Perception-constrained and Motor-level Nonlinear MPC for both Underactuated and Tilted-propeller UAVS

Martin Jacquet, Gianluca Corsini, Davide Bicego, Antonio Franchi

Perception-constrained and Motor-level Nonlinear MPC for both Underactuated and Tilted-propeller UAVs

Martin Jacquet[1], Gianluca Corsini[1], Davide Bicego[2,1], Antonio Franchi[2,1]

***Abstract*— In this paper, we present a Perception-constrained Nonlinear Model Predictive Control (NMPC) framework for the real-time control of multi-rotor aerial vehicles. Our formulation considers both constraints from a perceptive sensor and realistic actuator limitations that are the rotor minimum and maximum speeds and accelerations. The formulation is meant to be generic and considers a large range of multi-rotor platforms (such as underactuated quadrotors or tilted-propellers hexarotors) since it does not rely on differential flatness for the dynamical equations, and a broad range of sensors, such as cameras, lidars, etc... The perceptive constraints are expressed to maintain visibility of a feature point in the sensor's field of view, while performing a reference maneuver. We demonstrate both in simulation and real experiments that our framework is able to exploit the full capabilities of the multi-rotor, to achieve the motion under the aforementioned constraints, and control in real-time the platform at a motor-torque level, avoiding the use of an intermediate unconstrained trajectory tracker.**

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are becoming more and more popular thanks to the increasingly powerful hardware and software, and the strong aerial robotics community that provides new efficient control techniques. Moreover, the increasing availability, compactness and lightweightness of onboard sensors, have resulted in a growing interest in using such aerial platforms in a large range of applications, from search and rescue tasks to aerial monitoring or exploration, as well as work in high risks places or human-denied areas. These tasks can be either perception-only or require physical interaction. Besides, perception sensors are also of interest in regards to the localization in the surrounding environment [1] which is another criticality, especially when the robot is moving around humans. In all these tasks, perception has a fundamental role, as the lost of perceivability of the area of interest may lead to the inability to fulfill the mission, or to an immediate danger for the people around.

Considering perception in the planning and control of an aerial vehicle is somehow challenging, due to the computational load of the processing, and the difficulty to consider at the same time the perception constraints and the dynamic limitations of the platform. As these constraints are critical both for the sake of achieving the task and for safety reasons, it is, therefore, very important to design a control architecture able to ensure that the desired task is performed while satisfying the aforementioned constraints.

Recently, MPC has gained popularity in aerial robotics, and some work started to incorporate perception-based constraints [2]–[5] directly into the multi-rotor control architecture, since these constraints can be expressed and implemented as a general inequality, like any other constraint acting on the system. In particular, MPC is a model-based and optimization-based control technique using the dynamic model of the system to predict its behavior over a finite receding horizon.

Thanks to the prediction capability of this control scheme, it is meant to be in-between planning and control, providing a short term sequence of inputs to accomplish a given task. It allows a great reactivity of the platform since the short term controls are planned online, and this predictive aspect is also a way to be more compliant with the internal and the external constraints. However, the high nonlinearity of these constraints are limiting the use of these MPC frameworks as real-time controllers for the complete system. They are thus used as local trajectory planners or as attitude references, while the low-level or attitude control is left to unconstrained trackers or attitude regulators [6]–[9]. This leads to some important issues: *i)* there is no guarantee that the local trajectory tracker will fulfill the perceptive constraints which might lead to a loose of perceivability, and either *ii)* the MPC planner does not take into consideration the real constraints acting on the system inputs, such as motor bounds on speed and acceleration, or *iii)* it takes them into account, but the use of a low level tracker jeopardizes such feature.

In this work, we build upon the work presented in [10] and propose a Perception-based Nonlinear MPC approach that considers the real constraints on the actuators and feeds directly the low-level inputs to the rotor-speed controllers. As the NMPC solver acts as a controller, the proposed solution has also to cope with strong real-time constraints. In addition to this, the proposed formulation is extended beyond standard quadrotors since it does not rely on differential flatness for the modeling of the system's dynamics, and covers a larger spectrum of multi-rotor vehicles with, e.g., differently-oriented propellers, for instance under-actuated or fully-actuated platforms [11].

The paper is organized as follows. In Sec. II, we detail

[1]LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France, martin.jacquet@laas.fr, gianluca.corsini@laas.fr, davide.bicego@laas.fr, antonio.franchi@laas.fr

[2]Robotics and Mechatronics lab, Faculty of Electrical Engineering, Mathematics & Computer Science, University of Twente, Enschede, The Netherlands d.bicego@utwente.nl, a.franchi@utwente.nl

This work was partially funded by the ANR, under the Projects ANR-17-CE33-0007 'MuRoPhen' and ANR-18-CE33-0001 'The Flying Coworker', the cooperation program INTERREG Deutschland-Nederland as part of the SPECTORS project 143081, and by the European Union's Horizon 2020 research and innovation programme under grant agreement ID: 871479 AERIAL-CORE.

the modeling of the dynamics and the formulation of the perceptive constraints, while Sec. III contains the methodological aspects and the formalization of the optimal control problem. Finally, in Sec. IV we present the experimental results achieved using our framework.

II. MULTI-ROTOR MODELING

In this section we introduce the equality constraints due to the multi-rotor dynamics and the inequality constrains due to the motor-torque limitations and the perception requirements.

A. Equality Constraints due to the Multi-rotor Dynamics

The multi-rotor is modeled as a rigid body of mass m, with $n \geq 4$ actuators (motors plus propellers). The propellers rotation axes can be either all pointing in the same direction (collinear configurations) like in standard underactuated platforms, or pointing in different directions (tilted configurations) to have a larger actuation span (see [12] and references therein). The world inertial frame and the rigid-body frame (whose origin coincides with the center of mass of the multi-rotor) are denoted by $\mathscr{F}_W$, $\mathscr{F}_B$, respectively. The position of O_B – the origin of $\mathscr{F}_B$ in $\mathscr{F}_W$ – is denoted by ${}^W\mathbf{p}_B$, while ${}^W\mathbf{R}_B$ is the rotation matrix that represents the orientation of $\mathscr{F}_B$ with respect to (w.r.t.) $\mathscr{F}_W$, and similarly for all the other frame pairs. The robot state is defined as the concatenation of the body state $\mathbf{x}_b$ and the state of the actuators $\mathbf{x}_a$, i.e.,

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_b^\top & \mathbf{x}_a^\top\end{bmatrix}^\top \in \mathbb{R}^{12+n}. \tag{1}$$

The body state $\mathbf{x}_b$ is defined as

$$\mathbf{x}_b = [\mathbf{p}^\top \mathbf{v}^\top \boldsymbol{\eta}^\top \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^{12}, \tag{2}$$

where $\mathbf{p} = {}^W\mathbf{p}_B$, $\mathbf{v}$ is the velocity of O_B expressed in $\mathscr{F}_W$, $\boldsymbol{\eta} = [\phi\ \theta\ \psi]^\top$ is any Euler angle representation of ${}^W\mathbf{R}_B$, and $\boldsymbol{\omega}$ is the angular velocity of $\mathscr{F}_B$ w.r.t. $\mathscr{F}_W$, expressed in $\mathscr{F}_B$.

The actuator state is composed of the rotational speeds of all the propellers, denoted $\mathbf{w} = [w_1 \dots w_n]^\top$. In fact, such speeds cannot be changed instantaneously – thus can not be considered as an input – but undergo the dynamics of a rotating body with a mechanical inertia subject to external moments such as friction, aerodynamic drag, and the motor torque, the latter being the typical controllable input of a motor. According to the most common propulsion model, the rotational speed can be algebraically related to the force produced by the propeller (see, e.g., [13]) and therefore such force can equivalently represent the actuator state through a suitable change of coordinates. Therefore we set

$$\mathbf{x}_a = \boldsymbol{\gamma} \in \mathbb{R}^n, \tag{3}$$

where $\boldsymbol{\gamma} = [f_1 \dots f_n]^\top$ is the n-dimensional vector of the forces produced by the n propellers. Similarly, one can make a (feedback, or state-dependent) change of coordinates from the motor-torque input to the derivative of the force produced by the propeller. As a consequence, we can consider

$$\dot{\boldsymbol{\gamma}} = \mathbf{u}, \tag{4}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the n-dimensional vector of controllable inputs to be chosen by the controller. We remark that considering (4) is equivalent to require the motion controller to control directly the motor torque of each propeller, i.e., the lowest possible control input of the multi-rotor system. This choice is physically the most meaningful one and avoids to assume intermediate low-level controls, such as attitude or angular velocity control loops, as done often in the literature.

The kinematic equations of the multi-rotor body are

$$\dot{\mathbf{p}} = \mathbf{v} \tag{5a}$$

$$\dot{\boldsymbol{\eta}} = \mathbf{T}(\boldsymbol{\eta})\boldsymbol{\omega} \tag{5b}$$

where $\mathbf{T}(\boldsymbol{\eta})$ is the Jacobian matrix mapping $\boldsymbol{\omega}$ to $\dot{\boldsymbol{\eta}}$. The linear and angular accelerations are defined by

$$\begin{bmatrix}\dot{\mathbf{v}} \\ \dot{\boldsymbol{\omega}}\end{bmatrix} = \begin{bmatrix}m\mathbf{I}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{J}\end{bmatrix}^{-1} \left(\begin{bmatrix}-mg\mathbf{z}_W \\ -\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}\end{bmatrix} + \begin{bmatrix}\mathbf{R}(\boldsymbol{\eta}) & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{I}_3\end{bmatrix} \mathbf{G}\boldsymbol{\gamma} \right) \tag{6}$$

where $\mathbf{J} \in \mathbb{R}^{3\times3}$ is the positive definite body inertia matrix, $\mathbf{O}_3$ and $\mathbf{I}_3 \in \mathbb{R}^{3\times3}$ are respectively the zero and identity matrixes, g and $-\mathbf{z}_W$ are the intensity and the unit vector direction of the gravity force in $\mathscr{F}_W$. Finally, $\mathbf{G} \in \mathbb{R}^{6\times n}$ is the *force/moment allocation matrix* mapping the forces produced by each propeller to the total force and moment acting on the body. Notice that $\mathbf{G}$ includes the drag moments, which can be written as a linear function of the propeller forces. For a thorough derivation and explanation of all the terms in (6) the reader is referred to previous works, e.g., to [13]. Note that $\text{rank}(\mathbf{G}) = 4$ for standard under-actuated platforms while for tilted-propeller multi-rotors systems one can obtain $\text{rank}(\mathbf{G}) = 5$ or $\text{rank}(\mathbf{G}) = 6$, i.e., a larger actuation span. The proposed model is general enough to be valid in all these situations.

Grouping together (5), (6) and (4), the dynamics of the multirotor is expressed as a set of nonlinear differential equations, synthetically denoted in the following as

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{7}$$

which represent the equality constraints for the optimal control problem that will be defined next.

B. Inequality Constraints Induced by the Motor Torques

Because of the bounded torque of the motors – induced by the electrical current they can tolerate – we need also to add inequality constraints on the state and input. First of all the forces $\boldsymbol{\gamma}$ must satisfy the following constraint

$$\underline{\boldsymbol{\gamma}} \leq \boldsymbol{\gamma} \leq \overline{\boldsymbol{\gamma}} \tag{8}$$

were $\underline{\boldsymbol{\gamma}}$ is directly related to the minimum rotational speed of the motors and $\overline{\boldsymbol{\gamma}}$ is related to the rotational speed achieved at steady state when applying the maximum torque to the motor. Such limit speed is of course bounded due to the always present dissipative effects (friction, propeller air drag, etc). Then, the input $\mathbf{u}$ must satisfy the following constraint

$$\underline{\dot{\boldsymbol{\gamma}}}(\boldsymbol{\gamma}) \leq \mathbf{u} \leq \overline{\dot{\boldsymbol{\gamma}}}(\boldsymbol{\gamma}) \tag{9}$$

where $\underline{\dot{\boldsymbol{\gamma}}}$ and $\overline{\dot{\boldsymbol{\gamma}}}$ represent the minimum and maximum accelerations that can be obtained when applying the minimum and maximum torque to the motor, respectively. Such limits

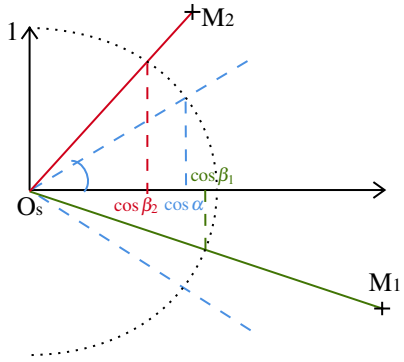

Fig. 1: The perceptive constraint is expressed using the cosine of the normalized bearing vector. In this example M_1 is in the FoV while M_2 is not.

depend also on the inertia of the propeller, the friction and the air drag, which in turn depend on the propeller speed. This justifies the dependency on γ of such limits.

However, it is not required to know all the parameters of the motors and propellers to compute the limits in (8) and (9). These quantities can be determined with an identification campaign, as shown in [10].

Notice that no limit is induced on $\mathbf{x}_b$ as far as the actuation capabilities of the multirotor are concerned. This differentiates our model from all the works in which fictitious constraints on $\boldsymbol{\eta}$ or $\boldsymbol{\omega}$ have been introduced in order to simplify the optimal control problem (see the Introduction).

C. Inequality Constraints Induced by Perception

It is assumed to have a sensor S rigidly mounted on the UAV such that the transformation between $\mathscr{F}_S$ – the sensor frame – and $\mathscr{F}_B$ is constant and known. Therefore, the perception constraints can be expressed as inequalities [4] which are functions of $\mathbf{x}_b$ and some sensor parameters:

$$c_i(\mathbf{x}_b) \in [\underline{b}_i, \overline{b}_i], \quad i = 1, \dots, P \tag{10}$$

where c_i is the scalar function for the i-th perceptive constraint, and $\underline{b}_i$, $\overline{b}_i$ are its respective lower and upper bounds.

In particular we focus our attention here on a common application of perceptive MPC, i.e., the feature-covering along a motion [2]–[4], [14]. The sensor S is assumed to be able to perceive a feature in the environment only if it falls inside a conic Field of View (FoV) with principal axis $\mathbf{z}_S$ and halfwidth equal to α. The goal is to keep perceivability of a group of features while performing other motion tasks. In the next section the constraints (10) will be specialized for this case.

III. METHODOLOGY

A. Constraints and Cost Terms from Perception

Assume to have P features in the environment, denoted with $M_1, \dots, M_P$. Define the angle β_i between the axis $\mathbf{z}_S$ and the vector $\overrightarrow{O_S M_i}$, where O_S is the origin of $\mathscr{F}_S$ (see

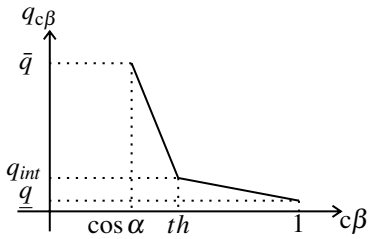

Fig. 2: Piecewise-linear function used to adapt the weight in (12).

Fig. 1). Maintaining all the features in the FoV for a certain time horizon is formulated, like in [4], as follows

$$\mathrm{c}\beta_i(t) \in [\cos\alpha, 1] \quad t \in [t_0, t_0 + T], \quad i = 1, \dots, P, \tag{11}$$

where $\mathrm{c}\beta_i$ is used as shorthand for $\cos\beta_i$. The quantity $\mathrm{c}\beta_i(t)$ depends on both $\mathbf{x}_b$ and the motion of the feature M_i. The first is included in the dynamic model, while the latter must be estimated along the motion using a predictor, such as a Kalman predictor, that fuses the sensor measurements, the knowledge of $\mathbf{x}_b$ over time, and possibly other parameters known a priori [14].

In addition to the constraint (11), having the feature close to the center of the FoV allows a larger span of possible motion than when it gets close to the border, where the active constraint limits the motion. Hence, the following term is also included in the cost function of the controller:

$$q_{\mathrm{c}\beta_i}(\mathrm{c}\beta_i - 1)^2, \tag{12}$$

where $q_{\mathrm{c}\beta_i}$ is a positive weight. Such term (12) is zero only when M_i is aligned with $\mathbf{z}_S$. The weight $q_{\mathrm{c}\beta_i}$ is adapted w.r.t. $\mathrm{c}\beta_i$ in a piecewise-linear fashion, as shown in Fig. 2. The goal of such adaptation is to: *i)* prioritize importance of M_i when it gets closer to the FoV boundary, and *ii)* avoid to perturbate the precision of the other tasks when M_i is closer to the FoV center.

B. Real-time Requirements

One of the strong requirements of our framework is that the solver has to compute in real time the low level inputs (motor-torque level) and to send them directly to the actuators. This is different from what is done in other related works that use the MPC to compute local trajectories or high level commands that are executed by a low level tracker [4] or an attitude controller [2]. In our case, a delay in the computation causes a delay in the propeller inputs, which can lead to unpredictable or unstable behaviors. To obtain the smallest delay possible, we employ a Real-Time Iteration solving strategy, where instead of having several linearization steps per iteration that refine the solution, only one step is performed. By doing so the output can become suboptimal but it still fulfills all the constraints and allows real-time computing.

This requirement also implies that the linearization step has to be as efficient as possible. Considering that the perceptive constraints are highly nonlinear, with the goal of enhancing the performances of the solver, we decided

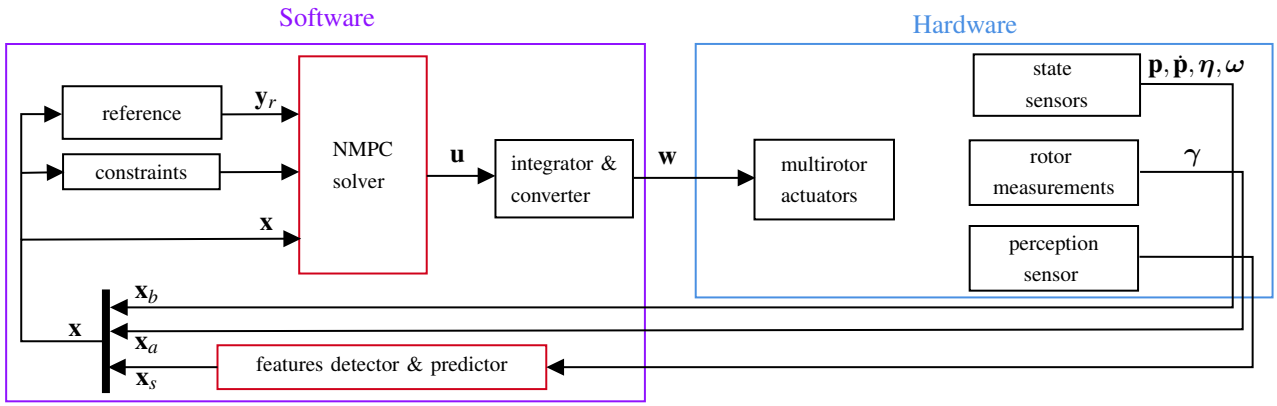

Fig. 3: Block diagram of the Perceptive NMPC framework, with main blocks highlighted in red. In simulation, the hardware part is replaced by a dynamic model of the multirotor and virtual sensors.

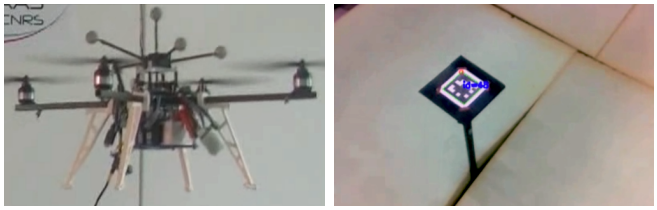

Fig. 4: On the left, the flying quadrotor equipped with a camera. On the right, the view from the sensor with the detected feature.

to fictitiously extend the state with the redundant quantities $\mathrm{c}\beta_i$, and to add them into the dynamics constraints. By doing so, the solver will locally linearize the dynamics of such variables and converge faster to a solution which fulfills all the desired constraints. This constitutes a trade-off between optimality and computational time. This results in enlarging the state of the system, including the sensors-state vector $\mathbf{x}_s = [\mathrm{c}\beta_1 \ \cdots \ \mathrm{c}\beta_P]^\top$, similarly to [2], thus obtaining

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_b^\top & \mathbf{x}_a^\top & \mathbf{x}_s^\top \end{bmatrix}^\top. \tag{13}$$

C. Optimization Problem Formulation

The discrete-time optimization problem over the receding horizon T is sampled in N shooting points, and is expressed at a given time instant t as

$$\min_{\substack{\mathbf{x}_0 \dots \mathbf{x}_N \\ \mathbf{u}_0 \dots \mathbf{u}_{N-1}}} \sum_{k=0}^{N} \|\mathbf{y}_k - \mathbf{y}_{r,k}\|^2_{\mathbf{Q}} \tag{14a}$$

$$s.t. \quad \mathbf{x}_0 = \mathbf{x}(t) \tag{14b}$$

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, 1, \dots, N-1 \tag{14c}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k), \quad k = 0, 1, \dots, N \tag{14d}$$

$$\underline{\gamma} \leq \gamma \leq \overline{\gamma}, \quad k = 0, 1, \dots, N \tag{14e}$$

$$\underline{\dot{\gamma}} \leq \mathbf{u}_k \leq \overline{\dot{\gamma}}, \quad k = 0, 1, \dots, N-1 \tag{14f}$$

$$\cos\alpha \leq \mathrm{c}\beta_{i,k}, \quad k = 0, 1, \dots, N, \ i = 1, \dots, P \tag{14g}$$

with $\mathbf{x} = [\mathbf{p}^\top \mathbf{v}^\top \boldsymbol{\eta}^\top \boldsymbol{\omega}^\top \gamma^\top \mathrm{c}\beta_1 \dots \mathrm{c}\beta_P]^\top$ being the state vector, $\mathbf{u} = \dot{\gamma}$ the input vector, and $\mathbf{f}$ the dynamic function of the system including $\mathbf{x}_s$. The matrix $\mathbf{Q}$ is a diagonal weight matrix. The output of the system $\mathbf{y}$ is expressed as a function $\mathbf{h}$ of the state and input. In particular, we use

$$\mathbf{y} = [\mathbf{p}^\top \dot{\mathbf{p}}^\top \boldsymbol{\eta}^\top \boldsymbol{\omega}^\top \ddot{\mathbf{p}}^\top \dot{\boldsymbol{\omega}}^\top \mathrm{c}\beta_1 \dots \mathrm{c}\beta_P]^\top. \tag{15}$$

The reference vector $\mathbf{y}_r$ is time dependent and varies during the time horizon. The entries of $\mathbf{y}_r$ corresponding to $\mathrm{c}\beta_i$, with $i = 1, \dots, P$ are all set to 1 and their weights are adapted as explained in Sec. III-A. The other entries of $\mathbf{y}_r$ are provided by an external reference generator, e.g., a waypoint planner and their corresponding weights are set to an higher or lower value depending on how important is the tracking task for the corresponding state variable.

IV. RESULTS

In this section, we present the experimental framework used to implement the aforementioned optimal control problem. We propose to explore simulations that present the capabilities of the framework, as well as a real flight experiment that show its applicability to real-world scenarios.

A. Experimental setup

The framework is implemented using MATMPC from [15], which is a MATLAB-based toolbox for nonlinear MPC. The solving routines of MATMPC are written in C to allow efficiency, while providing the MATLAB and Simulink comfort for prototyping a control framework. MATMPC is used with a fixed step Runge-Kutta integrator and the external solver qpOASES [16].

The presented results are obtained running the algorithm off-board on a laptop, in order to use MATLAB and Simulink, with an Intel Core i7 8850H processor and 32GB 2666MHz DDR4 on Ubuntu 16.04. A C/C++ implementation of MATMPC exists and will be used in our future works onboard the UAV. A recent onboard computer like an Intel NUC shows similar or greater capabilities than the computer used in these experiments, hence allowing a fully-onboard implementation of the framework.

The architecture of the controller is presented in Fig. 3. USB cables are used to send the inputs from the laptop to the motor controllers of the flying platform, and to get data from the perception sensor – in this case a standard monocular camera.

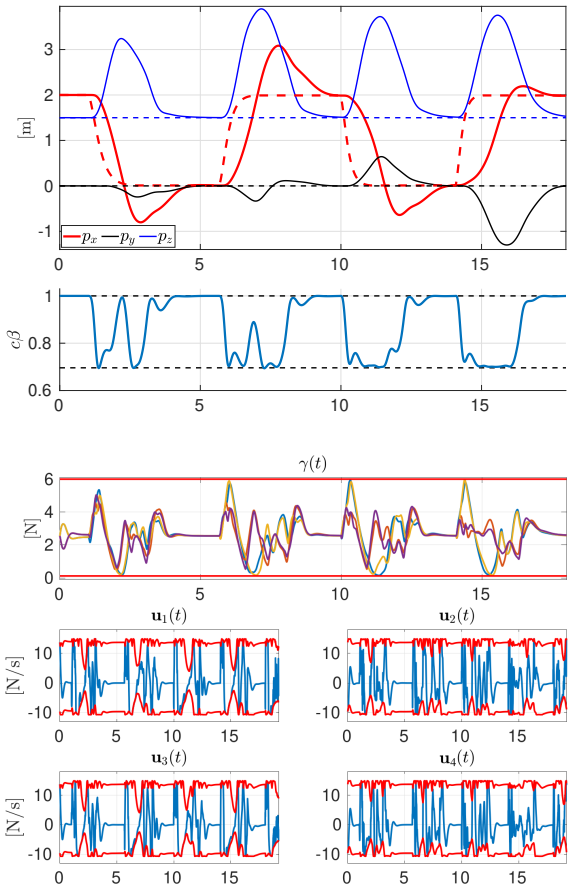

Fig. 5: The first plot shows the position tracking along the motion, with the reference values in dashed lines computed using the sensor measurements. The next two are the evolution of $c\beta$ and the vector of propeller forces $\boldsymbol{\gamma}$ along the same trajectory, with the corresponding bounds. The last four plots represent the evolution of the inputs $\mathbf{u}$ within the corresponding bounds.

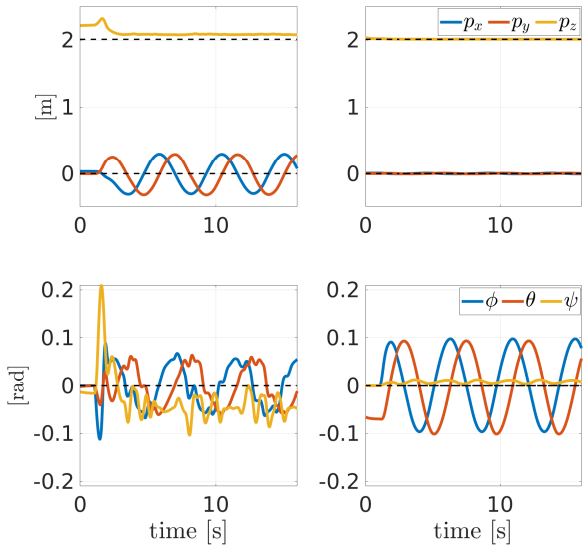

Fig. 6: On the left: the position and attitude tracking of the quadrotor, on the right: tracking for the tilted-propeller hexarotor. The controller exploits the additional actuation of the latter to stay closer to the requested hovering state (dashed black lines).

B. Simulations

First, the framework was tested in a simulated environment, using Simulink. The interface with the actual multirotor was replaced by a dynamic model which was used to update the state of the system. In order to simulate the perception sensor we used a simulated camera that computes the $c\beta$ angle directly from the position in the image plane of a marker, acting as a feature. This use case is meant to emulate a standard monocular camera of known calibration, tracking a marker of known shape, as e.g., an AprilTag, allowing to retrieve the full 6D pose ${}^{C}\mathbf{p}_{M}, {}^{C}\mathbf{R}_{M}$ of the marker in camera frame (see Fig. 4). To emulate a real camera, the frequency f_S of the simulated one was slowed down to 60Hz, meaning that the feature predictor was often using outdated measurements.

1) Increasing feature speed: The first simulation is meant to test the solver capability to explore the full constraint space to find a viable solution. A simulated quadrotor has to be on top of the moving marker of unknown trajectory and speed, while keeping perceivability at any instant. The marker goes back and forth at increasing speed, along two meters. This distance was chosen in order for the second point to be outside of the field of view of the sensor when hovering over the first. Since the target will eventually reach speeds with a few meters per second of magnitude (with high accelerations due to the small distance), the requested maneuver of the robot is very agile, and the induced tilt would break the perceivability constraint. As presented in Fig. 5 – which shows a part of the simulation in which the requested accelerations are already high – the motion of the target along the **x** axis is tracked, while the overall motion has to be modulated along the **y** and **z** axis. In this setup, the MPC controller is able to find a modified maneuver that fulfills the position tracking as well as the constraints. Doing so, it exploits the full range of the propeller forces and their derivatives, reaching several times the bounds for both.

2) Near hovering while observing a circular motion: The next use-case implemented is a requested hovering for the multirotor, while keeping perceivability of the moving marker. The marker has a circular motion, whose radius is chosen to be just outside the field of view while hovering. Fig. 6 presents the results of position and attitude tracking of an underactuated quadrotor and a tilted-propeller hexarotor. The latter is able to stay much closer to the hovering state by slightly modulating its attitude, while the first has to make a circular motion in order to maintain perceivability. With this simulation, we show that the controller is able to take advantage of the larger actuation of fully-actuated platforms in tasks where underactuation is in contrast with other objectives, such as perception.

The simulation results can be found as the third part of the attached multimedia file.

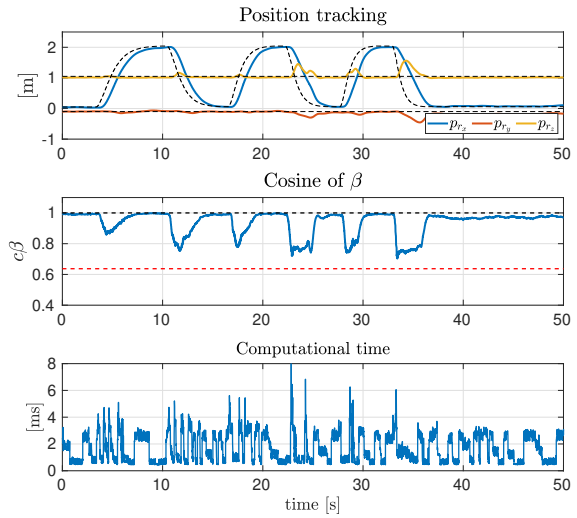

Fig. 7: Position tracking, constrained value of cβ and solving time of the NMPC problem along the real flight experiment.

C. Experimental results

We demonstrate here that our framework is able to cope with real-world constraints and control an actual multirotor. Since the framework is experimental, there is no backup strategy in case of failure of the solver, so the maneuvers are less agile than the one presented in the simulations, for safety reasons, since having a security cable would have made the motion unfeasible by blocking the propellers. Furthermore, as the accent was on the control aspects, the sensor used in this experiment was again a simulated camera.

We notice that the solving frequency, thus the control input frequency, ranges from 150 to 400 [Hz] - depending on the complexity of the optimal problem, and the number of applied constraints - which is fast enough to effectively control the multirotor (see Fig. 7).

V. CONCLUSIONS

In this work, we adapted the perceptive-aware MPC approaches of the state of the art to fit the need of more realistic system inputs and a motion that is ensured to be compliant with the perceptive constraints. The adopted formulation can be extended to a wide range of platforms and sensors. The solver is able to run in real time on a laptop, and future perspectives include transitioning to an onboard C++ implementation.

Compared to previous works, we also proposed a method able to track moving or fixed features, without any prior knowledge of their motion. The controller is then able to reach the limits of the actuators, coping with the desired motion. These properties can also be applied to visual/inertial odometry frameworks, where keeping a consistent number of features in the field of view is a requirement for being able to recover successfully the robot state.

In this work, potential collisions and occlusions are not considered, like, e.g., in [4], [17], since they do not rely on perception but rather use the *apriori* knowledge of the positions of these obstacles, and thus were not a critical point to stress according to our objectives. It will be the focus of future work with an actual sensor in the loop. In particular, feature-occlusion avoidance is of high interest in the scheme of predictive control, given the underlying computer vision processing, while still considering the real-time constraints.

REFERENCES

[1] M. Saska, T. Baca, J. Thomas, J. Chudoba, L. Preucil, T. Krajnik, J. Faigl, G. Loianno, and V. Kumar, "System for deployment of groups of unmanned micro aerial vehicles in gps-denied environments using onboard visual relative localization," *Autonomous Robots*, vol. 41, no. 4, pp. 919–944, 2017.

[2] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, "Pampc: Perception-aware model predictive control for quadrotors," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2018, pp. 1–8.

[3] B. Penin, R. Spica, P. Robuffo Giordano, and F. Chaumette, "Vision-based minimum-time trajectory generation for a quadrotor uav," in *2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Sep. 2017, pp. 6199–6206.

[4] B. Penin, P. Robuffo Giordano, and F. Chaumette, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3725–3732, 2018.

[5] M. Castillo-Lopez, S. A. Sajadi-Alamdari, J. L. Sanchez-Lopez, M. A. Olivares-Mendez, and H. Voos, "Model predictive control for aerial collision avoidance in dynamic environments," in *2018 Mediterranean Conf. on Control and Automation*, Jun. 2018, pp. 1–6.

[6] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3463–3469, 2017.

[7] M. Bangura and R. Mahony, "Real-time model predictive control for quadrotors," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 11 773–11 780, 2014.

[8] G. Darivianakis, K. Alexis, M. Burri, and R. Siegwart, "Hybrid predictive control for aerial robotic physical interaction towards inspection operations," in *2014 IEEE Int. Conf. on Robotics and Automation*, May 2014, pp. 53–58.

[9] K. Alexis, C. Papachristos, R. Siegwart, and A. Tzes, "Robust model predictive flight control of unmanned rotorcrafts," *Journal of Intelligent & Robotics Systems*, vol. 81, no. 3-4, pp. 443–469, 2016.

[10] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with actuator constraints for multi-rotor aerial vehicles," *conditionally accepted to Journal of Intelligent & Robotics Systems*, 2019. [Online]. Available: https://arxiv.org/abs/1911.08183

[11] F. Morbidi, D. Bicego, M. Ryll, and A. Franchi, "Energy-efficient trajectory generation for a hexarotor with dual- tilting propellers," in *2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Oct. 2018, pp. 6226–6232.

[12] C. F. Liew, D. DeLatte, N. Takeishi, and T. Yairi, "Recent developments in aerial robotics: A survey and prototypes overview," *arXiv preprint arXiv:1711.10085*, 2017.

[13] G. Michieletto, M. Ryll, and A. Franchi, "Fundamental actuation properties of multirotors: Force-moment decoupling and fail-safe robustness," *IEEE Trans. on Robotics*, vol. 34, no. 3, pp. 702–715, 2018.

[14] J. Thomas, J. Welde, G. Loianno, K. Daniilidis, and V. Kumar, "Autonomous flight for detection, localization, and tracking of moving targets with a small quadrotor," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1762–1769, 2017.

[15] Y. Chen, M. Bruschetta, E. Picotti, and A. Beghi, "Matmpc - a matlab based toolbox for real-time nonlinear model predictive control," in *2019 European Control Conference*, Jun. 2019, pp. 3365–3370.

[16] H. Ferreau, C. Kirches, A. Potschka, H. Bock, and M. Diehl, "qpOASES: A parametric active-set algorithm for quadratic programming," *Mathematical Programming Computation*, vol. 6, no. 4, pp. 327–363, 2014.

[17] C. Lim, B. Li, E. M. Ng, X. Liu, and K. H. Low, "Three-dimensional (3D) dynamic obstacle perception in a detect-and-avoid framework for unmanned aerial vehicles," in *2019 Int. Conf. on Unmanned Aircraft Systems*, Jun. 2019, pp. 996–1004.