone that resemble the perturbations that will affect parameters $\boldsymbol{p}$. Red spheres at $t = t_w$ mark closest points to desired target $\boldsymbol{r}_d(t_w)$, forming a point cloud around it.

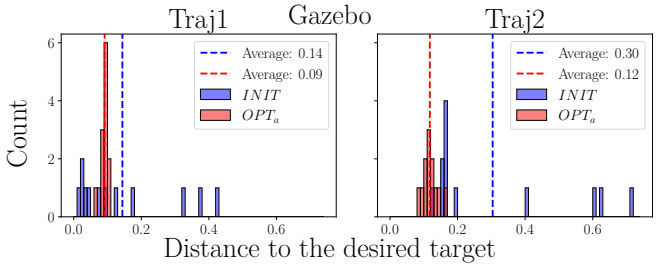

Fig. 4. Histograms of two trajectory types (left: Traj1, right: Traj2) for both cases ($INIT$ in blue, OPT_a in red) showing the distances to the desired target ($|\boldsymbol{r}_d(t_w) - \boldsymbol{r}(t_w)|$) in [m] and its average across N_p runs in Gazebo.

volume of the ellipsoid (6). This is done for Traj1 and Traj2 and for both $INIT$ and OPT_a cases. The adjustments of the parameters were made by modifying the Iris URDF model. The uncertainty intervals δp_i for the matrix $\boldsymbol{W}$ are chosen relative to the nominal parameters $\boldsymbol{p}_c$ as $\delta k_f = 0.1 k_f$, $\delta g_{x,y} = 0.1l$, $\delta g_z = 0.1h$, and $\delta m = 0.1m$, where h is the distance between the top and bottom plates of the drone chassis. Thus, the optimization problem of (11) will aim at minimizing the sensitivity of the quadrotor position $r(t_w)$ at *the instant of passing through the window center* t_w against variations in the parameters k_f, g_x, g_y, g_z and m.

In our initial assessment, as depicted in Fig. 3, we present the results for the N_p "perturbed runs" for both Traj1 and Traj2 (1st and 2nd row, respectively) under the Gazebo label (first two columns). The plots report the different targets reach $r(t_w)$ (red dots) that are the closest to $r_d(t_w)$ (window center) for each of the perturbed simulations (in green). The result is a point cloud around the desired target $r_d(t_w)$. It is worth noticing that, when there is no parameter uncertainty affecting the system (i.e. when $\boldsymbol{p} = \boldsymbol{p}_c$) at nominal tracking (in red), the drone successfully traverses the window with an accuracy below 0.12 [m] for all the trajectories and in both the optimized and the non-optimized cases. However, the outcomes differ significantly for the perturbed simulations. Specifically, the non-optimized $INIT$ case displays greater deviations from the desired target location $r_d(t_w)$, when compared to the optimized OPT_a case, as expected.

This difference is particularly pronounced in the 2nd trajectory, which is more challenging (due to its higher accelerations). By referring to Fig. 4, which displays the distance to the desired target for both trajectories, we observe that the $INIT$ case shows a larger average deviation of 0.14 [m] for Traj1 and 0.3 [m] for Traj2, in contrast to the OPT_a cases with averages of 0.09 [m] for Traj1 and 0.12 [m] for Traj2. Furthermore, the $INIT$ case exhibits a significantly larger variance, with certain samples deviating 0.42 [m] in Traj1 and 0.75 [m] in Traj2. In contrast, OPT_a samples tend to cluster around the average. These findings emphasize the effectiveness of the optimization problem presented in (11), which is able to capture how variations in the parameters affect deviations in the considered states.

B. Experimental Results

After the validation of the optimization problem (11) in simulation, we performed real experiments using the Acanthis drone (see Fig. 5). Acanthis is an experimental

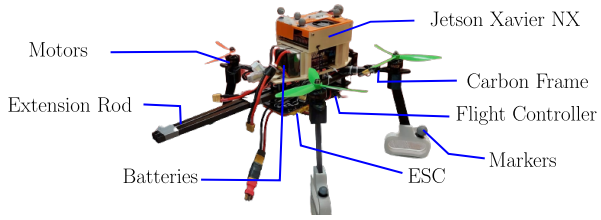

Fig. 5. Image of the experimental drone Acanthis.

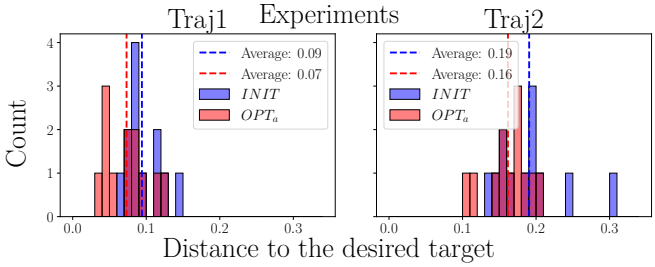

Fig. 6. Histograms of two trajectory types (left: Traj1, right: Traj2) for both cases ($INIT$ in blue, OPT_a in red) showing the distances to the desired target ($|\boldsymbol{r}_d(t_w) - \boldsymbol{r}(t_w)|$) in [m] and it's average across N_p experiments with Acanthis drone.

drone platform developed and maintained at INRIA/IRISA Rennes that utilizes the PX4 firmware. $N_p = 11$ perturbed experiments were performed by physically altering the system as depicted in Fig. 7. These perturbations were obtained by adding some tools and weights on the extension rod of the Acanthis drone. This allowed us to modify the set of parameters $\boldsymbol{p} = (k_f, g_x, g_y, g_z, m)$ at each experiment. In particular, the parameter k_f is affected mainly by the tools that we added under the motors that obstruct the airflow, while the other parameters are affected by the weights and location of the tools on the extension rod.

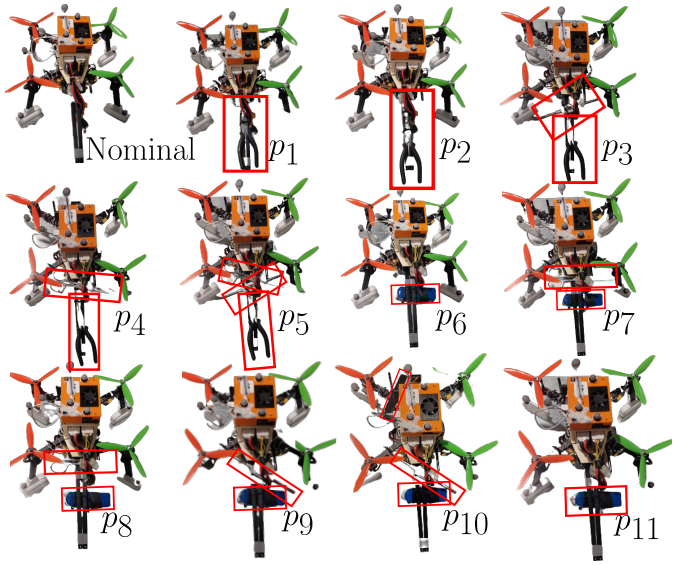

Fig. 7. Acanthis drone in its nominal state (top left), and subject to 11 distinct physical perturbations labeled as $p_1, ..., p_{11}$.

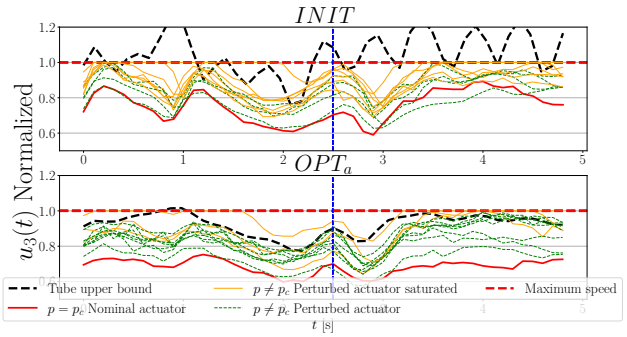

Fig. 8. Experimental actuator speed comparison for Traj2: $INIT$ (top) vs. OPT_a (bottom). The solid red line denotes nominal actuator speed, while the dashed black line signifies the input tube upper bound (eq. 10). The horizontal dashed red line represents the actuator limit. The dashed green and solid orange lines depict N_p perturbation runs as in Fig. 7: the solid orange lines indicate the cases in which the actuation was saturated, and the green dashed lines the cases in which no saturation occurred. Note how many fewer saturation cases are present in the OPT_a case vs. the $INIT$ case thanks to the use of the 'input tubes' in the constraints (11).

In the last two columns of Fig. 3, we present the results for Traj1 and Traj2 (1st and 2nd row, respectively), both for the optimized and non-optimized cases. Notably, it is evident from Traj1 that the deviation is more pronounced in the $INIT$ case. In fact, the point cloud – representing the different target reach locations $r(t_w)$ and denoted with red dots in Fig. 3 – is larger when compared to the OPT_a case. Fig. 6 provides insights for Traj1 in the experiments, where we find that the average distance to target for the $INIT$ case is 0.09 [m], while it reduces to 0.07 [m] for the OPT_a case. This reduction signifies an improvement of 2 [cm] on average. Additionally, the OPT_a case demonstrates less variance, further supporting its effectiveness.

Similar results are confirmed for Traj2. As it can be observed in Fig. 3, the deviation from the target is both larger and more spread in the $INIT$ case compared to the OPT_a case. Returning to Fig. 6, for Traj2, we find that the average distance to target is about 0.19 [m] for the $INIT$ case while it is around 0.16 [m] for the OPT_a case. However, it is important to note that the $INIT$ case exhibits a higher variance with deviations reaching up to 0.31 [m], whereas the maximum deviation in the OPT_a case is around 0.21 [m]. These results highlight the significant improvements in terms of the average distance to the target and the reduced spread achieved with the optimized OPT_a approach. In addition, Fig. 1 visualizes the flight logs data in PyBullet for Traj2. Collisions with the virtual window occurred 50% of the times in the $INIT$ case while they are reduced to 25% for the optimized OPT_a case across the 12 experiments. For a more detailed visualization, please refer to the attached video also available at this link.

To illustrate the impact of the input sensitivity on the constraints in the optimization problem (11), we refer to Fig. 8. This figure reports the normalized actuator speed $u_3(t)$ for Traj2 in real experiments, featuring both the $INIT$ case (top) and the OPT_a case (bottom). In the nominal case $\boldsymbol{p} = \boldsymbol{p}_c$ (in red), both $INIT$ and OPT_a maintained the actuator speeds well below their maximum limit, thus ensuring satisfactory reaching of the target. However, the

situation changes when the perturbations in Fig. 7 act on the drone. In the non-optimized $INIT$ case, a considerable number of experimental perturbed runs (in orange) quickly reached saturation of the maximum speed limit. In contrast, the OPT_a case displayed a quite better behavior, with only one of the N_p perturbations reaching saturation at the beginning (corresponding to p_8 in Fig. 7), potentially exceeding the predefined range of 10% deviation as per eq. (5). Furthermore, we observed that the actuator behavior (normalized speed) in the OPT_a case generally remained within the upper bound of the input tube (indicated by the dashed black line) obtained from the input sensitivity matrix (4). This highlights the significance of including input constraints (with their 'tubes') to ensure robustness against parameter variations of the inputs, particularly for aggressive trajectories such as Traj2.

V. CONCLUSIONS

In this paper, we have experimentally validated the optimization problem in which the effect of parametric uncertainties in a quadrotor model (quantified by the notion of *closed-loop state sensitivity*) can be minimized by acting on the reference trajectory to be tracked. We selected two distinct trajectories to be tracked by the UAV "Acanthis", introducing real-world physical perturbations to generate parameter uncertainties. The experimental results demonstrate the effectiveness of the proposed optimization in reducing the effects of uncertainties stemming from physical parameter perturbations, enabling more precise target attainment at relatively high speeds. We also showcased the significance of *input sensitivity* within the constraints of our optimization problem. Indeed, by using the input tube in the constraints, one can ensure the robustness of inputs against parametric uncertainty. Future work will consider the possibility of online replanning (e.g., within a MPC scheme) for runtime generation of robust trajectories during flight.

REFERENCES

[1] K. J. Astrom and B. Wittenmark, "Adaptive control 2nd edition," *Addison-Wesley Pub Co.*, vol. 1994, 1994.

[2] M. Achtelik, T. Bierling, J. Wang, L. Höcht, and F. Holzapfel, "Adaptive control of a quadcopter in the presence of large/complete parameter uncertainties," in *Infotech@ Aerospace 2011*, 2011, p. 1485.

[3] A. Haseltalab and R. R. Negenborn, "Adaptive control for autonomous ships with uncertain model and unknown propeller dynamics," *Control Engineering Practice*, vol. 91, p. 104116, 2019.

[4] K. Zhou and J. C. Doyle, *Essentials of robust control*. Prentice hall Upper Saddle River, NJ, 1998, vol. 104.

[5] R. Sanz, P. Garcia, Q.-C. Zhong, and P. Albertos, "Robust control of quadrotors based on an uncertainty and disturbance estimator," *Journal of Dynamic Systems, Measurement, and Control*, vol. 138, no. 7, p. 071006, 2016.

[6] S. Sun, A. Romero, P. Foehn, E. Kaufmann, and D. Scaramuzza, "A comparative study of nonlinear mpc and differential-flatness-based control for quadrotor agile flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3357–3373, 2022.

[7] D. Hanover, P. Foehn, S. Sun, E. Kaufmann, and D. Scaramuzza, "Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 690–697, 2021.

[8] A. Ansari and T. Murphey, "Minimum Sensitivity Control for Planning with Parametric and Hybrid Uncertainty," *The International Journal of Robotics Research (IJRR)*, vol. 35, no. 7, pp. 823–839, October 2016.

[9] S. Candido and S. Hutchinson, "Minimum Uncertainty Robot Path Planning using a POMDP Approach," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, December 2010, pp. 1408–1413.

[10] ——, "Minimum uncertainty robot navigation using information-guided pomdp planning," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 6102–6108.

[11] P. Robuffo Giordano, Q. Delamare, and A. Franchi, "Trajectory generation for minimum closed-loop state sensitivity," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 286–293.

[12] P. Brault, Q. Delamare, and P. Robuffo Giordano, "Robust trajectory planning with parametric uncertainties," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 095–11 101.

[13] C. Bohm, P. Brault, Q. Delamare, P. Robuffo Giordano, and S. Weiss, "Cop: Control & observability-aware planning," in *2022 IEEE Int. Conf. on Robotics and Automation*, 2022.

[14] P. Brault and P. Robuffo Giordano, "Tube-based trajectory optimization for robots with parametric uncertainty," submitted to IEEE Robotics and Automation Letters, http://rainbow-doc.irisa.fr/pdf/tube_based_traj_opt.pdf, 2023.

[15] S. Wasiela, P. Robuffo Giordano, J. Cortes, and T. Simeon, "A Sensitivity-Aware Motion Planner (SAMP) to Generate Intrinsically-Robust Trajectories," in *2023 IEEE Int. Conf. on Robotics and Automation*, 2023.

[16] A. Pupa, P. Robuffo Giordano, and C. Secchi, "Optimal energy tank initialization for minimum sensitivity to model uncertainties," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[17] A. Srour, A. Franchi, and P. Robuffo Giordano, "Controller and trajectory optimization for a quadrotor uav with parametric uncertainty," in *Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2023)*, Detroit, Michigan, United States, October 2023, uRL: https://hal.science/hal-04192321.

[18] F. Messerer and M. Diehl, "An Efficient Algorithm for Tube-based Robust Nonlinear Optimal Control with Optimal Linear Feedback," in *2021 60th IEEE Conference on Decision and Control (CDC)*, Dec. 2021, pp. 6714–6721, iSSN: 2576-2370.

[19] R. Mahony, V. Kumar, and P. Corke, "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor," *IEEE Robotics Automation Magazine (RAM)*, vol. 19, no. 3, pp. 20–32, Sep. 2012.

[20] G. Antonelli, E. Cataldi, F. Arrichiello, P. Robuffo Giordano, S. Chiaverini, and A. Franchi, "Adaptive trajectory tracking for quadrotor mavs in presence of parameter uncertainties and external disturbances," *IEEE Trans. on Control Systems Technology*, vol. 26, no. 1, pp. 248–254, 2018.

[21] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *International Journal of Robust and Nonlinear Control*, vol. 8, no. 11, pp. 995–1020, 1998.

[22] D. Brescianini, M. Hehn, and R. D'Andrea, "Nonlinear quadrocopter attitude control: Technical report," ETH Zurich, Tech. Rep., 2013.

[23] F. Zhou, B. Song, and G. Tian, "Bézier curve based smooth path planning for mobile robot," *Journal of Information & Computational Science*, vol. 8, no. 12, pp. 2441–2450, December 2011.

[24] A. R. Conn, K. Scheinberg, and P. L. Toint, "On the convergence of derivative-free methods for unconstrained optimization," *Approximation theory and optimization: tributes to MJD Powell*, pp. 83–108, 1997.

[25] G. Ansmann, "Efficiently and easily integrating differential equations with jitcode, jitcdde, and jitcsde," *Chaos: An interdisciplinary journal of nonlinear science*, vol. 28, no. 4, 2018.