Motor and Perception Constrained NMPC for Torque-controlled Generic Aerial Vehicles

Martin Jacquet, Antonio Franchi

Motor and Perception Constrained NMPC for Torque-controlled Generic Aerial Vehicles

Martin Jacquet[1] and Antonio Franchi[2,1]

***Abstract*—This letter presents a perception-aware and motor-level non-linear model predictive control scheme for multi-rotor aerial vehicles. Our formulation considers both real actuation limitations of the platform, and realistic perception objectives for the visibility coverage of an environmental feature while performing a reference task. It directly produces the rotor-level (torque) inputs of the platform motors at high frequency, hence it does not require an intermediate unconstrained controller to work. It is also meant to be generic, by covering standard coplanar quadrotors as well as tilted-propeller multi-rotors. We propose an open-source fully onboard implementation of the method, capable of running at 500 Hz under the intermittent and noisy measurements of one or more cameras. The implementation is extensively tested both in simulation and in real experiments with two substantially different multi-rotor platforms, an underactuated and a fully actuated one, both equipped with two cameras, clearly demonstrating the practicability and high performance of the method.**

***Index Terms*—Aerial Systems: Mechanics and Control; Aerial Systems: Perception and Autonomy**

I. Introduction

UNMANNED Aerial Vehicles (UAVs) are increasingly used in a large range of applications, from search and rescue tasks to aerial monitoring or exploration, as well as work in high-risk places or human-denied areas. To achieve autonomy in such tasks, perception-awareness of the environment is a mandatory feature. In this regard, in a large number of applications, it is fundamental to prevent a potential loss of visibility of a feature or an area of interest, which may lead to the inability to fulfill the task, or to an immediate danger for the people around.

To comply with this imperative, the control schemes of the UAVs must consider constraints coming both from the perception and the actuation domains. Optimization-based control techniques such as Model Predictive Control (MPC) or Quadratic Programming (QP) have the property to express in a common way constraints with different semantics. In particular, MPC, a control policy that uses the dynamic model of the system to predict its behavior over a finite receding horizon, has gained popularity in aerial robotics over the recent years where robust MPC formulations for UAVs have been proposed – see, e.g., [1], [2].

In fact, some recent works push toward incorporating perception-based constraints in MPC formulations [3]–[7]. The objective in all these works is to maintain visibility on a feature while moving for a given task. While [3] proposes a minimum time trajectory generation under visibility constraints, [4], [5], [8] propose an optimal-control based low-level trajectory generation with similar constraints. More recently, [6] proposed to include the feature pixel in the system dynamic model.

However, these works are using MPC as a global or local trajectory planner, while the low-level control is delegated to *unconstrained* trackers, which do not guarantee the fulfillment of the constraints while the system is tracking the computed trajectories. Such unconstrained tracker may jeopardize the feasibility and correctness of the task execution. In addition to this crucial limitation, the MPC planners proposed in such works do not consider the real constraints of the platform induced by the actuators, but rather simplify the problem introducing fictitious constraints on the system state which do not exist on the real platform, such as constant bounds on linear and angular velocities and sum of propeller thrusts [4] and/or maximum attitude (i.e., pitch and roll) angles. The angular velocity is indeed not bounded by any physical constraint, the bound on the sum of propeller thrusts is not constant because it depends on the current total moment applied. Similarly, no actual bound is present in the attitude of a real platform.

In previous works [9], [10], we defined a non-linear perception-aware MPC framework for a Generically Tilted Multi-Rotor (GTMR) vehicle – a model that includes both the standard underactuated collinear quadrotors and the more recent fully actuated platforms with non-collinear propellers [9], [11]. This model does not rely on any assumption of flatness – as it is often done in other optimization-based approaches, e.g., [3]–[5]), which is a powerful theoretical tool but does not capture the real nature of the input space of a multi-rotor system, which corresponds, ultimately, to the torques applied by the propeller motors. Our model instead includes both realistic (i.e., torque level) actuator constraints and perception constraints. Furthermore, the solver computes the motor torque inputs – at a sufficiently high rate – which is directly fed to the motor, thus removing any unconstrained low-level tracker.

However, even if [9], [10] demonstrate experimental validations of the proposed controller, they suffered the absence of an onboard implementation. All the computation was done

[1]LAAS-CNRS, Institut National des Sciences Appliquées, Université de Toulouse, CNRS, Toulouse, France, martin.jacquet@laas.fr, antonio.franchi@laas.fr

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

This work was partially funded by the ANR, under the Projects ANR-17-CE33-0007 'MuRoPhen' and European Commission project H2020 AERIAL-CORE (EC 871479).

The source code associated with this work can be found at: https://redmine.laas.fr/projects/perceptive-torque-nmpc

Manuscript received: October 15, 2020; Accepted November 29, 2020

This paper was recommended for publication by Editor Pauline Pounds upon evaluation of the Associate Editor and Reviewers' comments.

Digital Object Identifier (DOI): 10.1109/LRA.2020.3045654

offboard using Matlab and Simulink [12], and the rotor inputs were passed to the UAV using cables, which was adding artificial and non-modeled limitations to the motion. Also, the validation in [10] relied on a simulated sensor, using geometrical projection of a given feature position, thus emulating a perfect sensor measurements. Finally, the framework was mostly tested in simulations and had few real experiments.

In this work, we propose an improved and more accomplished version of the controller previously introduced in [10], using an onboard real-time implementation. This new version integrates a real sensor instead of a simulated [5], [10], and has a more realistic sensing model, with a pyramidal field of view and uncertainty estimation of intermittent measurements. All these improvements allowed an extensive experimental validation with quantitative metrics (control frequency, deviation from reference, or feature reprojection error).

The letter is organized as follows. In Sec. II, we detail the modeling of the multi-rotor and the sensor, we then introduce, in Sec. III, the complete optimal control problem formulation, including the equality and inequality constrains and the objective function, while in Sec. IV we present the detection process and its uncertainty model. Finally, we present the experimental results in Sec. V, before concluding about the pertinence and applicability of such a framework.

II. Modeling

In this section we detail the dynamic model of the multi-rotor along with the equality and inequality constraints from the actuators and the perception sensors.

A. Generically Tilted Multi-rotorDynamics

A Generically Tilted Multi-Rotor (introduced, e.g., in [13]) is modeled as a rigid body of mass m with $n \geq 4$ actuators. The actuators can either be in collinear or tilted configurations. The tilted configuration is defined by two angles α_a and β_a, as depicted in Figure 1.

The world inertial frame and the body frame are respectively denoted by $\mathcal{F}_W$ and $\mathcal{F}_B$. The position of the origin O_B of $\mathcal{F}_B$ with respect to (w.r.t.) $\mathcal{F}_W$ is denoted by ${}^W\mathbf{p}_B$ and the unit quaternion representing the rotation from $\mathcal{F}_B$ to $\mathcal{F}_W$ is denoted with ${}^W\mathbf{q}_B$; and similarly for all the other frame pairs.

The robot state $\mathbf{x}$ is expressed as the concatenation of the body state $\mathbf{x}_b$ and actuator state $\mathbf{x}_a$, which are defined as

$$\mathbf{x}_b = [\mathbf{p}^\top \mathbf{q}^\top \mathbf{v}^\top \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^{13}, \tag{1a}$$

$$\mathbf{x}_a = \boldsymbol{\gamma} \in \mathbb{R}^n, \tag{1b}$$

where $\mathbf{p} = {}^W\mathbf{p}_B$, $\mathbf{q} = [q_w \ q_x \ q_y \ q_z]^\top$ is a shorthand for ${}^W\mathbf{q}_B$, $\mathbf{v}$ is the velocity of O_B expressed in $\mathcal{F}_W$, $\boldsymbol{\omega}$ is the angular velocity of $\mathcal{F}_B$ w.r.t. $\mathcal{F}_W$, expressed in $\mathcal{F}_B$, and $\boldsymbol{\gamma}$ is a vector containing the n forces produced by the n propellers, which are directly linked by a change of coordinates to the propeller rotational speeds [13]. The quaternion representation of the attitude is relevant in order to be resilient to orientation singularities that would arise with a minimal representation, such e.g., Euler angles.

Similar to [9], we choose to define the system input as the motor torque of each actuator, which are the real control variables of the motors, thus the most physically meaningful, and allow to avoid an intermediate low-level controller. Doing a simple change of coordinates, the motor dynamics becomes

$$\dot{\boldsymbol{\gamma}} = \mathbf{u}, \tag{2}$$

where $\mathbf{u}$ are the real inputs of the system, see [9] for the details. Equation 2 completes the dynamic equations of the multi-rotor, which are recapped hereafter:

$$\dot{\mathbf{p}} = \mathbf{v} \tag{3a}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\begin{bmatrix}0\\ \boldsymbol{\omega}\end{bmatrix} \otimes \mathbf{q} \tag{3b}$$

$$\begin{bmatrix}\dot{\mathbf{v}}\\ \dot{\boldsymbol{\omega}}\end{bmatrix} = \begin{bmatrix} m\mathbf{I}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{J}\end{bmatrix}^{-1} \left(\begin{bmatrix} -mg\mathbf{z}_W \\ -\boldsymbol{\omega}\times\mathbf{J}\boldsymbol{\omega}\end{bmatrix} + \begin{bmatrix} \mathbf{q} \otimes \mathbf{G}_f\boldsymbol{\gamma} \otimes \mathbf{q}^* \\ \mathbf{G}_\tau\boldsymbol{\gamma}\end{bmatrix}\right) \tag{3c}$$

where $\otimes$ denotes the Hamilton product of two quaternions, $\mathbf{q}^*$ is the conjugate quaternion of $\mathbf{q}$, $\mathbf{J} \in \mathbb{R}^{3\times3}$ 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 matrices, g and $-\mathbf{z}_W$ are the intensity and the unit vector direction of the gravity force in $\mathcal{F}_W$. Finally, $\mathbf{G}_f$ and $\mathbf{G}_\tau \in \mathbb{R}^{3\times n}$ are respectively the *force* and *moment allocation matrices* [13], mapping the forces produced by each propeller to the total force and moment acting on the body. This model includes and goes way beyond the standard quadrotor models.

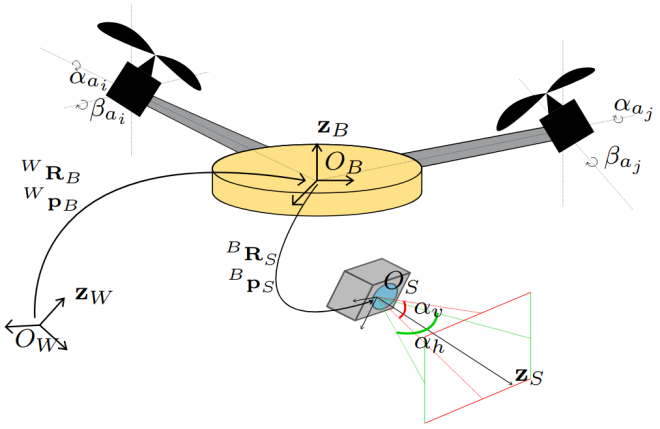

Fig. 1: Scheme of the multi-rotor, depicting the presented frames and angles, illustrated with a camera and two of the n propellers, i and j, with their respective tilting angles.

B. Generic sensor model

For the sake of generality, we abstract a generic onboard sensor as a punctual device centered in O_S, with a principal axis $\mathbf{z}_S$, able to retrieve the 3D-pose – in its own frame $\mathcal{F}_S$ – of a feature M, as long as it falls inside its field of view (FoV), defined as a pyramidal shape around the principal axis. We denote α_v and α_h the vertical and horizontal angles of the FoV. We assume the sensor rigidly attached to the multi-rotor body such that the pose transformation between $\mathcal{F}_S$ and $\mathcal{F}_B$ is constant and known.

This model can be applied to various types of sensors often equipped on mobile robots, such as some lidars, depth cameras

(stereo or infrared), or monocular cameras. For the latter, the capability of extracting the 3D-pose of the detected feature requires an extra assumption regarding the knowledge of the size of the feature, which can typically be given using a geometrical a priori information on the feature [8], structure-from-motion [14] or a machine learning-based algorithm [15]. The pose detection then induces some processing of the sensors data, but comes with a lighter hardware burden, since such cameras are often very lightweight and have large FoV compared to other sensors.

Finally, we note that only the angular measurement of the feature will be needed to express the visibility constraint and objectives, but the depth information is crucial in a MPC framework since it allows the prediction of relative motion of the feature w.r.t. the GTMR.

III. Optimal Control Problem Formulation

The problem tackled by the proposed framework is expressed as follows: a GTMR equipped with a generic sensor has to execute a given motion task while maintaining visibility on a feature. The computed control has to comply with the platform actuation constraints, as well as the visibility constraints, while achieving the task at best.

The proposed formulation is stated for one sensor and feature, yet the extension to several sensors and features is straightforward. This scalability is illustrated in Sec. V.

In the following, we detail all the constraints applied to the system and the objective function, and finally formalize the optimal control problem.

A. Actuators Constraints

The physical limitations of the system come from the limitations of the actuators themselves. The motors can receive a limited amount of current, hence have a minimum and maximum torque, which implies, due to friction, a bounded rotational speed for the motor. It also has a bounded acceleration since it undergoes the inertia of a rotating body. These limitations are equivalently recast as constraints on $\boldsymbol{\gamma}$ and $\dot{\boldsymbol{\gamma}}$:

$$\underline{\boldsymbol{\gamma}} \leq \mathbf{x}_a \leq \overline{\boldsymbol{\gamma}} \tag{4a}$$

$$\underline{\dot{\boldsymbol{\gamma}}}(\boldsymbol{\gamma}) \leq \mathbf{u} \ \leq \overline{\dot{\boldsymbol{\gamma}}}(\boldsymbol{\gamma}) \tag{4b}$$

where $\underline{\boldsymbol{\gamma}}$, $\overline{\boldsymbol{\gamma}}$, $\underline{\dot{\boldsymbol{\gamma}}}(\boldsymbol{\gamma})$ and $\overline{\dot{\boldsymbol{\gamma}}}(\boldsymbol{\gamma})$ can be obtained through an identification campaign, as shown in [9].

We note that these are the only real physical constraints applied to the system. Other potential constraints, such as limitations on the linear or angular velocities, can only artificially limit the range of capabilities of the platform, and would be contingent to a more specific task or context.

B. Perception Constraints

Keeping the visibility is of paramount importance, therefore, to prevent its loss, we introduce a corresponding hard constraint in the system. For a sensor S we can define a visibility constraint on a feature M that we want to maintain in the field of view. With a pyramidal FoV, the visibility is typically decoupled into horizontal and vertical constraints [16].

Writing the position of M in $\mathcal{F}_S$ as ${}^S\mathbf{p}_M = [x_M \ y_M \ z_M]^\top$, such constraints can be written in the virtual image plane as:

$$|x_M/z_M| \leq \tan\alpha_h, \qquad |y_M/z_M| \leq \tan\alpha_v. \tag{5}$$

C. Objective Function

The motion task is achieved by minimizing the distance to a reference motion. This task is expressed as a reference trajectory in position and attitude, denoted with $(\mathbf{p}_r, \mathbf{q}_r)$, usually provided together with its time derivatives, up to the second order, provided for all the sampling points over the receding horizon by an external trajectory planner.

The error w.r.t. the reference is defined as a weighted squared Euclidean norm of the difference denoted $\|\cdot\|^2_{\mathbf{Q}}$, where $\mathbf{Q}$ is the diagonal weight matrix, which acts as the tunable controller gains. The Euclidean distance between two unit quaternions is not suitable to represent the dissimilarity between two orientations, mainly because $\mathbf{q}$ and $-\mathbf{q}$ represent the same orientation in $SO(3)$. Following [17] we rather consider the geodesic distance in the manifold of unit quaternions, written as $d(\mathbf{q}_1, \mathbf{q}_2) = \|\log(\mathbf{q}_1 \otimes \mathbf{q}_2^*)\|$. In the following, for the sake of readability, we will keep the notation $\|\mathbf{q} - \mathbf{q}_r\|^2_{\mathbf{Q}}$ to refer to the weighted attitude error associated to this distance.

The perception task is also considered in the cost function. To achieve a robust tracking, the angular distance β between ${}^S\mathbf{p}_M$ and the sensor principal axis $\mathbf{z}_S$ is minimized; which is equivalent to maximizing its cosine, denoted $\mathrm{c}\beta$, whose evaluation is more efficient since it can be done by normalizing and projecting ${}^S\mathbf{p}_M$ on $\mathbf{z}_S$ [5]. To reduce the motion blur and improve the predictability of the feature motion, we also minimize the corresponding time derivative $\dot{\mathrm{c}\beta}$.

Finally, we write the output map $\mathbf{y}$ and its reference $\mathbf{y}_r$ as:

$$\mathbf{y} = [\mathbf{p}^\top \ \mathbf{q}^\top \ \dot{\mathbf{p}}^\top \ \boldsymbol{\omega}^\top \ \ddot{\mathbf{p}}^\top \ \dot{\boldsymbol{\omega}}^\top \ \mathrm{c}\beta \ \dot{\mathrm{c}\beta}]^\top \tag{6a}$$

$$\mathbf{y}_r = [\mathbf{p}_r^\top \ \mathbf{q}_r^\top \ \dot{\mathbf{p}}_r^\top \ \boldsymbol{\omega}_r^\top \ \ddot{\mathbf{p}}_r^\top \ \dot{\boldsymbol{\omega}}_r^\top \ 1 \ 0]^\top. \tag{6b}$$

D. Optimal Control Problem

We express the discrete-time optimization problem over the receding horizon T, sampled in N shooting points, at a given instant t, as

$$\min_{\substack{\mathbf{x}_0 \dots \mathbf{x}_N \\ \mathbf{u}_0 \dots \mathbf{u}_{N-1} \\ \mathbf{p}_{M_0} \dots \mathbf{p}_{M_N}}} \sum_{k=0}^{N} \|\mathbf{y}_k - \mathbf{y}_{r,k}\|^2_{\mathbf{Q}} \tag{7a}$$

$$s.t. \quad \mathbf{x}_0 = \mathbf{x}(t) \tag{7b}$$

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{u}_k), \qquad k \in \{0, N-1\} \tag{7c}$$

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{u}_k, \mathbf{p}_{M_k}), \qquad k \in \{0, N\} \tag{7d}$$

$$\underline{\boldsymbol{\gamma}} \leq \boldsymbol{\gamma}_k \leq \overline{\boldsymbol{\gamma}}, \qquad k \in \{0, N\} \tag{7e}$$

$$\underline{\dot{\boldsymbol{\gamma}}}_k \leq \mathbf{u}_k \leq \overline{\dot{\boldsymbol{\gamma}}}_k, \qquad k \in \{0, N-1\} \tag{7f}$$

$$|x_M/z_M|_k \leq \tan\alpha_h, \qquad k \in \{0, N\} \tag{7g}$$

$$|y_M/z_M|_k \leq \tan\alpha_v, \qquad k \in \{0, N\} \tag{7h}$$

where $\mathbf{x}$ is the state vector, $\mathbf{u}$ the input vector, $\mathbf{x}(t)$ is the measurement of the current state, $\mathbf{f}$ synthetically denotes the dynamics of the multi-rotor expressed in (2) and (3), and $\mathbf{h}$ denotes the system output map defined in (6).

IV. Feature detection under intermittent measurements

As a particular implementation of the proposed model, we choose to use calibrated monocular cameras to detect AruCo fiducial markers [18]. These sensors are among the most widely spread on UAVs, because of their accessibility and lightness. The choice of using fiducial markers also comes from the emphasis of this work on designing a generic perception-aware control framework, where the software burden of feature extraction, which is task-related, can be suitably abstracted. In this specific case, the feature M is defined as the geometrical center of the fiducial marker. In the following we present the measurement filtering process used to make the proposed framework cope with intermittent noisy measurements.

A. Temporal data association

In order to filter the intermittent measurements (since the detection runs at, e.g., 30Hz, while the control loop frequency is typically 500Hz to 1kHz), we use an *Extended Kalman Filter* (EKF). The state vector of the EKF is defined as the position in $\mathcal{F}_S$ of the feature M:

$$ {}^S\mathbf{p}_M = [{}^S x_M \ {}^S y_M \ {}^S z_M]^\top. \tag{8} $$

We express the transition model of the EKF as a first order integration of the feature motion relative to the camera. We consider a zero velocity model for the target, in $\mathcal{F}_W$. This assumption is pertinent even for a mobile target with a small velocity in $\mathcal{F}_W$, since the relative motion of the feature w.r.t. the camera between two successive measurements will be mainly induced by the camera motion (in particular, its angular speed, since the relative distance from the feature to the camera is typically a couple of meters). A similar model for faster feature motion could be proposed assuming constant speed or constant acceleration.

The time-continuous transition model is then writen as:

$$ {}^S\mathbf{p}_M(t) = \mathbf{p}_M(t-dt) + \frac{1}{dt}\mathbf{B}(t-dt)\mathbf{u}_{\text{EKF}}(t), \tag{9a} $$

where dt is the filter period, while $\mathbf{u}_{\text{EKF}}$ and $\mathbf{B}$ are given by:

$$ \mathbf{u}_{\text{EKF}} = [\dot{\mathbf{p}}^\top \ \boldsymbol{\omega}^\top]^\top \in \mathbb{R}^6, \tag{9b} $$

$$ \mathbf{B} = \begin{bmatrix} \mathbf{I}_3 & [{}^S\mathbf{p}_M]_\times \end{bmatrix} \in \mathbb{R}^{3\times 6}, \tag{9c} $$

where $[\cdot]_\times$ is the skew operator. Note that $\mathbf{u}_{\text{EKF}}$ is expressed in camera frame. Finally, the uncertainty estimation of $\mathbf{u}_{\text{EKF}}$, denoted $\boldsymbol{\Sigma}_{\mathbf{u}}$, is usually provided by the external state estimator of the UAV.

B. Uncertainty estimation

In addition to the measurements of M, we provide the EKF with an uncertainty estimation, denoted $\boldsymbol{\Sigma}_M \in \mathbb{R}^{3\times 3}$. This aims at improving the reprojection of the feature in the camera frame. Quantitative results of this improvement are discussed in Sec. V-C. Since the 6D-pose ${}^S\mathbf{T}_M$ of M in $\mathcal{F}_S$ is retrieved from the pixel measurements $\mathbf{c}_i \in \mathbb{R}^2, i \in 1\dots 4$, of the four corners of the fiducial marker, $\boldsymbol{\Sigma}_M$ should be expressed as a function of the pixel uncertainty. We assume an isotropic Gaussian pixel noise on the measurements. Hence, we define the covariance matrix $\boldsymbol{\Sigma}_{\mathbf{c}}$ of the stacked four corner measurements $\mathbf{c}_M = [\mathbf{c}_1^\top \dots \mathbf{c}_4^\top]^\top \in \mathbb{R}^8$ as:

$$ \boldsymbol{\Sigma}_{\mathbf{c}} = \sigma^2 \mathbf{I}_8 \in \mathbb{R}^{8\times 8}. \tag{10} $$

Then, $\boldsymbol{\Sigma}_M$ is obtained from $\boldsymbol{\Sigma}_{\mathbf{c}}$ using a first order covariance approximation scheme [19], by computing the Jacobian matrix of a mapping function $f\colon \ \mathbf{c}_M \mapsto {}^S\mathbf{T}_M$.

However, such a mapping function, is typically algorithmic, e.g., *PnP*, and thus is hard to differentiate. Instead, we rather consider its inverse, which is the perspective projection of the four corner:

$$ \begin{aligned} f^{-1}\colon \ \mathbb{R}^3 \times SO(3) \ &\to \mathbb{R}^8 \\ {}^S\mathbf{T}_M \ &\mapsto \mathbf{c}_M. \end{aligned} \tag{11} $$

Note that f as the mapping from the 6D-pose to the pixel vector, since the rotational part of the transformation is needed for the computation of the Jacobian, even if only the translational part is of interest to compute $\boldsymbol{\Sigma}_M$. We denote $\mathbf{J}_{f^{-1}} \in \mathbb{R}^{8\times 6}$ the Jacobian of f^{-1}, and $\mathbf{J}_M \in \mathbb{R}^{8\times 3}$ the Jacobian of the translational part, corresponding to the first three columns of $\mathbf{J}_{f^{-1}}$.

The propagation formula is then obtained using the relation:

$$ \boldsymbol{\Sigma}_{\mathbf{c}} = \mathbf{J}_M \boldsymbol{\Sigma}_M \mathbf{J}_M^\top, \tag{12} $$

which is inverted and further simplified using (10) as:

$$ \boldsymbol{\Sigma}_M = \sigma^2 (\mathbf{J}_M^\top \mathbf{J}_M)^{-1}. \tag{13} $$

To compute $\mathbf{J}_{f^{-1}}$, we use the known four corners coordinates in the marker frame $\mathcal{F}_M$: $\mathbf{x}_i = \begin{bmatrix} \pm l/2 & \pm l/2 & 0 \end{bmatrix}^\top$, l being the marker size. For each corner, we have the relation:

$$ \mathbf{c}_i = \mathit{pix}(\mathbf{h}_i) = \mathit{pix}(\mathbf{K}({}^S\mathbf{R}_M \mathbf{x}_i + {}^S\mathbf{p}_M)), \tag{14} $$

where ${}^S\mathbf{R}_M$ is the rotation from $\mathcal{F}_M$ to $\mathcal{F}_S$, $\mathbf{K}$ is the intrinsic camera matrix, $\mathbf{h}_i \in \mathbb{R}^3$ are the homogeneous coordinates obtained from the perspective projection of the camera, and $\mathit{pix}\colon \mathbb{R}^3 \to \mathbb{R}^2$ is the *pixelization* of $\mathbf{h}_i$ into the pixel coordinates, defined as:

$$ \mathit{pix}\colon \ [x \ y \ z]^\top \mapsto \begin{bmatrix} x/z & y/z \end{bmatrix}^\top. \tag{15} $$

The derivation of (14) gives, using the chain rule, the Jacobian:

$$ \mathbf{J}_{\mathbf{T}}^{\mathbf{c}_i}(\mathbf{x}_i) = \mathbf{J}_{\mathbf{h}_i}^{\mathbf{c}_i} \ \mathbf{J}_{\mathbf{T}}^{\mathbf{h}_i}(\mathbf{x}_i) \in \mathbb{R}^{2\times 6}, \tag{16a} $$

with:

$$ \mathbf{J}_{\mathbf{T}}^{\mathbf{h}_i}(\mathbf{x}_i) = \mathbf{K}\begin{bmatrix} \mathbf{I}_3 & -{}^S\mathbf{R}_M[\mathbf{x}_i]_\times \end{bmatrix} \in \mathbb{R}^{3\times 6}, \tag{16b} $$

$$ \mathbf{J}_{\mathbf{h}_i}^{\mathbf{c}_i} = \begin{bmatrix} 1/z & 0 & -x/z^2 \\ 0 & 1/z & -y/z^2 \end{bmatrix}. \tag{16c} $$

Finally, the four Jacobian matrices $\mathbf{J}_{\mathbf{T}}^{\mathbf{c}_i}$ computed for the four corners are stacked into the full Jacobian $\mathbf{J}_{f^{-1}}$.

V. Experimental Validation

In this section, we present the experimental validation of this framework. The proposed experiments aims to underline the main contributions of this framework, while demonstrating its applicability on a real platform.

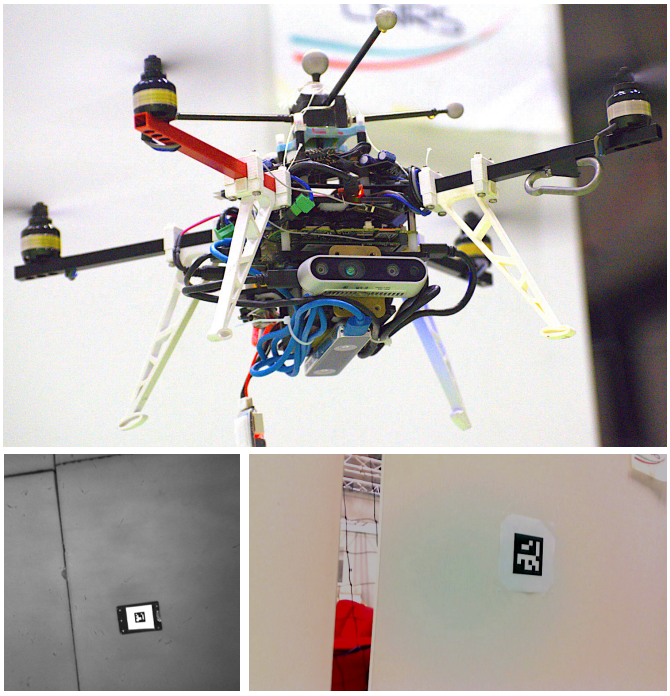

Fig. 2: The quadrotor used in the presented experiments, along with the onboard view of the two embedded cameras.

A. Experimental Setup

The framework is implemented in C++, using GenoM [20] which is a middleware-independent component generator, that can then be compiled for a given middleware, e.g., ROS. The NMPC component is based on CPPMPC, the C++ implementation of the MATMPC software [12]. The dynamic equations are translated to C code using CasADi [21], and the discretization is done using a 4^{th} order explicit Runge-Kutta integrator. The hardware interface as well as the state estimation and path planning are done using the TeleKyb3 softwares, available on the OpenRobots platform[1]. Note that the UAV state estimation is done using a combination of external motion capture and internal IMU.

The software framework can later be connected to the actual platform, or to a Gazebo simulated system that emulates the platform interface. Details on how to use this software can be found in the provided git.

The validations presented in this section are obtained using UAVs equipped with an onboard Intel NUC, with an Intel Core i7-8565U and 8GB of DDR3 RAM, on Ubuntu 18.04. The onboard sensor are the Intel RealSense T265 and D435 cameras. They were chosen for their easy onboard implementation, light weight, and because they cover two distinct types of cameras: a common $16/9$ color camera, and a grayscale fisheye camera (undistorted into a $90°$ square image to allow the feature reprojection). Both cameras run at 30Hz. Figure 2 shows the platform used for the experiments, with the two cameras: one down-looking and one front-looking.

Videos of the proposed experiments and simulations can be found in the attached multimedia file.

[1]https://git.openrobots.org/projects/telekyb3

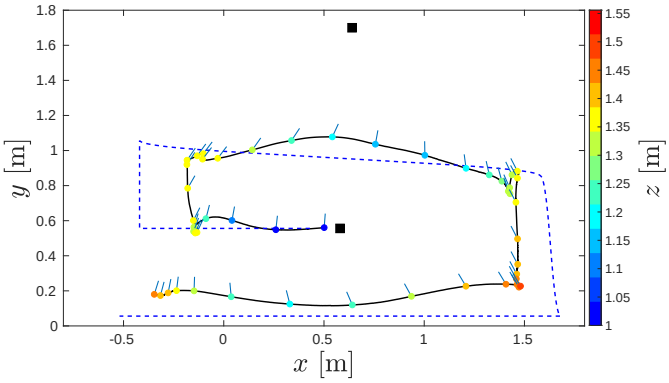

Fig. 3: (x, y) position of the UAV. The two black dots are the target position; the dashed line is the time-varying reference while the solid black line is the UAV path. The color dots represent the UAV position every 0.3 seconds, and the color represents the UAV altitude. The blue segments are the front camera principal axis orientation.

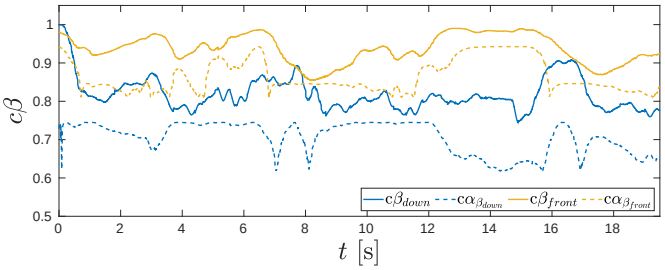

Fig. 4: The measured $\mathrm{c}\beta$ and corresponding lower bound $\cos\alpha_\beta$ for the two pairs camera/marker (front-looking in yellow and down-looking in blue).

B. Hover-to-hover Under Visibility Constraints

This experiment aims at demonstrating the capability of the proposed framework to modulate a reference task in order to maintain visibility over a set of features. In particular, the two cameras have to maintain visibility over a marker on a wall and on the ground,respectively. The two markers are fixed in $\mathcal{F}_W$, and the UAV is given a position reference trajectory that is not feasible under the visibility constraints.

Results of this experiment are presented in Figure 3, which depicts the (x, y) coordinates of the UAV, the feature positions and the reference trajectory. The color dots indicate the distance between the UAV altitude and the reference altitude, which is constant and set to $z = 1$. As the (x, y) distance between the reference and the feature increases, the deviation from the reference increases in order to accomodate for the objective and the constraints at best. Figure 3 also shows the z coordinates increasing or decreasing with depending on the UAV (x, y) position. Figure 4 shows the measured $\mathrm{c}\beta$ for the two cameras with their respective lower bounds, i.e., the cosine angular FoV $\cos\alpha_\beta$. The subscript β recalls that since we consider a pyramidal FoV, the minimum value of $\mathrm{c}\beta$ depends on its position in the image plane. In particular, α_β is minimum when the feature is on the central vertical line of the image, and maximum when it is on the image diagonal. It is a compact and visual representation of the perception constraints (5).

The deviation from the reference trajectory is dependent on the MPC weights $\mathbf{Q}$, and has to be tuned for a given task. In

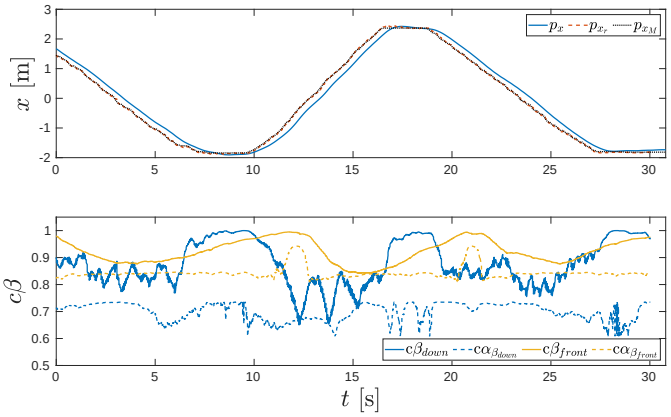

Fig. 5: Top: x coordinate of the position of the UAV (solid blue) vs the board measured marker coordinate (dashed red), and its ground truth (dotted black). Bottom: the measured $\cos\beta$ and its bound $\cos\alpha_\beta$ for the two pairs camera/marker (front- and down-looking).

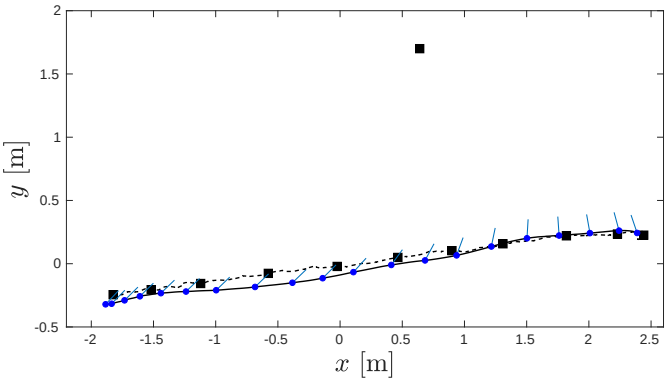

Fig. 6: (x, y) position of the UAV. The two black dots are the feature position; the dashed black line if the mobile feature path. The blue segments are the front camera principal axis orientation.

this case we imposed a better tracking on the (x, y) position than on z, which has a smaller impact in the cost function.

Finally, we want to remark that the actuation constraints (7e) and (7f) are largely satisfied during the whole experiment. We omit the plots to avoid an unnecessary information overload.

C. Mobile Feature Tracking

The setup for this experiment is the same as the one presented in V-B with the exception of the fact that the feature on the ground is mobile and the reference task given to the UAV is to stay on top of the ground detected feature at a constant altitude ($z = 1\,[\text{m}]$), while the visibility on both markers must be maintained. The rotation around the $\mathbf{z}_B$ axis is left free (this is achieved by setting the reference yaw as the last measured one, which adds just a little bit of a 'damping-like' effect). As a consequence, the controller heavily exploit such rotation to fulfill the perception constraints and objectives. This allows a smooth rotation without adding perturbations to the other tasks. It also illustrates that the controller is able to autonomously satisfy the perception constraints and objectives without the need of any additional user inputs. Results are reported in Figure 5, which shows the position tracking of the UAV along the coordinate that is mostly affected by feature motion (x axis) and the perception constraint fulfillment. As it can be seen from the plot the tracking is satisfactory and the constraints are always satisfied. The maximum speed and acceleration allowed for the feature in order for the UAV to fulfill the constraints is dependent on the sensor FoV and the requested altitude z. In the presented experiment, the average target speed is 0.5 [m/s].

Similar to Figure 3, Figure 6 depicts the (x, y) motion of the platform during one way of the maneuver. We see the yaw orientation changing during the motion in order to keep the visibility of the feature on the wall.

Table I presents the mean and standard deviation of the reprojection error between the measured feature poses and the ground truth (obtained using motion capture), with and without the uncertainty propagation method proposed in Sec. IV-B. These data are aggregated over several experiments covering three minutes of flight in each case, and in similar conditions. We note the disparity between the metrics for the two markers, which is caused by the design of the experiment. The front feature is often seen from the side, which worsen the camera measurements. For both features, the proposed method increases the reprojection precision reducing the average error by 30% to 40% and standard deviation by 15% to 60%.

Uncertainty estimation	Down feature		Front feature	
	mean	*std*	*mean*	*std*
without	0.084	0.158	0.349	0.091
with	0.058	0.065	0.215	0.077

TABLE I: Reprojection error (mean and standard deviation (*std*), both in meters) with and without uncertainty estimation, for both the front and down features.

Finally, similarily to the previous experiment, the actuation constraints (7e) and (7f) are largely satisfied during the whole experiment.

D. Computation Time

Figure 7 presents the histogram of the percentage distribution of the computation time per control step of the NMPC controller in a sequence of experiments such as the ones presented in V-B and V-C, covering a total flight time of 5 minutes. During these flights, 95% of the control steps are faster than $1.7\,[\text{ms}]$, and 99% are faster than $4.5\,[\text{ms}]$. There are outliers: about 1 solving step out of 1000 takes more than $10\,[\text{ms}]$, among which the maximum is of $22\,[\text{ms}]$. However, the analysis of the recorded data shows that these outliers are sparse and immediately followed by fast computation times, and do not jeopardize the stability of the UAV. These outliers do not seem correlated with any particular UAV state and are most likely caused by numerical issues. We empirically tested that a control period artificially downgraded to $40\,[\text{ms}]$ still allows the platform to fly and perform simple maneuvers. The observed outliers are still far below this value.

This illustrates the capability of the proposed framework to compute the torque-level inputs of the UAV, onboard and in real-time.

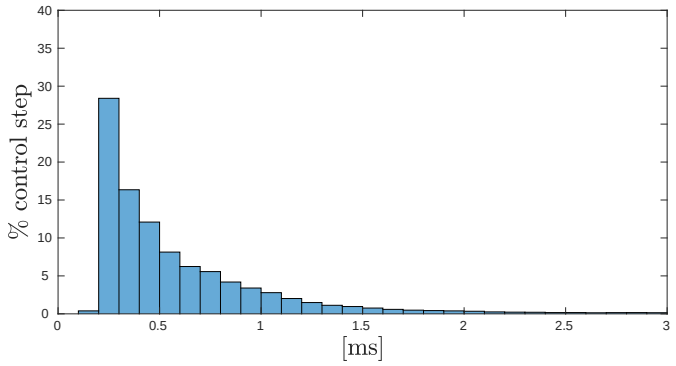

Fig. 7: Histogram of the percentage distribution of computation time per control step for a total flight time of 5 minutes.

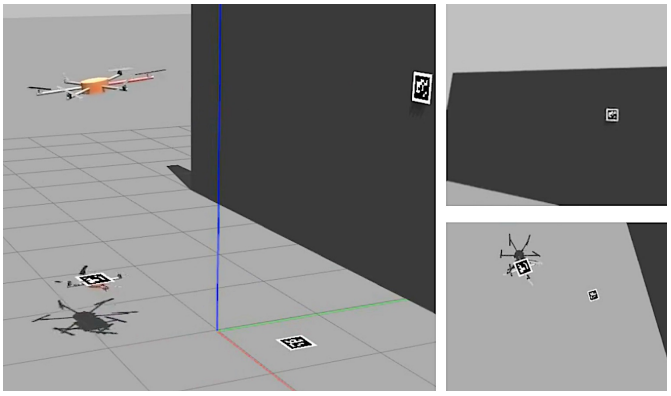

Fig. 8: Simulation of a fully actuated hexarotor with tilted propellers (top left), simultaneously tracking two static targets and a moving one in using two cameras and its 6D motion capability. Right: the two cameras views.

E. Simulation with tilted-propeller hexarotor

Finally, we present simulations with the fully-actuated tilted-propeller hexarotor shown in Figure 8 executed using the Gazebo simulator. Given the additional motion capabilty of a fully actuated platform, e.g., the ability to control independently its translation and orientation, we asked a more challenging task to the UAV. In particular visibility has to be maintained of a fixed marker on a wall with a front-looking camera, and of two markers on the ground with a down-looking camera; of which one is fixed and one is mobile.

This simulation aims at demonstrating the capability of the controller to exploit the full action span of the platform to fulfill both the perceptive constraints while obeying also the actuator constraints. In particular, the tilted-propeller hexarotor is able to hover with nonzero roll and pitch, as long as the motor velocities are within their bounds. Figure 9 shows that at some time, e.g., in the phase between 22 and 40 seconds, the platform takes advantage of this extra-actuation to hover while tilted, up to about 20°. The actuators thrusts and system inputs, presented in Figure 9 and 10, reach their lower and upper bounds during this phase, showing that the actuation constraints are active and the platform is exploited at the maximum of its capability by the proposed controller.

A video of the simulation is contained in the multimedia material attached to this work.

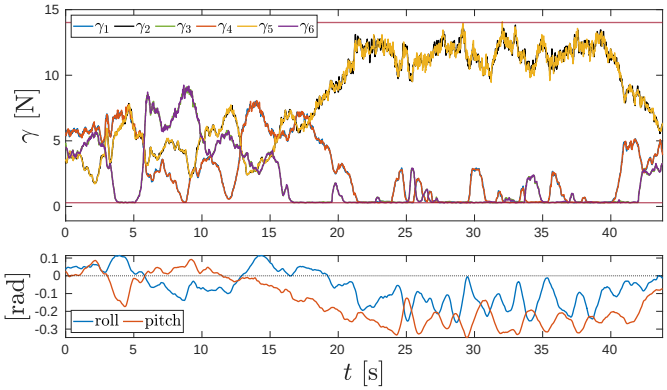

Fig. 9: Top: Thrusts generated by the 6 propellers. Bottom: and the roll and pitch orientations of the UAV.

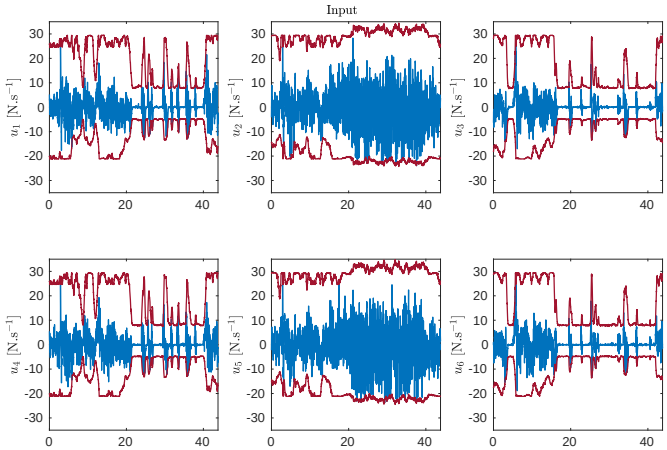

Fig. 10: System inputs for the 6 propellers, reaching their respective bounds.

VI. Conclusions

In this work, we extended previous formulations of perception-aware NMPC to a more realistic modeling, including real actuation constraints as well as perceptive constraints, while including a wider range of UAVs and sensors. We then proposed a simple yet efficient method to cope with intermittent and noisy measurements from a standard monocular camera. The proposed open-source implementation is able to run, at 500Hz, onboard UAVs with standard computation capabilities. This allowed an extensive testing of the software in real conditions, directly commanding the motors without any intermediate unconstrained tracker. The reported experiments propose a quantitative evaluation of the framework in typical scenarios where it would be beneficial.

Such a model-based approach does not come without drawbacks. The modeling of the system has to be done according to each scenario (characteristics of the UAV, number and type of sensors and features, etc.), and the same goes for the tuning of the cost function weights. Also, model-based approaches are more sensitive to variations between the real physical parameters of the platform and the ideal ones. Finally, the optimal solver is unable to find a solution when the requested task is not achievable under the constraints, as it can be the

case with perception. This requires the implementation of some failure-handling strategies.

However, the proposed framework and the increasing availability of numerical tools (such as, e.g., CasADi, MATMPC, and Acado) allows a greater usability of NMPC frameworks. Optimal control and NMPC in particular are a very efficient way to condense several constraints from various semantics, while performing maneuvers. A framework such as the one proposed in this work could be applied to a wide variety of tasks from aerial manipulation to the coverage of an environmental phenomenon.

Possible continuations of this work include the extension to multi-robots systems where the perception load is shared among the agents. Another criticality to be addressed is the failure handling, in the cases where either the platform needs to recover, e.g., under the effect of strong disturbances, by relaxing the perception constraints; or where the perception constraints cannot be satisfied because of the actuation limitations. In both cases, a policy could be implemented to try to recover visibility from the last known measurements and return to the nominal case. Finally, the possibility to integrate the feature tracking capabilities of the framework in a visual state estimation process is also under consideration.

REFERENCES

[1] 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.

[2] 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.

[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] 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.

[5] 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.

[6] K. Lee, J. Gibson, and E. A. Theodorou, "Aggressive perception-aware navigation using deep optical flow dynamics and pixelmpc," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1207–1214, 2020.

[7] K. Mueller, M. Fennel, and G. F. Trommer, "Model predictive control for vision-based quadrotor guidance," in *2020 IEEE/ION Position, Location and Navigation Symposium*, 2020, pp. 50–61.

[8] 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.

[9] D. Bicego, J. Mazzetto, R. Carli, M. Farina, and A. Franchi, "Nonlinear model predictive control with enhanced actuator model for multi-rotor aerial vehicles with generic designs," *Journal of Intelligent & Robotics Systems*, vol. 100, no. 3, pp. 1213–1247, 2020.

[10] M. Jacquet, G. Corsini, D. Bicego, and A. Franchi, "Perception-constrained and motor-level nonlinear MPC for both underactuated and tilted-propeller UAVs," in *2020 IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 4301–4306.

[11] M. Hamandi, F. Usai, Q. Sablé, N. Staub, M. Tognon, and A. Franchi, "Survey on Aerial Multirotor Design: a Taxonomy Based on Input Allocation," 2020, preprint, Conditionally accepted for The International Journal of Robotics Research. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02433405

[12] 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.

[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] R. T. Rodrigues, P. Miraldo, D. V. Dimarogonas, and A. P. Aguiar, "Active depth estimation: Stability analysis and its applications," in *2020 IEEE Int. Conf. on Robotics and Automation*, 2020, pp. 2002–2008.

[15] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze, "Fastdepth: Fast monocular depth estimation on embedded systems," in *2019 IEEE Int. Conf. on Robotics and Automation*, 2019, pp. 6101–6108.

[16] G. Allibert, E. Courtial, and F. Chaumette, "Predictive control for constrained image-based visual servoing," *IEEE Trans. on Robotics*, vol. 26, no. 5, pp. 933–939, 2010.

[17] D. Q. Huynh, "Metrics for 3D rotations: Comparison and analysis," *Journal of Mathematical Imaging and Vision*, vol. 35, no. 2, pp. 155–164, 2009.

[18] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[19] M. Fourmy, D. Atchuthan, N. Mansard, J. Solà, and T. Flayols, "Absolute humanoid localization and mapping based on imu lie group and fiducial markers," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots*, 2019, pp. 237–243.

[20] A. Mallet, C. Pasteur, M. Herrb, S. Lemaignan, and F. Ingrand, "GenoM3: Building middleware-independent robotic components," in *2010 IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 4627–4632.

[21] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: a software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.