Vision-Based Minimum-Time Trajectory Generation for a Quadrotor UAV

Bryan Penin, Riccardo Spica, Paolo Robuffo Giordano, and François Chaumette

***Abstract*— In this paper, we address the problem of using a camera with limited field of view for controlling the motion of a quadrotor in aggressive flight regimes. We present a minimum time trajectory planning method that guarantees visibility of the image features while allowing the robot to undertake aggressive motions for which the usual near-hovering assumption is violated. We exploit differential flatness and B-Splines to parametrize the system trajectories in terms of a finite number of control points, which can then be optimized by Sequential Quadratic Programming (SQP). The control strategy is similar to a Receding Horizon Control (RHC) approach for modifying online the reference trajectory in order to account for noise, disturbances and any non-modeled effect. The algorithm is validated in a physically realistic simulation environment.**

I. INTRODUCTION

Due to their unique combination of affordability and ability to perform hovering flight and acrobatic maneuvers in 3-D space, quadrotors have earned a prominent role among robotic platforms for applications such as surveillance, search and rescue, mapping [1] or inspection [2]. The interest for these platforms has generated a vast literature spanning problems such as state estimation [3], [4] motion control [5], [6] and trajectory planning [7]–[9]. One can also refer to [10] for a general overview. Considerable interest has been devoted, in particular, to the development of control architectures using monocular cameras, since these sensors can provide a very rich information about the surrounding environment while, at the same time, remaining affordable both in terms of cost, payload and energetic requirements.

A popular technique to use visual cues for *directly* controlling the robot motion is *visual servoing* [11]. This strategy was originally developed in the context of industrial robots, which are usually equipped with low-level high-gain control loops that allow neglecting the dynamics of the platform and, e.g., controlling it at the velocity level. Unfortunately, such simplification cannot be extended to flying robots, which, on the contrary, show non-negligible dynamics that make the visual control problem significantly more complex.

Many visual servoing controllers for fully actuated second order systems have been proposed in the literature (see, e.g. [12], [13]); however these solutions cannot be directly applied to quadrotors due to their underactuation (as well-known a quadrotor has only four inputs for controlling its 6-DOF pose in space).

B. Penin and F. Chaumette are with Inria at Irisa and Inria Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France bryan.penin@inria.fr, francois.chaumette@irisa.fr

R. Spica is with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA rspica@stanford.edu

P. Robuffo Giordano is with CNRS at Irisa and Inria Rennes Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes Cedex, France prg@irisa.fr

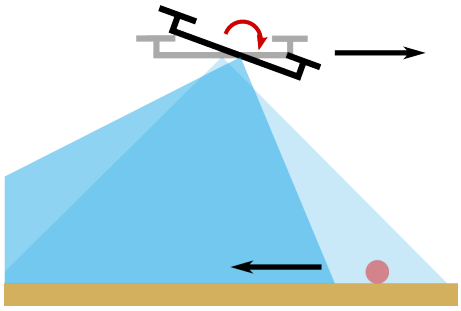

Fig. 1. Effect of underactuation for visual control of a quadrotor: the red target is pushed out of the field of view as the quadrotor moves towards it.

For applications where stability is privileged over performance, like near hovering flight or autonomous landing, one can exploit a *separation principle* to simplify the problem: a high-gain low-level controller uses inertial measurements to regulate the rotational dynamics in order to realize a desired linear acceleration; vision is then used, in a higher level control loop, to generate acceleration commands to control the position of the robot, which is treated as a fully actuated point mass. Examples along this line can be found in [14]–[16]. A similar alternative is to design the visual controller in a "rotation-compensated" camera frame or "virtual plane" as done in [17]–[19].

Unfortunately the underlying assumptions of these works fail for high speed maneuvers. Moreover the quadrotor underactuation is not explicitly taken into account by the control design. This latter aspect is particularly problematic when dealing with cameras with limited field of view, as illustrated in Fig. 1. This figure represents a quadrotor with a down-facing camera that needs to move in the right direction while using the red dot on the ground for visual feedback. In order to accelerate in the desired direction, the robot must necessarily rotate clockwise so as to correctly orient the thrust force generated by the propellers. While doing so, the field of view of the camera will also move and the robot might lose visibility of the red target. Guaranteeing visibility of the visual features is, on the other hand, of paramount importance since losing visual tracking leads to an increasingly poor state estimation (that would just be driven by the odometry, i.e., the onboard IMU) and, thus, possibly, to a controller/task failure.

Intuitively, in order to overcome this problem, the robot can either limit its rotational motion (thus reducing the

acceleration and increasing the time needed to reach the desired position) or compensate the rotation by also moving upwards for increasing the size of the scene projected within the camera field of view.

The literature proposes various solutions to deal with field of view limitations based on the use of potential fields [20], qualitative servoing [21] or switching control laws [22]. In some cases, redundancies can also be successfully exploited for this purpose [23]. However, to our understanding, most of these methods are only suited for relatively slowly-varying fully-actuated systems.

More recent works have considered underactuated robots and sensor limitations in the context of active exploration [24], [25]. However in these works the robot dynamics are simplified and the input constraints (i.e. the propeller speed) are not strictly imposed. An active sensing strategy considering the full quadrotor dynamics was proposed in [26], but without considering strict input constraints. Moreover, these works focus on environment coverage and a correct robot localization and none of them attempts to maintain visibility with respect to a specific set of features, which could, instead, be useful for target tracking applications. Ozawa et. al introduced a controller that takes into account the quadrotor underactuation and uses a virtual spring force to prevent the robot from rotating too much [27]. However, this clearly reduces the quadrotor reactivity and, in any case, does not strictly guarantee the satisfaction of visibility constraints.

To the best of our knowledge, none of the existing solutions can guarantee the satisfaction of visibility constraints while fully exploiting the robot actuation capabilities within the limits imposed by the inputs. A very general approach for dealing with the control of dynamic systems, subject to input and state constraints, and in presence of noise and other non-idealities, is Receding Horizon Control (RHC) or Model Predictive Control (MPC) [28]. This technique consists in calculating the control inputs by continuously solving online a constrained optimization problem based on predictions of the future robot state. MPC has been successfully used for quadrotor flight control in [29], [30], but without using vision. In [31], a visual error is used for the definition of the MPC optimization cost function. However, although the image error may be minimized along the trajectory, the image features are not guaranteed to remain visible.

In this paper, we take inspiration from RHC and present an algorithm that plans *online* minimum time trajectories for a quadrotor equipped with an onboard camera. With respect to previous works, our planner strictly guarantees visibility of the image features while allowing the robot to undertake aggressive motions for which the usual near-hovering assumption is violated. We exploit differential flatness and B-Splines to parametrize the system trajectories in terms of a finite number of control points, which can then be optimized by Sequential Quadratic Programming (SQP). To reject disturbances, we replan trajectories online as new estimations of the robot state become available. As an additional contribution, we show how the B-Spline properties can be cleverly exploited to efficiently adjust previoulsy computed trajectories to the current robot state. This allows to quickly compute good initial guesses at each replanning step thus reducing the number of iterations needed by SQP to converge.

The rest of the paper is structured as follows. Section II introduces the motion and sensing models and states the optimization problem that we wish to solve. Sect. III explains how the differential flatness property and a B-spline parameterization can be used to make the optimization problem suited for numerical resolution. Then, in Sect. IV, we describe the main paper contribution: a trajectory generation strategy that continuously re-plans an optimal trajectory based on current estimations of the robot state. In Sect. V we present some simulation results obtained in a physically realistic simulation environment. Finally, in Sect. VI we draw some conclusions and discuss future perspectives of our work.

II. PROBLEM FORMULATION

A. Preliminaries

With reference to Fig. 2, consider a quadrotor UAV moving in 3D space. Let us define a world frame $\mathcal{W}$ and a body frame $\mathcal{B}$ with origin $\mathcal{O}_\mathcal{B}$ attached to the robot center of mass (COM) and axis $\boldsymbol{z}_\mathcal{B}$ parallel to the propeller rotational axes. Let us also assume, without loss of generality, that the robot COM corresponds to the barycenter of the propellers. The robot state is $\boldsymbol{\chi} = (\boldsymbol{p}, \boldsymbol{R}, \boldsymbol{v}, \boldsymbol{\omega}) \in SE(3) \times \mathbb{R}^6$ where $\boldsymbol{p} \in \mathbb{R}^3$ is the position of the robot COM in $\mathcal{W}$, $\boldsymbol{R} \in SO(3)$ is the rotation matrix from $\mathcal{W}$ to $\mathcal{B}$, $\boldsymbol{v}$ the COM linear velocity expressed in $\mathcal{W}$ and $\boldsymbol{\omega}$ the angular velocity expressed in $\mathcal{B}$. As well-known (see, e.g. [10]), the robot dynamics can then be modeled as:

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{1a}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{g} - \frac{f}{m}\boldsymbol{z}_\mathcal{B} \tag{1b}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}[\boldsymbol{\omega}]_\times \tag{1c}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}([\boldsymbol{J}\boldsymbol{\omega}]_\times \boldsymbol{\omega} + \boldsymbol{\tau}) \tag{1d}$$

where m is the robot mass, $\boldsymbol{J} \in \mathbb{R}^3$ is the inertia tensor, $\boldsymbol{g} \in \mathbb{R}^3$ the (constant) gravity acceleration in the world frame, and $[\cdot]_\times$ the usual skew-symmetric operator. Finally $(f, \boldsymbol{\tau}) \in \mathbb{R}^4$ are the total thrust and torques applied by the propellers, which can be expressed in terms of the individual propeller thrusts $\boldsymbol{u} = (f_1, f_2, f_3, f_4) \in \mathbb{R}^4$ according to the following linear expression

$$\begin{bmatrix} f \\ \boldsymbol{\tau} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -l & 0 & l \\ -l & 0 & l & 0 \\ b & -b & b & -b \end{bmatrix} \boldsymbol{u} \tag{2}$$

where l is the distance between the propellers and the robot COM, and b is the drag constant. Vector $\boldsymbol{u}$ is then the robot control input. The set of admissible control inputs is the box $\mathcal{U} = [f_m, f_M]^4$ with $0 < f_m < f_M$.

Let us also assume the robot to be equipped with an on-board camera whose pose w.r.t. $\mathcal{B}$ is known from a

preliminary calibration. Without loss of generality we assume that the camera is down-facing with optical center in $\mathcal{O}_\mathcal{B}$ and optical axis parallel to $\boldsymbol{z}_\mathcal{B}$. An image processing algorithm provides a measure of the perspective projection of a collection of N fixed 3-D points w.r.t. the frame $\mathcal{B}$ given as follows

$$\boldsymbol{\beta}_i = \frac{\boldsymbol{R}^T(\boldsymbol{r}_i - \boldsymbol{p})}{\boldsymbol{z}_\mathcal{B}{}^T \boldsymbol{R}^T(\boldsymbol{r}_i - \boldsymbol{p})} = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \in \mathbb{P}^2, \quad i = 1, \dots, N \quad (3)$$

where $\boldsymbol{r}_i \in \mathbb{R}^3$ is the *known* position of the features in the inertial frame and $\mathbb{P}^2$ is the space of 3-D homogeneous vectors. We assume that the number of points and their configuration is such that the complete pose $(\boldsymbol{p}, \boldsymbol{R})$ of the robot can be reconstructed using visual information only. In particular, we consider $N = 4$ points on the ground plane since this is sufficient for our 3D case (one could also consider more complex features such as image moments). Finally, we also define the (square) image domain as $\Omega = \{\boldsymbol{\beta} \in \mathbb{P}^2 \; s.t. \; \max(\boldsymbol{\beta}^T \boldsymbol{x}_\mathcal{B}, \boldsymbol{\beta}^T \boldsymbol{y}_\mathcal{B}) \leq \sin(\alpha)\}$ where α is the camera field of view: the measurement (3) is available iff $\boldsymbol{\beta}_i \in \Omega$.

The sensory equipment is completed by an inertial measurement unit (IMU) providing a measure of the robot angular velocities $\boldsymbol{\omega}$ and specific force $\boldsymbol{R}^T(\dot{\boldsymbol{v}} - \boldsymbol{g})$ at a much higher frequency than the camera frame rate. We assume that a state estimator, such as the ones described in [4], [32], uses the visual and inertial measurements to provide an estimation of the current robot state at the IMU rate. Note, however, that between two image frames, the robot pose estimation can only be updated by dead-reckoning of the IMU data. Due to noise and IMU biases, this "inter-frame" estimation is expected to be of much lower accuracy than the one obtained after visual measurements.

B. Problem definition

Given the dynamic model (1), the input transformation (2), and the measurement equation (3), at a generic time t, we seek for a solution to the following optimization problem.

Problem 1: find $T, \boldsymbol{u}(s), s \in [t, t+T]$, such that:

$$\min_{\boldsymbol{u}(s), T} \quad T \quad (4a)$$

$$s.t. \quad \boldsymbol{\chi}(t) = \boldsymbol{\chi}_t \quad (4b)$$

$$\boldsymbol{\chi}(t+T) = \boldsymbol{\chi}^* \quad (4c)$$

$$\dot{\boldsymbol{\chi}} = \boldsymbol{h}(\boldsymbol{\chi}, \boldsymbol{u}) \quad (4d)$$

$$\boldsymbol{u}(s) \in \mathcal{U}, \forall s \in [t, t+T] \quad (4e)$$

$$\boldsymbol{\beta}_i(s) \in \Omega, \forall s \in [t, t+T], i = 1, \dots, N \quad (4f)$$

where $\boldsymbol{\chi}_t$ is the current robot state, $\boldsymbol{\chi}^*$ is the desired one, and (4d) was introduced to represent (1) in a compact form.

Note that Problem 1 is quite general. In particular, it does not impose any constraint on the initial and final states which, e.g., do not have to be *hovering* states (i.e. with $\boldsymbol{R} = \mathbf{I}$, $\boldsymbol{v} \equiv \mathbf{0}$, and $\boldsymbol{\omega} \equiv \mathbf{0}$). However, if both $\boldsymbol{\chi}_t$ and $\boldsymbol{\chi}^*$ are *hovering* states, $\boldsymbol{\beta}_i(t) \in \Omega, \boldsymbol{\beta}_i(t+T) \in \Omega, \forall i = 1, \dots N$, and the *hovering input* $\boldsymbol{u} = (mg/4, mg/4, mg/4, mg/4) \in \mathcal{U}$ is not an isolated point in $\mathcal{U}$, then a solution to Problem 1 always exists. Indeed, in this case, it is always possible to find a sufficiently large T such that the robot moves in near-hovering conditions, along a feasible and *almost* straight trajectory from the initial pose to the desired one [7], [8]. Due to the absence of rotational motions, the linear trajectory in 3-D space is also mapped to linear trajectories of the image features from $\boldsymbol{\beta}_i(t)$ to $\boldsymbol{\beta}_i(t+T)$. Thanks to the convexity of the image domain Ω, this guarantees that the feature visibility will be maintained.

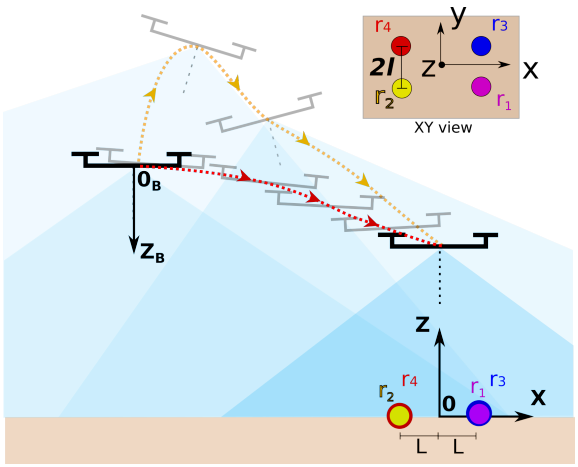

Fig. 2. Consider two pairs of dots on the ground horizontal plane (XY view in the upper right corner). It would be possible to cope with the field of view constraints by planning a near hovering trajectory (e.g. path in red), but in this work we aim at finding a trajectory similar to the path in yellow which is much more dynamic (and with a shorter completion time).

Problem 1 contains non-linear algebraic and differential constraints and, to the best of our knowledge, does not admit an explicit *analytic* solution. As it is often the case in these situations, we then attempt to find a sub-optimal solution using a numeric resolution strategy as discussed in the next section.

III. NUMERICAL RESOLUTION OF PROBLEM 1

In its original form, Problem 1 is not suited for a direct numerical resolution. First of all, the system dynamic equation (4d) represents a non-linear differential equality constraint, which is particularly hard to deal with in a numerical resolution scheme. In addition to this, the search space of the problem (the control input time law $\boldsymbol{u}(t)$) is infinite dimensional. In order to overcome these problems, Sect. III-A will explain how to exploit differential flatness for eliminating constraint (4d), while Sect. III-B will introduce a B-spline parameterization that allows obtaining a finite representation of the search space.

A. Differential flatness

Differential flatness was first introduced by Fliess [33]. A non-linear system is termed *flat* if it is possible to find an explicit expression of all the system states and control inputs in terms of a finite number of variables (called *flat outputs*) and their derivatives up to a finite order. It is well-known that the quadrotor dynamics (1) are flat with flat outputs $\boldsymbol{\gamma} = (\boldsymbol{p}, \psi)^T \in \mathbb{R}^4$ [6], [7], where ψ is the yaw angle from the usual roll/pitch/yaw decomposition of the rotation

matrix $\boldsymbol{R}$. Indeed, under the assumption $f > 0$, one can find an invertible algebraic mapping of the form:

$$\boldsymbol{\chi} = \phi_\chi(\boldsymbol{p}, \boldsymbol{v}, \dot{\boldsymbol{v}}, \ddot{\boldsymbol{v}}, \psi, \dot{\psi}) \tag{5a}$$

$$(f, \dot{f}, \ddot{f}, \boldsymbol{\tau}) = \phi_u(\dot{\boldsymbol{v}}, \ddot{\boldsymbol{v}}, \dddot{\boldsymbol{v}}, \psi, \dot{\psi}, \ddot{\psi}) \tag{5b}$$

The complete expression of (5) can be found in, e.g. [7], and it is not reported here for lack of space. For simplicity of notation, we indicate with $\boldsymbol{\sigma} = (\boldsymbol{p}, \boldsymbol{v}, \dot{\boldsymbol{v}}, \ddot{\boldsymbol{v}}, \dddot{\boldsymbol{v}}, \psi, \dot{\psi}, \ddot{\psi})$ the vector of all quantities appearing on the right side of (5), and with $\boldsymbol{\sigma}_\chi = (\boldsymbol{p}, \boldsymbol{v}, \dot{\boldsymbol{v}}, \ddot{\boldsymbol{v}}, \psi, \dot{\psi})$ only those involved in (5a).

Thanks to differential flatness, one can move the planning problem from the input space to the flat output space (i.e., the problem becomes a static problem): any sufficiently smooth trajectory of the flat outputs is, in fact, guaranteed to be an admissible trajectory for the original system dynamics. This property is extremely interesting for our purposes because it allows avoiding to deal with the non-linear differential equality constraint (4d), which would require the numerical integration of the system dynamics during the numerical optimization phase. A prediction of the state (and, consequently, of the visual measurements) at any time in $[t, t+T]$ can, instead, be computed *algebraically* from the planned flat-output trajectory. For these reasons differential flatness has been widely used for trajectory planning in the past [7], [8], [34].

Remark 1: In [17], [35] the authors exploited some visual features, measured in a virtual image plane parallel to the ground, as flat outputs. On the one hand, this elegant solution allows to move the planning problem directly into the (virtual) image space. On the other hand, with this strategy, it becomes harder to deal with the visibility constraints which need to be brought to the virtual image plane by taking into account the current robot orientation. The existence of a differential flatness transformation for the actual measurements (3) appears, instead, to be very complex, if not impossible, to prove as pointed out in [17]. For this reason, we opted to express the planning problem in terms of the robot states rather than planning directly on the visual feature space.

B. Trajectory parameterization

As already mentioned, in order to numerically solve the optimization problem, it is also necessary to introduce a finite parameterization of the search space (i.e. of the flat outputs). In this paper, we opted for the use of piecewise polynomial functions in the B-spline form [36]. Given a vector of *control points* $\boldsymbol{\theta} = (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{n_p}, \psi_1, \ldots, \psi_{n_\psi}) \in \mathbb{R}^{3n_p+n_\psi}$, and two (fixed) normalized *knot vectors* $\boldsymbol{s}_p \in [0,1]^{l_p}$, $\boldsymbol{s}_\psi \in [0,1]^{l_\psi}$, the flat output trajectories can be represented as:

$$\begin{cases} \boldsymbol{p}(s) = \displaystyle\sum_{k=1}^{n_p} B_k^{d_p}\left(\frac{s-t}{T}\right)\boldsymbol{p}_k \\ \psi(s) = \displaystyle\sum_{k=1}^{n_\psi} B_k^{d_\psi}\left(\frac{s-t}{T}\right)\psi_k \end{cases}, \forall s \in [t, t+T], \tag{6}$$

where B_k^d is the k-th B-spline basis function of order d, which can be computed recursively as described in [33].

Given (5), in order to ensure state continuity and input boundedness, one has to guarantee Lipschitz continuity of $\ddot{\boldsymbol{v}}$ and $\dot{\psi}$ (and continuity of lower order derivatives). This condition can be met by using *open-uniform* distributions of $l = n + d$ knots (i.e. $s_i = 0$, for $i = 1, \ldots, d$, $s_i = 1$, for $i = n+1, \ldots, l$, and $s_d, \ldots, s_{n+1}$ equally spaced in $[0,1]$) and by taking $d = d_p = 4$ for $\boldsymbol{p}$ and $d = d_\psi = 2$ for ψ.

Problem 1 can, finally, be restated as a Nonlinear Program (NLP) as follows.

Problem 2: find $\boldsymbol{\theta}, T$, such that:

$$\min_{\boldsymbol{\theta}, T} \quad T \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{\sigma}_\chi(t) = \boldsymbol{\sigma}_{\chi_t}, \tag{7b}$$

$$\boldsymbol{\sigma}_\chi(t+T) = \boldsymbol{\sigma}_{\chi^*}, \tag{7c}$$

$$\boldsymbol{\beta}_i(s) \in \Omega, \forall s \in [t, t+T], i = 1, \ldots, N, \tag{7d}$$

$$\boldsymbol{u}(s) \in \mathcal{U}, \forall s \in [t, t+T], \tag{7e}$$

where $\boldsymbol{\sigma}_{\chi_t} = \phi_\chi^{-1}(\boldsymbol{\chi}_t)$ and $\boldsymbol{\sigma}_{\chi^*} = \phi_\chi^{-1}(\boldsymbol{\chi}^*)$.

At this point, any general-purpose optimization strategy can be used to find a numerical solution to Problem 2. For this paper, we exploited the Sequential Quadratic Programming optimization routine implemented in NLOPT [37]. Unfortunately, due to the non trivial non-linearity of (7d–7e), Problem 2 cannot be proven to be convex. The optimization will thus, in general, return a local minimum.

IV. RECURSIVE ONLINE CONTROL

Once Problem 2 is solved, the resulting flat output trajectory could be used in (5) for computing the control inputs $\boldsymbol{u}$ to be fed to the system. In ideal conditions, thanks to the flatness property, using these inputs would result in the system following *exactly* the planned trajectory. In practice, however, different sources of disturbance (e.g. noise, miscalibrations, neglected dynamics, and so on) will make the robot to quickly diverge from the planned trajectory when using such an open-loop control strategy. In order to cope with these uncertainties and disturbances, we then incorporate a *feedback* action in the proposed optimization scheme.

A first possibility would be to feed the planned trajectory into a low-level trajectory tracker such as the ones described in [5], [7]. On the one hand, this would allow to reject, to some extent, the disturbances acting on the system. On the other hand, however, the optimality of the resulting trajectory could be compromised and, more importantly, the visibility constraints (7d) could be violated. In this paper, instead, we take inspiration from Model Predictive Control [28] to perform an *on-line* re-planning of an optimal trajectory by solving Problem 2 each time a new visual measurement is available. By doing so, we expect to improve the system performance while, more importantly, ensuring the satisfaction of the visibility constraint (7d).

A. Trajectory re-planning strategy

The re-planning strategy is best explained by a visual example, shown in Fig. 3.

Let us assume that, in a previous planning step, at time $t = t_{k-1}$, the resolution of Problem 2 generated a trajectory

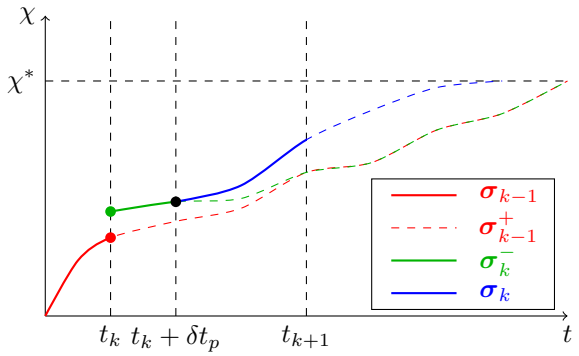

Fig. 3. Single instance of the re-planning process. The red line represents the trajectory computed in a previous planning iteration. The robot is following this trajectory when, at time t_k, a visual measurement and a new state estimation become available (green dot). The red trajectory is split and clamped to this measurement, resulting in the green line. The first part of this latter is immediately used as reference for the controller. The second part (the dashed green line) is fed as initial guess to the solver of Problem 2, and also used to predict the state in which the system will be at time $t_k + \delta t_p$, when the optimization will be over. Finally, the blue line is the new optimal trajectory resulting from the numerical resolution of Problem 2. The process is repeated again at t_{k+1}, when a new measurement is available.

$\boldsymbol{\sigma}_{k-1}$, represented in red in the figure. The system is now at time $t = t_k$ and a new visual measurement becomes available to be used in the innovation step[1] of the state observer to produce an estimation of the current system state $\hat{\boldsymbol{\chi}}_t$. This estimation will, in general, be different from the expected system state $\Phi_\chi(\boldsymbol{\sigma}_{k-1}(t))$ due to the non-idealities mentioned above. A new optimal trajectory should, hence, be planned by solving Problem 2 and using the current state estimate to compute the initial condition $\boldsymbol{\sigma}_{\chi_t}$.

Unfortunately, the resolution of Problem 2 requires a non-negligible time to complete. This time will, in general, vary, depending on the quality of the initial guess for the optimization variables, on the number of necessary iterations and on the available computational resources. Here, for simplicity, we assume that the processing will be concluded after, at most, a constant maximum duration δt_p, possibly by introducing a watchdog timer and accepting an intermediate sub-optimal solution.

For computing the system control inputs while the optimization is running, we simply "adapt" the previous trajectory to the new initial conditions by using a fast procedure that does not involve the resolution of Problem 2. First of all, we *split* the trajectory $\boldsymbol{\sigma}_{k-1}$ at time t_k, as described in Sect. IV-B, to extract only its second part $\boldsymbol{\sigma}^+_{k-1}$ (the dashed red curve in Fig. 3). Then, we look for a new trajectory $\boldsymbol{\sigma}^-_k$ (represented in green in Fig. 3) that is "as close as possible" to $\boldsymbol{\sigma}^+_{k-1}$, but starts from $\Phi^{-1}_\chi(\hat{\boldsymbol{\chi}}_t)$. Details about this step are provided in Sect. IV-C. Note that this "temporary" trajectory $\boldsymbol{\sigma}^-_k$ is sub-optimal and its calculation does not take into account any of the actuation and visibility constraints (7d–7e), which, as a consequence, could be violated. However, we accept this risk in order to be able to provide an immediate update of the reference trajectory to the new state estimation while a better solution is being computed by appropriately resolving Problem 2 as follows.

During the optimization process, the system will, most probably, move away from the current state $\boldsymbol{\chi}_t$. As a consequence, if $\boldsymbol{\chi}(t)$ were used as initial condition in (4b) (or, equivalently, (7b)), the newly planned trajectory would not start from the actual state of the robot at time $t + \delta t_p$. We mitigate this problem by using the trajectory $\boldsymbol{\sigma}^-_k$ also to predict (by a simple B-spline evaluation) the value of the flat outputs corresponding to the state $\hat{\boldsymbol{\chi}}_{t_{opt}}$ in which the system will be when the optimization will be over. This value is used as initial condition in Problem 2.

Finally, since we use recursive optimization methods to find a solution to Problem 2, we also need to provide an initial guess for the optimal trajectory. This initial guess is computed by splitting the trajectory $\boldsymbol{\sigma}^-_k$ at time $t+\delta t_p$ (green dashed line in Fig. 3) as described in Sect. IV-B and taking the second part (green dashed line in Fig. 3) of the trajectory.

The optimization can finally run and a new optimal trajectory (the blue one in Fig. 3) will be generated. Such trajectory will be used to control the system starting from time $t + \delta t_p$ until a new measurement becomes available at time $t = t_{k+1}$. At the arrival of a new measurement the above procedure is repeated.

This strategy allows to re-plan online an optimal trajectory each time a new visual measurement is available. Each one of the generated trajectories could be used directly in (5b) to calculate the robot inputs. As already mentioned, however, an alternative possibility is, instead, to use them as references for a fast trajectory tracker. This second possibility is appealing because it allows to fully exploit the sensing capabilities of the robot: between two visual measurements, in fact, an estimation of the quadrotor state can be obtained, at a much higher frequency, by using the IMU for dead reckoning. A fast trajectory tracker can, thus, use this information to reduce the effect of non-idealities between two planning steps.

Note that, as the quadrotor approaches the desired state, the planning distance and time horizon tend to zero, potentially introducing numerical issues in the resolution of Problem 2. To overcome this problem, when the system is close to the desired goal, we deactivate the re-planning and directly feed the trajectory tracker with the desired state $\boldsymbol{\chi}^*$.

B. B-spline splitting

An advantage of using B-spline trajectories for motion planning is that there exist efficient algorithms to perform different manipulations on their shape. One such manipulation, that we perform multiple times in the recursive algorithm described in Sect. IV-A, is the *splitting*. Details about how to split a B-spline curve at a point and how to calculate the knots and control points of the resulting parts can be found in many sources, such as [38].

An undesirable effect of the splitting operation is that it also modifies the knot sequence and possibly (depending on the position of the split) even eliminates some knots. In order to maintain a constant number of uniformly distributed

[1]Note that we trigger the planning at camera rate and not at the estimation rate. This is motivated by computational limitations and by the fact that, as already mentioned, the inter-frame estimation obtained by dead reckoning is expected to have a much lower accuracy.

knots (and thus a constant number of control points acting "evenly" on the whole spline length), after the split, we perform a sequence of *knot insertion* and *knot removal* operations (see [38]) meant to redistribute the knots of the new trajectory evenly.

C. Adapting previous trajectories to new initial conditions

In this section we describe how to efficiently "adapt" a previously computed B-spline trajectory (e.g. the trajectory $\boldsymbol{\sigma}^+_{k-1}$ represented by a red dashed line in Fig. 3) to a new estimation of the current robot state (green dot in Fig. 3). To perform this operation we exploit two important properties of B-splines. The *local support* property stands that the shape of the curve in a knot span (s_k, s_{k+1}) is only determined by a subset of d of the B-spline control points. The *convex hull* property guarantees, instead, that in each knot span, the spline curve is locally contained in the convex hull of the same subset of control points. In practice this allows to conclude that changing the first control points (those determining the initial state of the system) will not affect the shape of the spline towards its end (in particular the final system state will not change) and that two splines with similar control points (according to some norm) are also geometrically close to each other.

Given a spline $\boldsymbol{\sigma}^+_{k-1}$, with control points $\boldsymbol{\theta}$, the control points $\boldsymbol{\theta}^-$ of the new spline $\boldsymbol{\sigma}^-_k$ can then be computed by solving the following linear quadratic optimization.

Problem 3: find a vector of control points $\boldsymbol{\theta}^-$ such that

$$\min_{\boldsymbol{\theta}^-} \quad \frac{1}{2}\sum_{j=1}^{n_p} \left\|\boldsymbol{p}_j - \boldsymbol{p}_j^-\right\|^2 + \frac{1}{2}\sum_{j=1}^{n_\psi} \left\|\psi_j - \psi_j^-\right\|^2 \tag{8}$$

$$s.t. \quad \boldsymbol{\sigma}_\chi(t) = \boldsymbol{\sigma}_{\chi_t}, \tag{9}$$

Note that Problem 3 does not take into account the actuation and visibility constraints in (7d–7e). While we cannot formally guarantee that these constraints will not be violated, we want to stress that the resulting trajectory is only used for a short amount of time, namely the time needed for the numerical resolution of Problem 2. Introducing a saturation of the control commands one still guarantees the satisfaction of (7e) at the cost of introducing a deviation of the robot from its nominal trajectory. Finally, by introducing some *security margins* in the definition of Ω, one can also reduce the probability of losing feature track.

V. SIMULATION RESULTS

In this section we report the results obtained by using our planning method in a physically realistic simulation environment. A video of the simulation is attached to the paper.

The quadrotor dynamics were simulated using V-Rep[2] with a time step of 6 ms. The planning strategy described in Sects. III and IV was implemented in C++ and the SQP method of NLOPT [37] was used to numerically resolve Problem 2. The generated trajectories were sent to TeleKyb [39] which then computed the actual control inputs using controller [5], [32].

[2]http://www.coppeliarobotics.com/

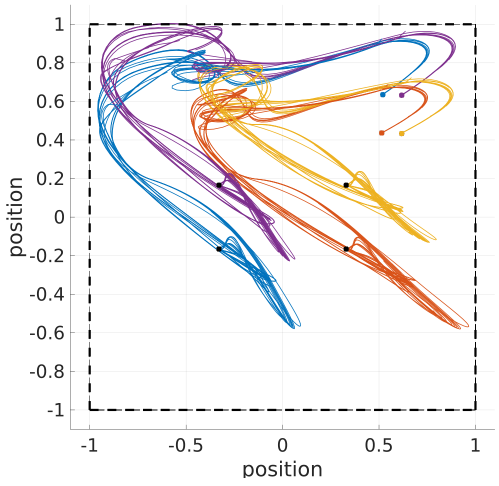

Fig. 5. Image feature trajectories planned at different planning steps. Four dashed segments represent the boundaries of the image domain. The image features are initially in the upper right corner.

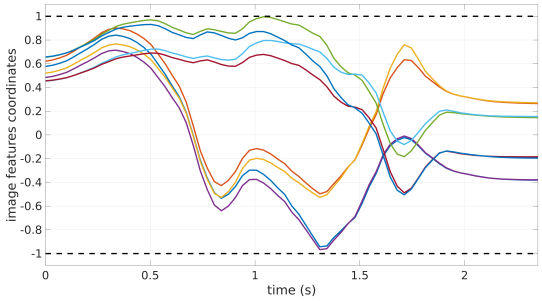

Fig. 6. Actual image feature coordinates measured during the replanning. The horizontal dashed lines represent limits of the image domain.

We simulated visual measurements at a rate of 15Hz for four targets positioned in $(\pm 0.2, \pm 0.1)$. In our implementation, each planning operation (resolution of Problem 2) takes about 30ms during which the system uses an adaption of a previously planned trajectory, obtained by resolving Problem 3. Thus, a new trajectory is sent to the controller at the rate of 30Hz.

The simulated camera had a field of view of 90 degrees ($\alpha = \pi/4$) and each propeller could generate thrusts between 0.1 N and 7 N. For realism purposes, we introduced a Gaussian noise into the state measurements (up to 2% absolute error) and into the motors thrust sent by the controller (up to 5% absolute error). We also purposely used different inertial parameters for the replanning algorithm and for the actually simulated quadrotor in V-Rep in order to introduce presence of (typical) modelling errors between planned trajectory and actual execution. In particular, we used the following values:

	mass	Inertia matrix (diagonal)
Planning	1.0	(0.01562 0.01562 0.03125)
Simulation	1.08	(0.016 0.0145 0.027)

Table 1. Inertial parameters used for the replanning and in V-Rep

Figure 4 shows some snapshots of the simulation. The robot started from an initial hovering state at $\boldsymbol{p}$ =

Fig. 4. Successive snapshots taken from V-Rep at different time instants. The straight line represents the vertical axis of the camera, the blue line is the planned trajectory and the red line is the actual system trajectory. The camera view is shown in the upper right corner.

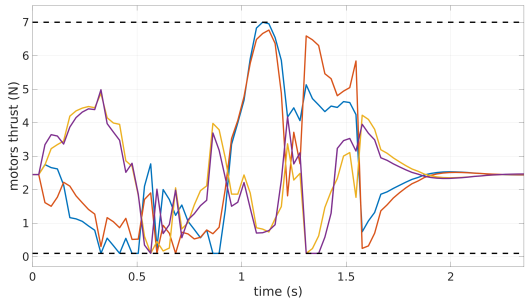

Fig. 7. Motor thrusts evolution for the four propellers with horizontal dashed lines representing the actuation domain $\mathcal{U} = [0.1, 7]$.

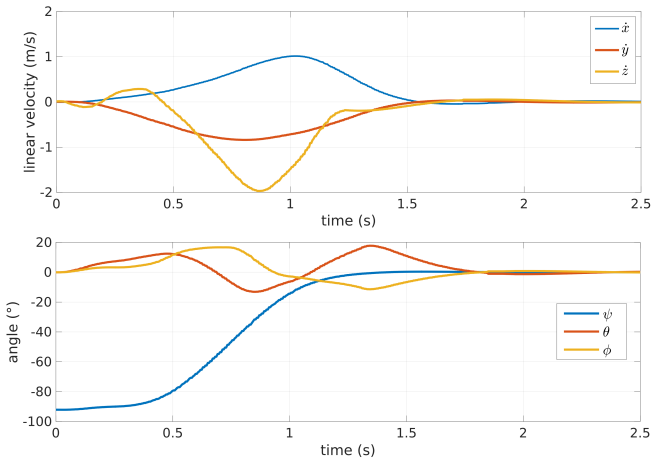

Fig. 8. Llinear velocity (upper figure) and pitch and roll angles (bottom figure) during motion.

$(-1.1, 1.1, 2)$ and $\psi = 1.6$ rad and was required to reach another hovering state with $\boldsymbol{p}^* = (0, 0, 0.6)$ and $\psi^* = 0$. The solid red line in Fig. 4 shows the resulting quadrotor trajectory in space while the blue line represents the currently planned trajectory. Figure 5 shows the predicted evolution (given the currently planned trajectories) of the four points in the image plane at equally spaced time instants. Notice how, due to unmodeled disturbances, the evolution of the system deviates from the expected one thus requiring to continuously replan new trajectories. The actual evolution of the four image point coordinates is shown in Fig. 6 whereas Fig. 7 shows the thrust generated by each propeller. The dashed lines in Figs. 5 to 7 represent the constraints.

The robot was able to accomplish the task in a total time of approximately 2.3 s over which the trajectory planning algorithm was triggered 34 times.

During motion, the quadrotor reached a translational speed up to 1.0 m/s along the X axis, and rotations up to 20 deg as illustrated in Fig. 8. From Fig. 6 one can see that the features moved very close to the limits of the field of view. Finally Fig. 7 shows that also the motor thrusts hit the actuation limits. These results clearly show that the performed trajectory was rather aggressive and that the actuation and sensing capabilities of the robot were exploited. Therefore, we showed that in the presence of modelling uncertainties and noise, the feedback introduced by updating the reference trajectory was able to reject some of these disturbances while satisfying the several constraints. Indeed, we encourage the reader to watch the video attached to this paper: there, we show how an 'open-loop' execution of the initially planned trajectory quickly fails to meet the visibility constraints because of the (purposely introduced) actuation noise and model uncertainties. On the other hand, as discussed, the online replanning allows gaining a sufficient level of robustness against these non-idealities.

VI. CONCLUSION AND FUTURE WORK

In this paper, we addressed the problem of controlling the motion of a quadrotor UAV using an on-board camera. In contrast with the previous literature, we proposed a method that explicitly deals with the robot underactuated dynamics and with the system limitations in terms of propeller thrusts and camera field of view. In particular, we presented a minimum time trajectory planning method that guarantees visibility of the image features while allowing the robot to undertake aggressive motions for which the usual near-hovering assumption is violated.

We exploited differential flatness and a B-Splines parameterization to reformulate the optimization problem in terms of a finite number of parameters (the B-spline control points) that can be numerically optimized by Sequential Quadratic Programming (SQP). The strategy also uses a Receding Horizon Control (RHC) approach for modifying online the reference trajectory in order to account for noise, disturbances and any other non-modeled effect. Moreover, it cleverly exploits B-spline properties to efficiently compute good initial guesses for each planning step by adapting previous solutions.

The algorithm was validated in a physically realistic simulation environment where the strategy proved to be able to generate aggressive maneuvers and to reach the limits of

the robot capabilities while rejecting to some extent the noise and the modelling uncertainties.

Future work includes allowing short visibility violations as we perform aggressive trajectories and taking state estimation uncertainties into account. Since we are interested in highly dynamic maneuvers, in the future we also plan to consider the dynamics of the propellers. If they can be modeled as linear, the full system remains flat and the overall strategy can be maintained also in this case. Finally, the full setup will be validated in real experimental conditions with a quadrotor UAV equipped with a down-facing camera.

ACKNOWLEDGEMENTS

The authors wish to thank Pierre-Brice Wieber for his help and useful discussions.

REFERENCES

[1] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *15th Int. Symp. on Robotics Research*, vol. 2, Flagstaff, AZ, Aug. 2011.

[2] I. Sa, S. Hrabar, and P. Corke, "Inspection of pole-like structures using a vision-controlled vtol uav and shared autonomy," in *2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Chicago,IL, Sep. 2014, pp. 4819–4826.

[3] A. Martinelli, "Vision and IMU Data Fusion: Closed-Form Solutions for Attitude, Speed, Absolute Scale, and Bias Determination," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 44–60, 2012.

[4] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli, and R. Siegwart, "Monocular Vision for Long-term Micro Aerial Vehicle State Estimation: A Compendium," *J. of Field Robotics*, vol. 30, no. 5, pp. 803–831, 2013.

[5] T. Lee, M. Leokyand, and N. H. McClamroch, "Geometric tracking control of a quadrotor UAV on SE(3)," in *49th IEEE Conf. on Decision and Control*, Atlanta, GA, Dec. 2010, pp. 5420–5425.

[6] V. Mistler, A. Benallegue, and N. K. M'Sirdi, "Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback," in *10th IEEE Int. Symp. on Robots and Human Interactive Commun.*, Bordeaux, Paris, France, Sep. 2001, pp. 586–593.

[7] D. Mellinger and V. Kumar, "Minimum Snap Trajectory Generation and Control for Quadrotors," in *2011 IEEE Int. Conf. on Robotics and Automation*, Shangai, China, May 2011, pp. 2520–2525.

[8] R. Spica, A. Franchi, G. Oriolo, H. H. Bülthoff, and P. Robuffo Giordano, "Aerial Grasping of a Moving Target with a Quadrotor UAV," in *2012 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Vilamoura, Portugal, Oct. 2012, pp. 4985–4992.

[9] J. Yu, Z. Cai, and Y. Wang, "Minimum jerk trajectory generation of a quadrotor based on the differential flatness," in *Guidance, Navigation and Control Conference (CGNCC), 2014 IEEE Chinese*. IEEE, 2014, pp. 832–837.

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

[11] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *2006 IEEE Robotics & Automation Mag.*, vol. 13, no. 4, pp. 82–90, December 2006.

[12] R. Mahony and S. Stramigioli, "A port-Hamiltonian approach to image-based visual servo control for dynamic systems," *Int. J. of Robotics Research*, vol. 31, no. 11, pp. 1303–1319, 2012.

[13] E. Zergeroglu, D. M. Dawson, M. S. de Querioz, and A. Behal, "Vision-based nonlinear tracking controllers with uncertain robot-camera parameters," *IEEE/ASME Trans. on Mechatronics*, vol. 6, no. 3, pp. 322–337, Sep 2001.

[14] N. Guenard and T. Hamel, "A Practical Visual Servo Control for an Unmanned Aerial Vehicle," *IEEE Trans. on Robotics*, vol. 24, no. 2, pp. 331–340, 2008.

[15] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-Based Visual Servo Control of the Translation Kinematics of a Quadrotor Aerial Vehicle," *IEEE Trans. on Robotics*, vol. 25, no. 3, pp. 743–749, 2009.

[16] R. Mebarki, V. Lippiello, and B. Siciliano, "Nonlinear visual control of unmanned aerial vehicles in gps-denied environments," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 1004–1017, 2015.

[17] J. Thomas, G. Loianno, J. Polin, K. Sreenath, and V. Kumar, "Toward Autonomous Avian-Inspired Grasping for Micro Aerial Vechicles," *Bioinspiration & biomimetics*, vol. 9, no. 2, p. 025010, 2014.

[18] H. Jabbari, G. Oriolo, and H. Bolandi, "Dynamic IBVS Control of an Underactuated UAV," in *2012 IEEE Int. Conf. on Robotics and Biomimetics*, Guangzhou, China, Dec. 2012, pp. 1158–1163.

[19] H. Jabbari and G. Oriolo and H. Bolandi, "An Adaptative Scheme for IBVS of an Underactuated UAV," *Int. J. of Robotics Research*, vol. 29, no. 1, pp. 92–104, 2014.

[20] O. Kermorgant and F. Chaumette, "Dealing With Constraints in Sensor-Based Robot Control," in *2006 IEEE Trans. on Robotics*, vol. 30, no. 1, February 2014, pp. 244–257.

[21] A. Remazeilles, N. Mansard, and F. Chaumette, "A Qualitative Visual Servoing to ensure the Visibility Constraint," in *2006 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, Beijing, China, October 2006, pp. 4297–4303.

[22] N. R. Gans and S. A. Hutchinson, "Stable visual servoing through hybrid switched-system control," *IEEE Trans. on Robotics*, vol. 23, no. 3, pp. 530–540, 2007.

[23] E. Marchand and G. D. Hager, "Dynamic sensor planning in visual servoing," in *1998 IEEE Int. Conf. on Robotics and Automation*, Leuven, Belgium, May 1998, pp. 1988–1993.

[24] B. Charrow, S. Liu, V. Kumar, and N. Michael, "Information-theoretic mapping using cauchy-schwarz quadratic mutual information," in *2015 IEEE Int. Conf. on Robotics and Automation*, May 2015, pp. 4791–4798.

[25] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments," in *2015 IEEE Int. Conf. on Robotics and Automation*, May 2015, pp. 1071–1078.

[26] M. W. Achtelik, S. Lynen, S. Weiss, M. Chli, and R. Siegwart, "Motion- and uncertainty-aware path planning for micro aerial vehicles," *Journal of Field Robotics*, vol. 31, no. 4, pp. 676–698, 2014.

[27] R. Ozawa and F. Chaumette, "Dynamic visual servoing with image moments for a quadrotor using a virtual spring approach," in *2011 IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 5670–5676.

[28] H. Michalska and D. Q. Mayne, "Robust receding horizon control of constrained nonlinear systems," *IEEE transactions on automatic control*, vol. 38, no. 11, pp. 1623–1633, 1993.

[29] M. W. Mueller and R. DAndrea, "A model predictive controller for quadrocopter state interception," in *European Control Conference*, 2013, pp. 1383–1389.

[30] K. Alexis, C. Papachristos, G. Nikolakopoulos, and A. Tzes, "Model predictive quadrotor indoor position control," in *Control & Automation (MED), 2011 19th Mediterranean Conference on*. IEEE, 2011, pp. 1247–1252.

[31] M. Sheckells, G. Garimella, and M. Kobilarov, "Optimal Visual Servoing for Differentially Flat Underactuated Systems," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2016.

[32] R. Spica, P. Robuffo Giordano, M. Ryll, H. Blthoff, and A. Franchi, "An Open-Source Hardware/Software Architecture for Quadrotor UAVs," in *2nd Workshop on Research, Education and Development of Unmanned Aerial System*, Compigne, France, Nov. 2013.

[33] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *International journal of control*, vol. 61, no. 6, pp. 1327–1361, 1995.

[34] M. B. Milam, "Real-time Optimal Trajectory Generation for Constrained Dynamical Systems," Ph.D. dissertation, California Institute of Technology, 2003.

[35] G. Allibert, E. Courtial, and Y. Touré, "A Flat Model Predictive Controller for Trajectory Tracking in Image Based Visual Servoing," *IFAC Proceedings Volumes*, vol. 40, no. 12, pp. 993–998, 2007.

[36] C. D. Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.

[37] S. G. Johnson, "The nlopt nonlinear-optimization package." http://ab-initio.mit.edu/nlopt.

[38] N. M. Patrikalakis and T. Maekawa, *Shape Interrogation for Computer Aided Design and Manufacturing*. Springer, 2009.

[39] V. Grabe, M. Riedel, H. H. Bülthoff, P. R. Giordano, and A. Franchi, "The TeleKyb Framework for a Modular and Extendible ROS-based Quadrotor Control," in *Mobile Robots (ECMR), 2013 European Conference on*. IEEE, 2013, pp. 19–25.