Robust Trajectory Planning with Parametric Uncertainties

Pascal Brault, Quentin Delamare, Paolo Robuffo Giordano

Robust Trajectory Planning with Parametric Uncertainties

Pascal Brault†, Quentin Delamare† and Paolo Robuffo Giordano†

***Abstract*— In this paper we extend the previously introduced notion of *closed-loop state sensitivity* by introducing the concept of *input sensitivity* and by showing how to exploit it in a trajectory optimization framework. This allows to generate an optimal reference trajectory for a robot that minimizes the state and input sensitivities against uncertainties in the model parameters, thus producing inherently robust motion plans. We parametrize the reference trajectories with Béziers curves and discuss how to consider linear and nonlinear constraints in the optimization process (e.g., input saturations). The whole machinery is validated via an extensive statistical campaign that clearly shows the interest of the proposed methodology.**

I. INTRODUCTION

A lot of progress has been made in autonomous robotics over the recent years, however still much remains for attaining the needed level of precision in modern complex tasks. In fact, proposing solutions to overcome the uncertainties of the robot models (including actuation and sensing) and/or of the environment is not trivial. Some classical approches offer to estimate the model/environment parameters during the control task [1], to design robust controllers [2], or to exploit passivity-based methods [3]. However in most of these works, robustness goes against precision of the task execution. Other works have recently emerged, whose goal is to improve the performance of systems by means of trajectory optimization. Concretely, the main idea is to find trajectories whose optimization improves the state/parameters convergence speed or accuracy, see, e.g., [4]–[12]. But estimating the state/parameters online might introduce undesired transients and coupled estimation/dynamics that can be hard to manage. Another possibility consists in MPC, see, e.g., [13]: the principle is to exploit a dynamical model of the robot/environment inside the controller intern in order to anticipate the future. Since this method consists of rescheduling at each step time, in real time during a control task, it often requires a lot of computing power.

An alternative point of view is to focus on the generation of feedforward trajectories with minimal *state sensitivity*, see, e.g., [14]–[16], but with the main limitation of working only for 'open-loop' cases. To overcome this problem, in [17] it has been proposed to plan trajectories with minimal 'closed-loop' state sensitivity, which allows to take into account the coupling between the robot and its motion controller in an explicit way.

Building upon [17], in this work we introduce the novel notion of *input sensitivity*. To the best of our knowledge, this metric has never been considered before and we strongly think that both state and input sensitivities must be minimized in order to ensure the best possible performance.

Illustrating the necessity of the *input sensitivity* minimization in this framework, let us consider a mobile robot doing pick-up and drop-down operations in a factory. Since we have a perfect knowledge of the system in the *nominal* case where the control parameters match the real ones, a reference path can be chosen such that the input is physically feasible by the actuators. However in the *perturbed* case where the control parameters deviate from the real ones, if the input gap is too large because of this discrepancy, then a trajectory that was feasible in the *nominal* case might then be impossible to track: once the actuation limit are reached, the controller has no more margin on the inputs to correct the state. On top of that, one could imagine that the periodic motion is entirely known and that the actuators were chosen accordingly, thus electromechanically and thermally dimensioned. In the *perturbed* case, even if the actuation limits are not reached, if the tracking errors become greater than what was anticipated the input might be also greatly increased by the tracking controller, thus too much electrical current would be asked over a cycle, which could result in exceeding the average critical equivalent thermal torque of the actuators. This may obviously result in damage of the motors and the whole system on the long run. By understanding how security could be compromised because of a shift in the inputs, we find it very important to plan paths with minimum *input sensitivity* so that the change in the inputs between the *nominal* and *perturbed* cases would be minimal. A main contribution of this work is therefore to analyze how the input sensitivity can be optimized in conjuction with the state sensitivity to ensure a good tracking performance and minimal input deviations.

Similarly to [18]–[21] that deal with related topics, we also take into account actuator limitations in our formulation, thus showing that the various sensitivity quantities and related gradients can also be used in a nonlinear optimization context (opening the door to consider even more complex constraints). As opposed to classical approaches to robust planning/control, such as belief space [22] or robust MPC [23], which need to evaluate the worst-case deviation caused by uncertainty by large-scale sampling, here we address the robustness by increasing the predictability of inputs and outputs over a trajectory by minimizing a (much simpler) single sensitivity index.

In [17], the state sensitivity method was validated by simulating several perturbed cases for only one arbitrarily chosen trajectory. In this work, we conduct an upgraded statistical analysis in a large number of initial trajectories which are then individually perturbed for assessing the validity of our

† are with CNRS, Univ Rennes, Inria, IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France, e-mails: pascal.brault@irisa.fr, quentin.delamare@irisa.fr, prg@irisa.fr.

method.

This article is structured as follows: in Sect. III we derive the closed-loop sensitivity metrics w.r.t. parameters. Then in Sect. IV we explain how one can minimize those metrics by defining several linear and nonlinear constrained optimization problems. This framework is tested for a planar quadrotor (Sect. V) in an extensive campaign of perturbed simulations in Sect. VI: the analysis of the results demonstrates the improvement in *closed-loop* performance when minimizing *sensitivity metrics* along the trajectories. Sect. VII concludes this paper, and opens to further perspectives.

II. PRELIMINARY STATEMENT

As in [17], we begin by defining the main equations characterizing an autonomous system whose performance we want to improve. First, we consider a general dynamical model for representing a robot behavior

$$\dot{\boldsymbol{q}} = \boldsymbol{f}(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{p}), \qquad \boldsymbol{q}(0) = q_0, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^{n_q}$ is the state of the system, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ its inputs, and $\boldsymbol{p} \in \mathbb{R}^{n_p}$ the vector of the model parameters (mass, inertia, etc.), which directly affects the system dynamics.

We now define $\boldsymbol{r}_d(\boldsymbol{a}, t) \in \mathbb{R}^{n_r}$ as a desired motion task to be tracked by some variables of interest of the system, represented by the output function $\boldsymbol{r}(\boldsymbol{q}) \in \mathbb{R}^{n_r}$, $n_r \leqslant n_q$. Vector $\boldsymbol{a} \in \mathbb{R}^{n_a}$ is the parameter that shapes the trajectory and $t \in [0, T] = \mathbb{T}$ is the time. Let us assume that a controller exists for tracking the desired motion. For generality we consider possible internal states $\boldsymbol{\xi} \in \mathbb{R}^{n_\xi}$ that may represent, e.g., an integral action of dynamic extensions. Thereby, we define the control law as

$$\begin{cases} \dot{\boldsymbol{\xi}} = \boldsymbol{g}\left(\boldsymbol{\xi}, \boldsymbol{q}, \boldsymbol{a}, \boldsymbol{p}_c, \boldsymbol{k}_c\right), & \boldsymbol{\xi}(0) = \xi_0 \\ \boldsymbol{u} = \boldsymbol{h}\left(\boldsymbol{\xi}, \boldsymbol{q}, \boldsymbol{a}, \boldsymbol{p}_c, \boldsymbol{k}_c\right) \end{cases}, \tag{2}$$

where $\boldsymbol{k}_c \in \mathbb{R}^{n_k}$ is the controller gains vector and $\boldsymbol{p}_c \in \mathbb{R}^{n_p}$ is the *nominal* parameters vector: most of the time, there is little chance that the control loop parameters, i.e. $\boldsymbol{p}_c$, match the 'real' parameters, i.e. $\boldsymbol{p}$, since the accuracy of the system model is limited.

Knowing (1–2) and by using the same reasoning as in [17], we can now define and compare two different cases which highlight one of the typical issues in robot control. On the first hand, in the *nominal* case where $\boldsymbol{p} = \boldsymbol{p}_c$, the controller is able to perform the tracking task with utmost accuracy, delivering its best closed-loop performance, with the smallest error possible $\boldsymbol{e}(t) = \boldsymbol{r}_d(\boldsymbol{a}, t) - \boldsymbol{r}(\boldsymbol{q})$. On the other hand, in the *perturbed* case where $\boldsymbol{p} \neq \boldsymbol{p}_c$, the dynamics given to the controller $\boldsymbol{f}\left(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{p}_c\right)$ differs from $\boldsymbol{f}\left(\boldsymbol{q}, \boldsymbol{u}, \boldsymbol{p}\right)$ and the tracking task is done on a system that does not match the reality. With this lack of knowledge, the closed-loop behaviour will perform worse, resulting in a tracking error $\boldsymbol{e}(t)$ that might be larger than in the previous *nominal* case.

III. CLOSED-LOOP SENSITIVITY METRICS

In this section we explain how to compute the main quantities needed to describe how sensitive a system is during a control task. The computation of the *closed-loop sensitivities* is required for the optimization process in the cost function. We also show how the associated gradients w.r.t. the optimization vector $\boldsymbol{a}$ can be obtained. Many of these derivations are presented in [17]: we summarize here the main notions for the reader's convenience and adapt them to the particular problem considered in this work.

A. Definitions

First we define the *state sensitivity*

$$\boldsymbol{\Pi}(t) = \left.\frac{\partial \boldsymbol{q}(t)}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_q \times n_p} \tag{3}$$

which represents the evolution of the state w.r.t. variations in the parameter vector $\boldsymbol{p}$, and is evaluated on the *nominal* value $\boldsymbol{p} = \boldsymbol{p}_c$. This quantity has already been introduced in [17], thereby we shorlty recall its use for this work. To improve the system behaviour in presence of parameter inaccuracies, one can minimize some norm of $\boldsymbol{\Pi}(t)$ w.r.t. the optimization variables $\boldsymbol{a}$. An optimal shape of the trajectory $\boldsymbol{r}_d(\boldsymbol{a}, t)$ with a minimal state sensitivity would make the *closed-loop* state evolution $\boldsymbol{q}(t)$ in the *perturbed* case as close as possible to its evolution for the *nominal* case.

We introduce in this work the novel notion of *input sensitivity*

$$\boldsymbol{\Theta}(t) = \left.\frac{\partial \boldsymbol{u}(t)}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_u \times n_p} \tag{4}$$

which quantifies the amount of variations that would occur on the inputs w.r.t. deviations in $\boldsymbol{p}$. This is also evaluated for $\boldsymbol{p} = \boldsymbol{p}_c$. As explained in Sect. I, the main motivation for considering this metric is that the discrepancy between the control parameters $\boldsymbol{p}_c$ and the true ones $\boldsymbol{p}$ may result in some undesired inputs variation: in any system, components such as actuators are specifically chosen for the desired application, hence they need to be operated as close as possible to the conditions they were designed for.

B. Numerical integration

In the general case, it is not possible to compute a closed-form of $\boldsymbol{\Pi}(t)$, however it is possible to differentiate (3) over time, which yields

$$\dot{\boldsymbol{\Pi}}(t) = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{q}}\boldsymbol{\Pi} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\boldsymbol{\Theta} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{p}}, \qquad \boldsymbol{\Pi}(0) = 0, \tag{5}$$

where $\boldsymbol{f}$ is the system dynamics (1).

Integrating (5) is not obvious because $\boldsymbol{\Theta}(t)$ is not known *a priori*. However, using the expression of $\boldsymbol{u}(t)$ in (2), we can rewrite (4) as

$$\boldsymbol{\Theta}(t) = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{q}}\boldsymbol{\Pi} + \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\xi}}\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{p}}. \tag{6}$$

There is still an unknown term, the *internal state sensitivity* that we denote as

$$\boldsymbol{\Pi}_{\boldsymbol{\xi}}(t) = \left.\frac{\partial \boldsymbol{\xi}(t)}{\partial \boldsymbol{p}}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_\xi \times \boldsymbol{n}_p}. \tag{7}$$

Derivation of (7) leads to

$$\dot{\boldsymbol{\Pi}}_{\boldsymbol{\xi}}(t) = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{q}}\boldsymbol{\Pi} + \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{\xi}}\boldsymbol{\Pi}_{\boldsymbol{\xi}}, \qquad \boldsymbol{\Pi}_{\boldsymbol{\xi}}(0) = 0. \tag{8}$$

Equations (5), (6) and (8) can be regrouped in one single set of differential equations, which allows to compute the *state and input sensitivities*. For the sake of readability, we introduce the notation $\boldsymbol{x}_{,\boldsymbol{y}}$ in order to refer to the jacobian of a vector function $\boldsymbol{x}$ w.r.t. one of its arguments $\boldsymbol{y}$. With this shorthand, the set of differential equations becomes

$$\begin{cases} \dot{\boldsymbol{\Pi}}(t) = \boldsymbol{f}_{,\boldsymbol{q}}\boldsymbol{\Pi} + \boldsymbol{f}_{,\boldsymbol{u}}\boldsymbol{\Theta} + \boldsymbol{f}_{,\boldsymbol{p}}, & \boldsymbol{\Pi}(0) = 0, \\ \dot{\boldsymbol{\Pi}}_{\boldsymbol{\xi}}(t) = \boldsymbol{g}_{,\boldsymbol{q}}\boldsymbol{\Pi} + \boldsymbol{g}_{,\boldsymbol{\xi}}\boldsymbol{\Pi}_{\boldsymbol{\xi}}, & \boldsymbol{\Pi}_{\boldsymbol{\xi}}(0) = 0, \\ \boldsymbol{\Theta}(t) = \boldsymbol{h}_{,\boldsymbol{q}}\boldsymbol{\Pi} + \boldsymbol{h}_{,\boldsymbol{\xi}}\boldsymbol{\Pi}_{\boldsymbol{\xi}}. \end{cases} \tag{9}$$

Thereby one can obtain the evolutions of $\boldsymbol{\Pi}$, $\boldsymbol{\Pi}_{\xi}$ and $\boldsymbol{\Theta}$ over time. Since the real parameters of the system $\boldsymbol{p}$ are not known, it is impossible to integrate this set of differential equations to evaluate the sensitivities w.r.t. the true parameters. That being said, one can evaluate these quantities at $\boldsymbol{p}_c$ instead. $\boldsymbol{\Pi}(\boldsymbol{p}_c)$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}}(\boldsymbol{p}_c)$ and $\boldsymbol{\Theta}(\boldsymbol{p}_c)$ are still close enough from $\boldsymbol{\Pi}(\boldsymbol{p})$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}}(\boldsymbol{p})$ and $\boldsymbol{\Theta}(\boldsymbol{p})$ respectively, as we assume that $\boldsymbol{p}_c$ is a good approximation of $\boldsymbol{p}$, and, indeed, the validity of this assumption will be confirmed in the extensive tests of Sect. VI.

C. Gradient derivation

As in [17], we now show how to obtain $\partial\boldsymbol{\Pi}/\partial\boldsymbol{a}$ and $\partial\boldsymbol{\Theta}/\partial\boldsymbol{a}$ that are respectively, the gradients of $\boldsymbol{\Pi}$ and $\boldsymbol{\Theta}$ w.r.t. $\boldsymbol{a}$. This subsection is an updated version of the gradients computation, in which $\partial\boldsymbol{\Theta}/\partial\boldsymbol{a}$ is also treated. Since $\boldsymbol{\Pi}$ and $\boldsymbol{\Theta}$ are matrices, their gradients are tensors: for simplicity, we express each component of the gradient $\partial\boldsymbol{\Pi}/\partial a_i$ w.r.t. each individual i-th component of a_i. Let then

$$\boldsymbol{\Pi}_{a_i}(t) = \left.\frac{\partial\boldsymbol{\Pi}(t)}{\partial a_i}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_q \times n_p}, \tag{10}$$

$$\boldsymbol{\Pi}_{\boldsymbol{\xi}_{a_i}}(t) = \left.\frac{\partial\boldsymbol{\Pi}_{\boldsymbol{\xi}}(t)}{\partial a_i}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_\xi \times n_p}, \tag{11}$$

$$\boldsymbol{\Theta}_{a_i}(t) = \left.\frac{\partial\boldsymbol{\Theta}(t)}{\partial a_i}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_u \times n_p} \tag{12}$$

be the gradients (matrices) of respectively $\boldsymbol{\Pi}$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}}$ and $\boldsymbol{\Theta}$ w.r.t. a_i. Analogously to $\boldsymbol{\Pi}_{\boldsymbol{\xi}}$, the quantity $\boldsymbol{\Pi}_{\boldsymbol{\xi}_{a_i}}$ is introduced to evaluate $\boldsymbol{\Pi}_{a_i}$. We also define

$$\boldsymbol{\Gamma}_{\boldsymbol{q}_i}(t) = \left.\frac{\partial\boldsymbol{q}(t)}{\partial a_i}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_q}, \tag{13}$$

$$\boldsymbol{\Gamma}_{\boldsymbol{\xi}_i}(t) = \left.\frac{\partial\boldsymbol{\xi}(t)}{\partial a_i}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_\xi}, \tag{14}$$

$$\boldsymbol{\Gamma}_{\boldsymbol{u}_i}(t) = \left.\frac{\partial\boldsymbol{u}(t)}{\partial a_i}\right|_{\boldsymbol{p}=\boldsymbol{p}_c} \in \mathbb{R}^{n_u} \tag{15}$$

respectively as the gradients of the *system states*, the *internal states* and the *inputs* w.r.t. changes in a_i, which are also necessary to evaluate $\boldsymbol{\Pi}_{a_i}$. It is possible to compute these quantities along the whole trajectory by the same reasoning as in (9), resulting in

$$\begin{cases} \dot{\boldsymbol{\Gamma}}_{\boldsymbol{q}_i} = \boldsymbol{f}_{,\boldsymbol{q}}\boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{f}_{,\boldsymbol{u}}\boldsymbol{\Gamma}_{\boldsymbol{u}_i}, & \boldsymbol{\Gamma}_{\boldsymbol{q}_i}(0) = 0, \\ \dot{\boldsymbol{\Gamma}}_{\boldsymbol{\xi}_i} = \boldsymbol{g}_{,\boldsymbol{q}}\boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{g}_{,\boldsymbol{\xi}}\boldsymbol{\Gamma}_{\boldsymbol{\xi}_i} + \boldsymbol{g}_{,a_i}, & \boldsymbol{\Gamma}_{\boldsymbol{\xi}_i}(0) = 0, \\ \boldsymbol{\Gamma}_{\boldsymbol{u}_i} = \boldsymbol{h}_{,\boldsymbol{q}}\boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{h}_{,\boldsymbol{\xi}}\boldsymbol{\Gamma}_{\boldsymbol{\xi}_i} + \boldsymbol{h}_{,a_i}, \end{cases} \tag{16}$$

which allows us to compute $\boldsymbol{\Gamma}_{\boldsymbol{q}_i}$, $\boldsymbol{\Gamma}_{\boldsymbol{\xi}_i}$ and $\boldsymbol{\Gamma}_{\boldsymbol{u}_i}$ by forward integration.

Let now $\boldsymbol{x}_{,\boldsymbol{y},\boldsymbol{z}}$ refer to the second order jacobian (tensor) of the jacobian function $\boldsymbol{x}_{,\boldsymbol{y}}$ w.r.t. one of its arguments $\boldsymbol{z}$. Also let $\boldsymbol{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a tensor and $\boldsymbol{v} \in \mathbb{R}^{n_3}$ be a vector, we define

$$(\boldsymbol{T} \circ \boldsymbol{v})_{i,j} = \sum_{k=1}^{n_3} t_{i,j,k} v_k, \quad \forall i \leqslant n_1, j \leqslant n_2.$$

Differentiating (9) w.r.t. a_i with those notations yields

$$\begin{cases} \dot{\boldsymbol{\Pi}}_{a_i} = (\boldsymbol{f}_{,\boldsymbol{q},\boldsymbol{q}} \circ \boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{f}_{,\boldsymbol{q},\boldsymbol{u}} \circ \boldsymbol{\Gamma}_{\boldsymbol{u}_i})\,\boldsymbol{\Pi} + \boldsymbol{f}_{,\boldsymbol{q}}\boldsymbol{\Pi}_{a_i} + \\ \qquad (\boldsymbol{f}_{,\boldsymbol{u},\boldsymbol{q}} \circ \boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{f}_{,\boldsymbol{u},\boldsymbol{u}} \circ \boldsymbol{\Gamma}_{\boldsymbol{u}_i})\,\boldsymbol{\Theta} + \boldsymbol{f}_{,\boldsymbol{u}}\boldsymbol{\Theta}_{a_i} + \\ \qquad (\boldsymbol{f}_{,\boldsymbol{p},\boldsymbol{q}} \circ \boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{f}_{,\boldsymbol{p},\boldsymbol{u}} \circ \boldsymbol{\Gamma}_{\boldsymbol{u}_i}) \\ \dot{\boldsymbol{\Pi}}_{\boldsymbol{\xi}_{a_i}} = (\boldsymbol{g}_{,\boldsymbol{\xi},\boldsymbol{\xi}} \circ \boldsymbol{\Gamma}_{\boldsymbol{\xi}_i} + \boldsymbol{g}_{,\boldsymbol{\xi},\boldsymbol{q}} \circ \boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{g}_{,\boldsymbol{\xi},a_i})\,\boldsymbol{\Pi}_{\boldsymbol{\xi}} + \\ \qquad (\boldsymbol{g}_{,\boldsymbol{q},\boldsymbol{\xi}} \circ \boldsymbol{\Gamma}_{\boldsymbol{\xi}_i} + \boldsymbol{g}_{,\boldsymbol{q},\boldsymbol{q}} \circ \boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{g}_{,\boldsymbol{q},a_i})\,\boldsymbol{\Pi} + \\ \qquad \boldsymbol{g}_{,\boldsymbol{\xi}}\boldsymbol{\Pi}_{\boldsymbol{\xi}_{a_i}} + \boldsymbol{g}_{,\boldsymbol{q}}\boldsymbol{\Pi}_{a_i} \\ \boldsymbol{\Theta}_{a_i} = (\boldsymbol{h}_{,\boldsymbol{\xi},\boldsymbol{\xi}} \circ \boldsymbol{\Gamma}_{\boldsymbol{\xi}_i} + \boldsymbol{h}_{,\boldsymbol{\xi},\boldsymbol{q}} \circ \boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{h}_{,\boldsymbol{\xi},a_i})\,\boldsymbol{\Pi}_{\boldsymbol{\xi}} + \\ \qquad (\boldsymbol{h}_{,\boldsymbol{q},\boldsymbol{\xi}} \circ \boldsymbol{\Gamma}_{\boldsymbol{\xi}_i} + \boldsymbol{h}_{,\boldsymbol{q},\boldsymbol{q}} \circ \boldsymbol{\Gamma}_{\boldsymbol{q}_i} + \boldsymbol{h}_{,\boldsymbol{q},a_i})\,\boldsymbol{\Pi} + \\ \qquad \boldsymbol{h}_{,\boldsymbol{\xi}}\boldsymbol{\Pi}_{\boldsymbol{\xi}_{a_i}} + \boldsymbol{h}_{,\boldsymbol{q}\boldsymbol{\Pi}_{a_i}} \end{cases} \tag{17}$$

with the initial conditions $\boldsymbol{\Pi}_{a_i}(0) = 0$ and $\boldsymbol{\Pi}_{\boldsymbol{\xi}_{a_i}}(0) = 0$.

To sum up, one can now feed the gradients $\boldsymbol{\Pi}_{a_i}$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}_{a_i}}$ and $\boldsymbol{\Theta}_{a_i}$ w.r.t. $a_{i \in [\![1, n_a]\!]}$ to the optimizer by forward integrating both (16) and (17), in order to obtain $\boldsymbol{\Gamma}_{\boldsymbol{q}_i}$, $\boldsymbol{\Gamma}_{\boldsymbol{\xi}_i}$ and $\boldsymbol{\Gamma}_{\boldsymbol{u}_i}$, then $\boldsymbol{\Pi}_{a_i}$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}_{a_i}}$ and $\boldsymbol{\Theta}_{a_i}$ along the trajectory. Computing $\boldsymbol{\Pi}$, $\boldsymbol{\Pi}_{\boldsymbol{\xi}}$ and $\boldsymbol{\Theta}$ with (9) gives the current values of the sensitivities, and the gradients indicate the direction ensuring a reduction of the metrics.

D. Trajectory representation

We consider Bézier curves to specify the shape of our system motion, see, e.g., [24]. To this end, let $P_{i \in [\![0, n_a - 1]\!]} \in \mathbb{R}^{n_r}$ be a control point of the robotic path, and also let $s = t/T \in [0, 1]$ be the normalized time, then the associated Bézier curve is the set of points defined by the parametric representation

$$\boldsymbol{\mathcal{P}}(s) = \sum_{i=0}^{n_a - 1} \binom{n_a - 1}{i} s^i (1 - s)^{n_a - 1 - i} \cdot P_i. \tag{18}$$

This class of curves presents interesting properties compared to 'plain polynomials': indeed, during the trajectory optimization, a little modification of a polynomial coefficient can have a strong influence on the whole curve, which may lead to numerical instabilities. On the other hand, the displacement of a control point of a Béziers curve in its admissible space is less impactful, thus ensuring a better overall behaviour. Besides, thanks to its nature, every point of a Bézier curve is inside the convex envelop of the control points, which can be very useful for collision avoidance.

IV. TRAJECTORY PLANNING

To enhance the global performance of a robot, we can consider two different problems: knowing our system dynamics

$\boldsymbol{f}$ refering to (1), a reference trajectory $\boldsymbol{r}_d(\boldsymbol{a}, t)$ defined over the time interval $\mathbb{T}$, the controller internal dynamics $\boldsymbol{g}$ and its input function $\boldsymbol{h}$ both defined in (2), the optimization consists in finding the optimal vector $\boldsymbol{a}_{opt}$, such that

$$\boldsymbol{a}_{opt} = \arg\min_{\boldsymbol{a}\in\mathcal{A}} \omega_1 ||\boldsymbol{\Pi}(T)||^2 + \omega_2 \int_0^T ||\boldsymbol{\Theta}(\tau)||^2 \,\mathrm{d}\tau, \quad (19)$$

where $||\cdot||$ is a suitable norm for $\boldsymbol{\Pi}$ and $\boldsymbol{\Theta}$, we chose to use ω_1 and ω_2 as weights for the optimization, whose use is described afterwards (other possibilities exist, such as Pareto optimality [25]), and $\mathcal{A}$ is the set of possible values for $\boldsymbol{a}$. Problem (19) aims at minimizing the final output error $\boldsymbol{e_r}(\boldsymbol{a}, T)$ as well as the integral tracking input error $\boldsymbol{e_u}(t)\ \forall t \in \mathbb{T}$, in the perturbed case. This task is relevant when needing to reach an accurate pose.

Reducing the input deviation by minimizing $||\boldsymbol{\Theta}||$ in addition to $||\boldsymbol{\Pi}||$ allows the system to follow optimal paths during which the inputs can approach their saturations $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$, while reducing the chance to reach them. If we optimize to only reduce $||\boldsymbol{\Pi}||$, $||\boldsymbol{\Theta}||$ might be greater after optimization, which will result in a desired task that the system will not be able to achieve in the perturbed case if the inputs deviate too much and run into the actuation saturations. Thereby, there would be higher probability to get a bad behaviour from this 'optimized' motion.

For the sake of finding a motion that will guarantee better performance of the system, we chose to break down problem (19) into several related sub-problems defined by the weighting vector $\boldsymbol{\Omega} = (\omega_1, \omega_2)$, used as follows:

- $\boldsymbol{\Omega_\Pi} = (1, 0)$ allocates the whole optimization for the state sensitivity, and gives $\boldsymbol{a}_{\boldsymbol{\Pi}_{opt}}$;
- $\boldsymbol{\Omega_\Theta} = (0, 1)$ aims at only diminishing the input deviation during the control task and results in $\boldsymbol{a}_{\boldsymbol{\Theta}_{opt}}$;
- $\boldsymbol{\Omega}_W = \left(\frac{1}{||\boldsymbol{\Pi}_{opt}||}, \frac{1}{||\boldsymbol{\Theta}_{opt}||}\right)$ allows the optimizer to reduce both the state and input sensitivities at the same time, such as : it normalizes both $||\boldsymbol{\Pi}||$ and $||\boldsymbol{\Theta}||$ costs that are not similar metrics by nature, and also grants more weight to the sensitivity that has the lowest value after its first minimization. This way, the last case should be the one that will give overall the smallest errors for both the state and input. This optimization outputs $\boldsymbol{a}_{W_{opt}}$.

Summarizing, the described constrained minimization problems can be solved by any suitable optimizer. Since we are able to compute the gradients of the metrics studied in Sect. III, a gradient descent algorithm with linear and nonlinear constraints will be used.

Let us consider a case where the initial and final values of $\boldsymbol{r}_d(\boldsymbol{a}, t)$ and their needed time derivatives are given to the optimizer, these linear constraints impose the admissible set $\mathcal{A}$ such that $\boldsymbol{M}\boldsymbol{a} = \boldsymbol{d}$, where $\boldsymbol{M}$ depends on the trajectory representation curve, and $\boldsymbol{d}$ contains the motion limit conditions. Since any robot necessarily has actuation limits $\boldsymbol{u}_{min}$ and $\boldsymbol{u}_{max}$, these physical capabilities must be considered as nonlinear constraints. Vector $\boldsymbol{a}$ can then be optimized with any constrained nonlinear optimization routine (for instance, we used the well-known 'fmincon' function in Matlab), starting from an initial guess $\boldsymbol{a}_0$ satisfying both the linear and nonlinear constraints, e.g.,

$$\begin{aligned} \boldsymbol{M}\boldsymbol{a}_0 &= \boldsymbol{d}, \\ \forall t \in \mathbb{T},\ \boldsymbol{u}_{min} &\leqslant \boldsymbol{u}(t) \leqslant \boldsymbol{u}_{max}. \end{aligned} \quad (20)$$

Later, the optimization routine can be halted with standard termination criteria (e.g., based on the gradient norms). Note that since problem (19) is in general non-convex in $\boldsymbol{a}$, the optimization algorithm can only guarantee convergence towards a local minimum.

Since we need to evaluate the costs $||\boldsymbol{\Pi}||$ and $||\boldsymbol{\Theta}||$, a suitable norm choice needs to be made. In this work we chose the Frobenius matrix norm which derives from the scalar product associated with its matrix space $\mathbb{R}^{m\times n}$, i.e., for a matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$,

$$||\boldsymbol{M}||^2 = \mathrm{Tr}\left(\boldsymbol{M}^T\boldsymbol{M}\right) = \sum_{i,j} m_{i,j}^2. \quad (21)$$

V. APPLICATION TO A PLANAR QUADROTOR

Let us now summarize what is already detailed in [17] with a few minor changes, in order to get the expressions of (1–2) for a planar quadrotor.

We consider $\mathcal{F}_B = \{\boldsymbol{O}_B, \boldsymbol{x}_B, \boldsymbol{z}_B\}$ as the body frame attached to the the quadrotor center of mass, with z_B aligned with the thrust direction, see Fig. 1. For the planar quadrotor, the state consists of the cartesian position (x, z) as well as its first time derivative, the linear velocity (v_x, v_z), both expressed in the world frame $\mathcal{F}_W = \{\boldsymbol{O}_W, \boldsymbol{x}_W, \boldsymbol{z}_W\}$, and of the body orientation $\theta = (\boldsymbol{z}_W, \boldsymbol{z}_B)$ as well as the angular velocity $\omega = \dot{\theta}$, therefore $\boldsymbol{q} = \begin{bmatrix}\boldsymbol{r}^T\ \boldsymbol{v}^T\ \theta\ \omega\end{bmatrix}^T \in \mathbb{R}^6$. Let (f, τ) be respectively the total thrust and torque of the quadrotor, we can distinguish these effective inputs and the actual inputs (ω_L, ω_R), being the left/right propeller rotation rates. These four values are related by

$$\begin{bmatrix} f \\ \tau \end{bmatrix} = \begin{bmatrix} k_f & k_f \\ k_\tau & -k_\tau \end{bmatrix} \begin{bmatrix} \omega_R^2 \\ \omega_L^2 \end{bmatrix} = \boldsymbol{T} \begin{bmatrix} \omega_R^2 \\ \omega_L^2 \end{bmatrix}, \quad (22)$$

where k_f and k_τ are calibration parameters depending on the propeller characteristics, see e.g. [26]. We will then take $\boldsymbol{u} = \begin{bmatrix}\omega_L^2\ \omega_R^2\end{bmatrix}^T$ as the quadrotor control inputs throughout the following developments.

Let the planar quadrotor dynamical model be

$$\begin{cases} \dot{\boldsymbol{r}} = v \\ \dot{\boldsymbol{v}} = \begin{bmatrix} 0 \\ -g \end{bmatrix} + \dfrac{f}{m} \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} \\ \dot{\theta} = \omega \\ \dot{\omega} = \dfrac{\tau}{I} \end{cases} \quad (23)$$

where m and I are respectively the quadrotor mass and inertia, and g is the gravity acceleration magnitude. The relation between (f, τ) and (ω_L, ω_R) implies that the dynamics are not only affected by m and I, but also by k_f and k_τ. Therefore, we take $\boldsymbol{p} = [m\ I\ k_f\ k_\tau] \in \mathbb{R}^4$ as the parameters vector for the sensitivities reduction.

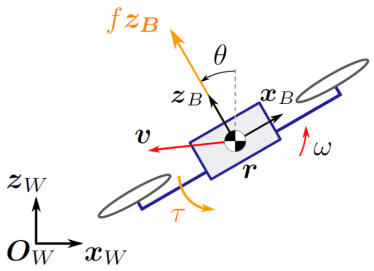

Fig. 1. Illustration of the main quantities characterizing the planar quadrotor model.

The chosen control task is that of letting the quadrotor output $\boldsymbol{r}(q) = [x\ z]^T \in \mathbb{R}^2$ track a desired motion $\boldsymbol{r}_d(\boldsymbol{a}, t) \in \mathbb{R}^2$. This is done by implementing a DFL (Direct Feedback Linearization) controller with integral term for the best performance in the nominal case $\boldsymbol{p}_c = \boldsymbol{p}$, see, e.g. [27]. For the quadrotor, $\boldsymbol{\xi} = [\xi_f\ \xi_{df}\ \xi_x\ \xi_z] \in \mathbb{R}^4$ is the controller internal states vector, where ξ_f and ξ_{df} are the dynamic extensions of f, and (ξ_x, ξ_z) are the states of the integral action. The dynamics of the controller internal states are

$$\begin{bmatrix} \dot{\xi}_f \\ \dot{\xi}_{df} \\ \dot{\boldsymbol{\xi}}_{xz} \end{bmatrix} = \begin{bmatrix} \xi_{df} \\ \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{A}^{-1}(\boldsymbol{\eta} - \boldsymbol{b}) \\ \boldsymbol{r}_d - \boldsymbol{r} \end{bmatrix} = \boldsymbol{g}(\boldsymbol{\xi}, \boldsymbol{q}, \boldsymbol{a}, \boldsymbol{p}_c, \boldsymbol{k}_c) \tag{24}$$

and the quadrotor control inputs can be written as

$$\boldsymbol{u} = \boldsymbol{T}_c^{-1} \begin{bmatrix} \xi_f \\ \begin{bmatrix} 0 & 1 \end{bmatrix} \boldsymbol{A}^{-1}(\boldsymbol{\eta} - \boldsymbol{b}) \end{bmatrix} = \boldsymbol{h}(\boldsymbol{\xi}, \boldsymbol{q}, \boldsymbol{a}, \boldsymbol{p}_c, \boldsymbol{k}_c) \tag{25}$$

where the expression of $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{\eta}$ which is an adjusted linear combination of the feedforward and feedback terms, are detailed in [27], and $\boldsymbol{k}_c \in (\mathbb{R}_*^+)^5$ are suitable control gains. More explicit information on how the controller is designed for this task can be found in [17].

VI. STATISTICAL ANALYSIS

In order to test the effectiveness of the previously described method, we conducted a new statistical analysis of larger scale than in [17], which aims at testing the soundness of (19) when applied to various trajectories. The idea of this analysis is to generate a set of N_{traj} non-optimized trajectories and corresponding $\boldsymbol{a}_{\boldsymbol{\Pi}_{opt}}$, $\boldsymbol{a}_{\boldsymbol{\Theta}_{opt}}$ and $\boldsymbol{a}_{W_{opt}}$ on which we would like to test the framework, and then to evaluate the resulting performance for each trajectory case, by means of statistical analysis of the dynamical behaviour against N_{sim} parameter perturbations.

Concretely, the first phase of trajectory generation is done by picking a random target in a right half disc from the origin (as there is a spatial symmetry of the quadrotor dynamics w.r.t. the initial position), $\boldsymbol{r}_d(T) = \rho(\cos(\phi), \sin(\phi))$ where $-\frac{\pi}{2} \leqslant \phi \leqslant \frac{\pi}{2}$ [rad] and $1 \leqslant \rho \leqslant 3$ [m]. The boundary conditions are set to zero, what forces the quadrotor heading to be vertical at the initial and final points of this rest-to-rest desired motion. We add some actuation saturation on the total thrust f of the quadrotor, such that $0 < f < 2mg$, translated into propellers rotation rates, by using the inverse mapping of (22). The lower limit condition originates from the DFL controller structure, which is singular for $\xi_f = 0$. The upper limit keeps the inputs below some saturation to represent a maximum propeller speed.

For each initial guess $\boldsymbol{a}_0$ generated according to these rules, the optimizer outputs the associated optimized motions. Afterwards, we run $N_{sim} = 500$ simulations of the quadrotor tracking these four trajectories, while randomly drawing all the parameters $\boldsymbol{p}$ from a uniform distribution with a perturbation of $\delta_{\boldsymbol{p}} = 10\%$. From these $4N_{sim}$ simulations we can measure the final output error square norm

$$\boldsymbol{E}_r = ||\boldsymbol{r}_d(T) - \boldsymbol{r}(T)||^2 \tag{26}$$

and the average input error square norm

$$\boldsymbol{E}_{\boldsymbol{u}} = \int_0^T ||\boldsymbol{u}_{\boldsymbol{p}_c}(\tau) - \boldsymbol{u}_{\boldsymbol{p}}(\tau)||^2 \, \mathrm{d}\tau. \tag{27}$$

Then, on each of these eight resulting sets of error values, we compute the mean value and the standard deviation. As a synthesis, starting from a single non-optimized trajectory we end up with sixteen numbers, namely the means $\mu_{\boldsymbol{r},\boldsymbol{u}}$ and the standard deviations $\sigma_{\boldsymbol{r},\boldsymbol{u}}$ of the output and the input errors, for all four trajectories.

Finally, we aggregate these numbers over the whole set of $N_{traj} = 30$ trajectories, by computing the boxplot characteristics of these means and standard deviations. In other words, for every initial trajectory, we compute the median, the first and third quartiles, and the first and last centiles of the N_{traj} error means, and standard devations, for all four non-optimized and optimized trajectories. We have done this whole campaign of optimizations and perturbed simulations for two controller cases : one set with no integral term ($k_i = 0$, case NI) and a second one with an integral term ($k_i > 0$, case I). Anyhow, the control gains have been chosen in order to give real and negatives closed-loop poles.

Fig. 2 shows the resulting boxplots of a full campaign of $N_{sim} = 500$ simulations for $N_{traj} = 30$ initial trajectories for the DFL with no integral term (case NI), displaying only $\mu_{\boldsymbol{r},\boldsymbol{u}}$. In the top row ($\boldsymbol{\Pi}_{opt}$), we can see a slight improvement of the error means height (left). However, we can see in the same row (right) that the boxplot of the input tracking error means is higher and spreads way more than the initial guess: this means that the generation of a minimal state sensitivity trajectory that will statistically reduce the output tracking error can also have the side effect of making the input less predictable, and more subject to variations due to parameters error. For the middle row ($\boldsymbol{\Theta}_{opt}$), we observe a slight decrease in the input error means (right), and a big increase in the final output error means (left) which is the reverse case, that leads to the same conclusions. The bottom row W_{opt} displays improvements on the two error means after optimization, which is what we seek: doing $\boldsymbol{\Pi}_{opt}$ and $\boldsymbol{\Theta}_{opt}$ is useful determin $\boldsymbol{\Omega}_W$ (see Sect. IV), in order to decrease both sensitivities in one adequate optimization. As we stressed in the previous sections, the trajectories that are fed to the system have to be minimal in both sensitivities, for accuracy and security. In that way, the reduced errors after

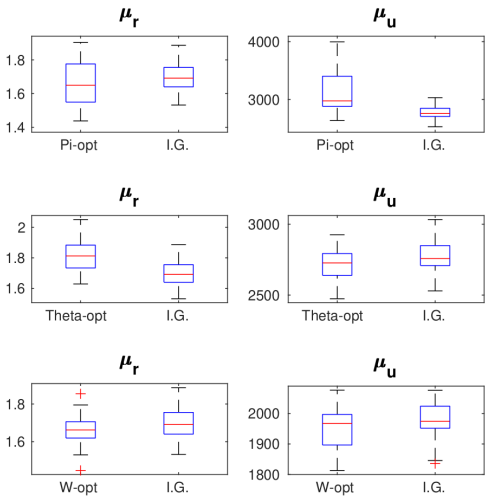

Fig. 2. Boxplots of the evaluated performances for the conducted statistical campaign when comparing all the initial guesses $\boldsymbol{a}_0$ to their associated optimized trajectories of problem (19), $\boldsymbol{a}_{\boldsymbol{\Pi}_{opt}}$, $\boldsymbol{a}_{\boldsymbol{\Theta}_{opt}}$ and $\boldsymbol{a}_{W_{opt}}$, when the DFL controller has no integral term (case NI). On the left, repartition of the final output error means for $\boldsymbol{\Pi}_{opt}$ (top), $\boldsymbol{\Theta}_{opt}$ (middle) and W_{opt} (bottom). On the right, repartition of the input error means for $\boldsymbol{\Pi}_{opt}$ (top), $\boldsymbol{\Theta}_{opt}$ (middle) and W_{opt} (bottom).

W_{opt} demonstrate the effectiveness of the method, at least in the conditions we described.

Let us now analyze the results for one specific trajectory, exctracted from the second campaign of simulations for the DFL with an integral term (case I). In Fig. 3 we can observe the initial guess and its associated optimized trajectories in the 2D plane. For the *nominal* case, the quadrotor exactly follows those trajectories with no errors. However, we want to analyze the performance of each trajectory in the *perturbed* case. For this specific motion, after running the N_{sim} perturbed simulations for the four trajectories, we collect all final outputs which gives a cloud of N_{sim} 2D points for $\boldsymbol{a}_0$, $\boldsymbol{a}_{\boldsymbol{\Pi}_{opt}}$, $\boldsymbol{a}_{\boldsymbol{\Theta}_{opt}}$ and $\boldsymbol{a}_{W_{opt}}$. Then we chose to plot, for each point cloud, the associated 90% confidence ellipse (see, Fig. 3), giving us a good idea on the quadrotor precision when trying to follow a specific trajectory. We can see that $\boldsymbol{\Pi}_{opt}$ and W_{opt} are showing both very good results in term of final output errors mean and dispersion. $\boldsymbol{\Theta}_{opt}$ gives a smaller ellipse than the initial guess. Note that, interestingly, contrary to the previous statistical analysis where $\boldsymbol{\Pi}$ was shown to increase when minimizing $\boldsymbol{\Theta}$, it is not the case for these specific conditions. From this we draw the conclusion that the optimized trajectories give lower state errors, with the lowest ones occuring for $\boldsymbol{\Pi}_{opt}$ and W_{opt}.

Eventually, Fig. 4 shows the resulting of the initial guess and after W_{opt} boxplots of a full campaign for the DFL with the integral term (case I). We chose to only display the results for W_{opt} since these are the ones which are the most accomplished in terms of performance improvement. Anyway, the results for the two other optimizations for this case have the same trend. On the left we can observe the error means for the output (top) and the inputs (bottom), with significant improvements in both. On the right we can also observe the standard deviation means for the output (top) and the inputs (bottom). We can easily verify that the errors

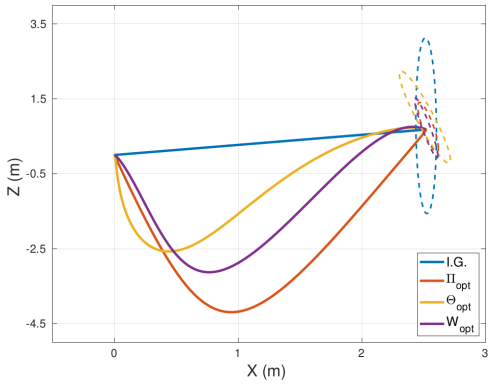

Fig. 3. Display of $\boldsymbol{\mathcal{P}}_{\boldsymbol{a}_0}$, $\boldsymbol{\mathcal{P}}_{\boldsymbol{a}_{\boldsymbol{\Pi}_{opt}}}$, $\boldsymbol{\mathcal{P}}_{\boldsymbol{a}_{\boldsymbol{\Theta}_{opt}}}$ and $\boldsymbol{\mathcal{P}}_{\boldsymbol{a}_{W_{opt}}}$ for problem (19). For each trajectory, we show the 90% confidence ellipse associated to the cloud point of the final outputs $\boldsymbol{r}_i(T)$, $i \in [\![1, N_{sim}]\!]$, with $\delta_{\boldsymbol{p}} = 10\%$.

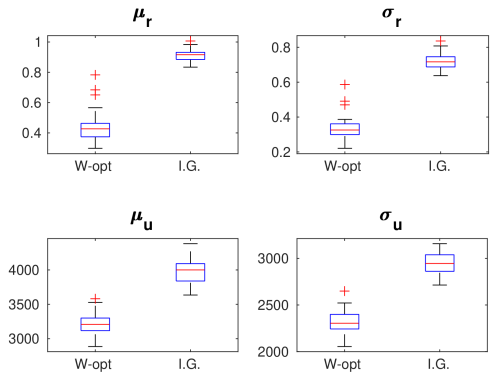

Fig. 4. Boxplots of the evaluated performances for the conducted statistical campaign when comparing all the initial guesses $\boldsymbol{a}_0$ to their associated trajectories resulting from the weighted optimization of problem (19), $\boldsymbol{a}_{W_{opt}}$, when the DFL controller has integral term (case I). At the top, repartition of the final output error means (left) and standard deviations (right); at the bottom, repartition of the input errors means (left) and standard deviations (right).

after the simulations were significantly reduced: in average, the state error means and standard deviations were decreased by a factor two, which is very good, and we also see clear improvements concerning the inputs. These results strongly demonstrate the effectiveness of the method.

VII. CONCLUSIONS

In this work we have derived a method that allows performance improvement by robust trajectory generation via constrained linear and nonlinear optimizations. From our point of view, the results shown in the previous section highlight that the sensitivity reduction method proposed in this work can bring significant accuracy and security enhancement to systems, by reducing both state and input sensivities to parametric perturbations. In future works, we plan to consider more complex optimization problems such as energy or time minimization. We also aim for an experimental validation of the approach along with quantitative comparisons with other similar studies, in order to properly evaluate their benefits and drawbacks.

REFERENCES

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

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

[3] B. Brogliato, R. Lozano, B. Maschke, and O. Egeland, "Dissipative systems analysis and control," *Theory and Applications*, vol. 2, 2007.

[4] J. Van Den Berg, P. Abbeel, and K. Goldberg, "Lqg-mp: Optimized path planning for robots with motion uncertainty and imperfect state information," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 895–913, 2011.

[5] S. Ponda, R. Kolacinski, and E. Frazzoli, "Trajectory optimization for target localization using small unmanned aerial vehicles," in *AIAA guidance, navigation, and control conference*, 2009, p. 6015.

[6] A. Censi, L. Marchionni, and G. Oriolo, "Simultaneous maximum-likelihood calibration of odometry and sensor parameters," in *2008 IEEE International Conference on Robotics and Automation*. IEEE, 2008, pp. 2098–2103.

[7] B. T. Hinson, M. K. Binder, and K. A. Morgansen, "Path planning to optimize observability in a planar uniform flow field," in *2013 American Control Conference*. IEEE, 2013, pp. 1392–1399.

[8] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory optimization for well-conditioned parameter estimation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 28–36, 2014.

[9] A. Franchi, A. Petitti, and A. Rizzo, "Decentralized parameter estimation and observation for cooperative mobile manipulation of an unknown load using noisy measurements," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5517–5522.

[10] L. M. Miller, Y. Silverman, M. A. MacIver, and T. D. Murphey, "Ergodic exploration of distributed information," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 36–52, 2015.

[11] K. Hausman, J. Preiss, G. S. Sukhatme, and S. Weiss, "Observability-aware trajectory optimization for self-calibration with application to uavs," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1770–1777, 2017.

[12] P. Salaris, R. Spica, P. R. Giordano, and P. Rives, "Online optimal active sensing control," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 672–678.

[13] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[14] 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*. IEEE, 2010, pp. 1408–1413.

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

[16] A. Ansari and T. Murphey, "Minimum sensitivity control for planning with parametric and hybrid uncertainty," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 823–839, 2016.

[17] 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.

[18] A. Chamseddine, Y. Zhang, C. A. Rabbath, C. Join, and D. Theilliol, "Flatness-based trajectory planning/replanning for a quadrotor unmanned aerial vehicle," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 4, pp. 2832–2848, 2012.

[19] S. Li, Y. Wang, and J. Tan, "Adaptive and robust control of quadrotor aircrafts with input saturation," *Nonlinear Dynamics*, vol. 89, no. 1, pp. 255–265, 2017.

[20] M. Cutler and J. How, "Actuator constrained trajectory generation and control for variable-pitch quadrotors," in *AIAA Guidance, Navigation, and Control Conference*, 2012, p. 4777.

[21] X. Wang, Ramirez-Jaime and Puig, "Nonlinear model predictive control with constraints satisfaction for a quadcopter," in *IOP Conference series: Journal of Physics*, 2017.

[22] Iam Choon Khoo, Min-Yi Shih, M. V. Wood, B. D. Guenther, Pao Hsu Chen, F. Simoni, S. S. Slussarenko, O. Francescangeli, and L. Lucchetti, "Dye-doped photorefractive liquid crystals for dynamic and storage holographic grating formation and spatial light modulation," *Proceedings of the IEEE*, vol. 87, no. 11, pp. 1897–1911, 1999.

[23] J. Baillieul and T. Samad, *Encyclopedia of systems and control*. Springer Publishing Company, Incorporated, 2015.

[24] 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, 2011.

[25] I. Y. Kim and O. L. De Weck, "Adaptive weighted-sum method for bi-objective optimization: Pareto front generation," *Structural and multidisciplinary optimization*, vol. 29, no. 2, pp. 149–158, 2005.

[26] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics and Automation magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[27] V. Mistler, A. Benallegue, and N. M'sirdi, "Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback," in *Proceedings 10th IEEE International Workshop on Robot and Human Interactive Communication. ROMAN 2001 (Cat. No. 01TH8591)*. IEEE, 2001, pp. 586–593.