2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Fast Model Predictive Image-Based Visual Servoing for Quadrotors*

Pedro Roque[1], Elisa Bin[1], Pedro Miraldo[2], and Dimos V. Dimarogonas[1]

***Abstract*— This paper studies the problem of Image-Based Visual Servo Control (IBVS) for quadrotors. Although the control of quadrotors has been extensively studied in the last decades, combining the IBVS module with the quadrotor's dynamics is still hard, mainly due to the under-actuation issues related to the quadrotor control as opposed to the 6 DoF control outputs generated by the IBVS modules. We propose an alternative formulation to solve this problem, by particularly using linear Model Predictive Control (MPC), that allows us to relax the UAVs under-actuation issues. Stability guarantees of the proposed scheme are presented. The proposed model is validated with synthetic data and tested in a real UAV's setup.**

I. INTRODUCTION

Guided by its applications, *e.g.*, search & rescue [1], inspection [2], [3], mapping & exploration [4], photography [5], agriculture [6], and construction [7], research on Unmanned Aerial Vehicle (UAV) has been one of the more active topics in robotics. There are many types of UAVs that can be categorized depending on their dimension and number/type of actuators. The two main categories are fixed-wing *v.s.* rotary-wing UAVs. In this work we focus on the latter, namely the use of quadrotors. These vehicles are characterized by having four rotational propellers. Quadrotor UAVs are underactuated [8]. The reason for this is the fact that there is no direct actuation providing motion on the xy–direction. In this paper, we tackle the control of a quadrotor using computer vision, namely Visual Servoing (VS) [9], [10]. VS aims at providing control inputs for a camera (attached to a robotic agent) to take it from an initial to a goal position in the environment. However, the output of the VS module provides six degrees of freedom control inputs (linear and angular velocities) to the camera, which is not suitable for the quadrotor's under-actuated dynamics.

This paper tackles the under-actuation issues of the quadrotor Visual Servoing using Model Predictive Control (MPC) [11]. In [12] a preliminary work to navigate a UAV with IBVS is presented. We model a linear MPC scheme with the quadrotor dynamics and the VS control velocity goal, providing the actuation inputs to the UAV agent, *i.e.*, roll, pitch, yaw, and thrust quantities. Different from existing works, such as [13], we focus on the approximation using a linear MPC, with the purpose of having a fast, low cost solver. To the best of our knowledge, this is the first time where VS is successfully combined with MPC for low power quadrotors, enabling its application in real-time scenarios. The conditions for the feasibility and stability of the linear MPC are studied, and the proposed control scheme is tested in simulated and real scenarios.

In addition to fixing the under-actuation issues of quadrotors, with the proposed framework we can efficiently include additional restrictions on the motion of the robotic agent. An advantage of the proposed formulation with respect to the state-of-the-art is the possibility to include a system model to track velocities that non-holonomic systems cannot, in general, track optimally. Moreover, the inclusion of state and control constraints means that this framework can handle actuation limits from the hardware. Also worth noting, it is possible to use previous control predictions to actuate the system, in the event of a feature detection failure.

Despite the related background, the coupling of VS with MPC to model the UAV's dynamics has not yet fully studied. In [14], [15], the authors propose the use of MPC for aircraft collision avoidance and use single point features to guide the robotic agent around the object, along a conical spiral trajectory. [16] proposes an autonomous vision-based landing of helicopter-based unmanned aerial vehicle (UAV). A real-time MPC-based optimization scheme to drive the robotic agent through the requested flight pattern is proposed. In [13], the authors present an observer-based model predictive control scheme for quadrotors to explicitly bound the roll and pitch angles and alleviate the feature loss on large rotation.

Other relevant works on similar topics are available in the literature. [17], [18] concerns Image-Based Visual Servoing (IBVS) for aerial vehicles. The authors use a PID to track a desired thrust and angular momentum's instead of MPC, and no actuation limits are considered. [19] also did not account for actuation limits and focuses on planning and low-level feedback control. [20]–[22] aim at a high level planning, instead of low-level control. The high computational cost involved required a powerful off-board computer. In [23], the authors propose a nonlinear MPC with visual servoing (focusing the center of the image on a given set of points). However, no guarantees of stability are given, which we do due to our linear MPC nature.

In addition to the use of MPC schemes for the control of aircrafts using computer vision, MPC was used in several

*This work was supported by the H2020 ERC Grant BUCOPHSYS, the EU H2020 Co4Robots project, the Swedish Foundation for Strategic Research (SSF), the Swedish Research Council (VR) and the Knut och Alice Wallenberg Foundation (KAW), the Wallenberg AI, Autonomous Systems and Software Program (WASP), and by the Portuguese National Funding Agency for Science, Research and Technolog through the LARSyS - FCT Plurianual funding 2020.

[1]P. Roque, E. Bin, and D. V. Dimarogonas are with the Division of Decision and Control Systems, KTH Royal institute of Technology, Stockholm, Sweden. E-Mail: {padr,ebin,dimos}@kth.se

[2]P. Miraldo is with the Institute for Systems and Robotics (LARSyS), Instituto Superior Técnico, Lisboa, Portugal. This work was done when P. Miraldo was at KTH. E-Mail: pedro.miraldo@tecnico.ulisboa.pt

works to help with the issues of the IBVS. In [24], [25], the authors propose control schemes for IBVS for manipulators with eye-in-hand configurations. [26] studies a similar problem, in which the authors include the possibility of having a wide field of view camera in the manipulator's hand effector. In [27], the authors use an MPC to simultaneously solve the problem of feature correspondence and provide control inputs, and in [28] a Nonlinear MPC is used to guide fixed-wing aircraft around an obstacle. To the best of our knowledge, there is no work that studies the use of MPC to deal with the under-actuation issues raised in the VS control loop in quadrotors.

The paper is organized as follows. In Sec. II we present the background and problem statement. In Sec. III, we propose the model predictive image-based visual servoing control scheme. The experimental results with simulated and real data are presented in Sec. IV. Sec. V concludes the paper.

II. PROBLEM STATEMENT

In this paper, we use bold small letters to represent vectors. Matrices are denoted by bold capital letters. Regular letters denote scalars. The hat in a vector means that it is the respective estimate (*e.g.*, $\hat{\mathbf{a}}$ denotes the estimated coordinates of $\mathbf{a}$), and the tilda the estimation error (*i.e.*, $\tilde{\mathbf{a}} := \hat{\mathbf{a}} - \mathbf{a}$). Rotation matrices are defined as $\mathbf{R}_a^b$, where a is the origin frame and b the target frame. When the origin/target frame is the inertial frame, we omit the corresponding letter. For two sets A and B, $A \oplus B$ denotes their Minkowski sum [29].

Next, we present the background and the problem statement. We start by describing the 6D visual servoing model, Sec. II-A. Then, we present the quadrotor's system dynamics and present the problem studied in this paper, Sec. II-B.

A. 6D Visual Servoing

Visual Servoing (VS) is the task of controlling a robot using computer vision. There are two main types of VS schemes, position-based and image-based. The former requires a two step procedure, in which the first gets the pose of the robot, and in the second, the control is done in 3D. Image-based VS aims at controlling the robot directly using image features (2D). For a detailed comparison of the two control schemes, see [10]. In this paper we aim at solving the Image-Based Visual Servoing (IBVS).

The goal of IBVS is to minimize an error

$$\tilde{\mathbf{s}}(t) := \mathbf{s}(\mathbf{m}(t), \mathbf{a}) - \mathbf{s}^*, \tag{1}$$

where $\mathbf{m}(t)$ is a set of images features, $\mathbf{s}(\mathbf{m}(t), \mathbf{a})$ is a vector of k visual features in the image plane, $\mathbf{a}$ is the set of camera intrinsic parameters, and $\mathbf{s}^*$ represents the desired feature's position vector. Here we consider the case of a fixed goal pose and motionless target, *i.e.*, $\mathbf{s}^*$ is constant and changes in $\tilde{\mathbf{s}}$ depend only on the camera motion.

Given the camera extrinsic parameters [30] (*i.e.*, the rigid transformation from the base of a vehicle until the center of the perspective camera) the spacial velocity of the camera is expressed by $\boldsymbol{\upsilon}_c = \left[\mathbf{v}_c^T \boldsymbol{\omega}_c^T\right]^T$, where $\mathbf{v}_c, \boldsymbol{\omega}_c \in \mathbb{R}^3$ are the instantaneous linear and angular velocities of the origin of the camera frame, respectively. As the desired feature's position is static, the features and error kinematics with respect to the camera velocity are

Fig. 1: Quadrotor model. Axis x, y and z represent the vehicle body frame axis. Constant a_l represents the arm length of the motors, corresponding to the distance of the motors to the origin of the body frame.

$$\underbrace{\dot{\mathbf{s}} = \mathbf{L}_s \boldsymbol{\upsilon}_c}_{\textbf{Features kinematics}} \quad \Rightarrow \quad \underbrace{\dot{\tilde{\mathbf{s}}} = \dot{\mathbf{s}} - \dot{\mathbf{s}}^* = \mathbf{L}_s \boldsymbol{\upsilon}_c}_{\textbf{Error Kinematics}}, \tag{2}$$

where the matrix $\mathbf{L}_s \in \mathbb{R}^{k\,x\,6}$ is known as the interaction matrix or feature Jacobian [10], related to $\mathbf{s}$, and of the form

$$\mathbf{L}_s = \begin{bmatrix} -1/Z & 0 & x/Z & xy & -(1+x^2) & y \\ 0 & -1/Z & y/Z & 1+y^2 & -xy & -x \end{bmatrix}, \tag{3}$$

where $\mathbf{s} := (x, y)$ are the feature projection to the normalized image plane, and Z its depth.

By inverting equation (2) and imposing an exponential error decrease profile (*i.e.*, $\dot{\tilde{\mathbf{s}}} = -\eta \tilde{\mathbf{s}}$ and η is a control gain), we obtain the velocity control law

$$\boldsymbol{\upsilon}_c = -\eta \mathbf{L}_s^+ \tilde{\mathbf{s}}, \tag{4}$$

where $\mathbf{L}_s^+ \in \mathbb{R}^{6 \times k}$ is the Moore-Penrose pseudo-inverse of $\mathbf{L}_s$.

IBVS Stability: The stability of the IBVS control law in (4) is dependent on the rank of the interaction matrix, which needs to be full rank to avoid singularities. For this condition to be met, the system needs to track at least 3 features. Tracking 3 features, however, may still lead to some configurations in which $\eta \mathbf{L}_s^+$ is singular. Moreover, with only 3 points, there exist four local minima poses [10] (while only one will be correct, up to four valid solutions can be computed for which $\mathbf{e}$ converges). To avoid this, we track 4 non-collinear features.

B. System Model

The state of the quadrotor is defined by its velocity $\mathbf{v}$ in the inertial frame, its attitude $\boldsymbol{\alpha} := [\theta, \phi, \psi]$ with respect to the inertial frame, where θ, ϕ and ψ are the rotation with respect to the inertial x–axis (roll), y–axis (pitch) and z–axis (yaw), and its angular velocity on the body frame ω, respectively. These states are concatenated under

$$\begin{aligned} \mathbf{x} \in \mathbb{R}^9 &:= \begin{bmatrix} \mathbf{v} & \alpha & \omega \end{bmatrix} \\ &= \begin{bmatrix} v_x & v_y & v_z & \theta & \phi & \psi & \omega_x & \omega_y & \omega_z \end{bmatrix}, \end{aligned} \tag{5}$$

where the subscripts indicate the axis of reference. The quadrotor non-linear dynamics is defined as in [8]:

$$\dot{\mathbf{v}} = \mathbf{R}_B \frac{\boldsymbol{\nu}}{m} + \mathbf{g}, \tag{6a}$$

$$\dot{\boldsymbol{\alpha}} = \mathbf{T}\boldsymbol{\omega}, \text{ and} \tag{6b}$$

$$\dot{\boldsymbol{\omega}} = \mathbf{M}^{-1}(\boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{M}\boldsymbol{\omega}), \tag{6c}$$

where m and $\mathbf{M} \in \mathbb{R}^{3\times3}$ are the mass and inertia matrix of the UAV, $\mathbf{T} \in \mathbb{R}^{3\times3}$ is the body to the inertial frame attitude Jacobian, and $\mathbf{R}_B \in \mathcal{SO}(3)$ the rotation matrix from the body to inertial-frame. In particular, the inertia is a diagonal matrix with components M_x, M_y and M_z for the each axis. Vectors $\boldsymbol{\nu}$ and $\boldsymbol{\tau}$ are respectively the force and torque vectors, where the force vector only has a body z-axis component, due to the geometrical location of the motors in the UAV body – see Fig. 1:

$$\boldsymbol{\nu} = \begin{bmatrix} 0 & 0 & \nu_z \end{bmatrix}. \tag{7}$$

Due to this property, the dynamics in equations (6) are under-actuated.

The set where all states $\mathbf{x}$ lie in is denoted as X. The set of control inputs $\boldsymbol{\nu}$ and $\boldsymbol{\tau}$ is denoted as U, defined as

$$U \triangleq \{-\frac{\nu_z}{4} a_l \leq \tau_{x,y} \leq \frac{\nu_z}{4} a_l, 0 \leq \nu_z \leq 4 \cdot 9.81 \cdot m\}, \tag{8}$$

where a_l is the distance from the center of the UAV to the center of the rotors. As ν_z is the total thrust produced by the 4 rotors, the torque that the vehicle can apply is the force produced by one rotor, multiplied by the lever-arm a_l. Finally, the vehicle cannot produce acceleration more than 4 times its mass.

For IBVS problems, the movement of the camera is assumed to be small. Therefore, the the vehicle will operate close to the hovering equilibrium point, around which we linearize the dynamics in (6), *i.e.*,

$$\mathbf{x}_{eq} = \mathbf{0}, \tag{9}$$

which can be written, in the compact form, as

$$\dot{\mathbf{x}} = \mathbf{f_c}(\mathbf{x}, \mathbf{u}) = \mathbf{A}_c\mathbf{x} + \mathbf{B}_c\mathbf{u}, \tag{10}$$

with

$$\mathbf{A}_c \in \mathbb{R}^{9\times9} = \begin{bmatrix} \mathbf{0}^{3\times3} & [\mathbf{g}]_\times & \mathbf{0}^{3\times3} \\ \mathbf{0}^{3\times3} & \mathbf{0}^{3\times3} & \mathbf{I}^{3\times3} \\ \mathbf{0}^{3\times3} & \mathbf{0}^{3\times3} & \mathbf{0}^{3\times3} \end{bmatrix}, \; [\mathbf{g}]_\times := \begin{bmatrix} 0 & -g & 0 \\ g & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{11}$$

$$\mathbf{B}_c \in \mathbb{R}^{9\times4} = \begin{bmatrix} \mathbf{0}^{3\times3} & \frac{1}{m}\boldsymbol{\epsilon}_z \\ \mathbf{0}^{3\times3} & \mathbf{0}^{3\times1} \\ \mathbf{M}^{-1} & \mathbf{0}^{3\times1} \end{bmatrix}, \tag{12}$$

where

$$\boldsymbol{\epsilon}_z = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{13}$$

and $\mathbf{I} \in \mathbb{R}^{3\times3}$ are the 3×3 identity matrix. The control input vector $\mathbf{u}$ for the linearized system is then composed by

$$\mathbf{u} = \begin{bmatrix} \boldsymbol{\tau}^T & \nu_z \end{bmatrix}^T. \tag{14}$$

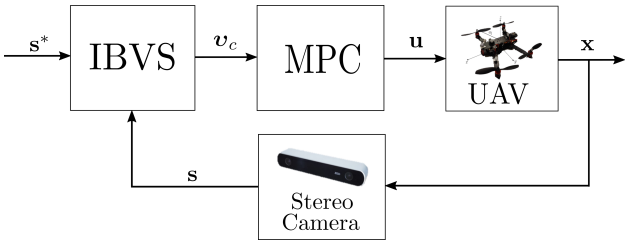


Fig. 2: System architecture: at each camera sampling time, we detect and match the features of interest. IBVS provides a desired velocity in the camera frame, which is transformed to the UAV body. This velocity is then used as a reference to be tracked by the MPC framework, yielding optimal control inputs that are then sent to the hardware-level controller (PX4 [31]). Note that the Stereo Camera is attached the UAV frame.

Finally, we consider a discretization of the continuous model, using a sampling time of h (in the experiments we use $h = 0.01$[s]). In this case, we obtain the linearized discrete-time model

$$\mathbf{x}(t+1|t) = \mathbf{f}(\mathbf{x}(t|t), \mathbf{u}(t|t)) = \mathbf{A}\mathbf{x}(t|t) + \mathbf{B}\mathbf{u}(t|t), \tag{15}$$

where $\mathbf{A}$ and $\mathbf{B}$ are the discrete-time state-space matrices for $\mathbf{A}_c$ and $\mathbf{B}_c$, through

$$\mathbf{A} = e^{\mathbf{A}_c h} = \mathbf{I}^{9\times9} + h \begin{bmatrix} \mathbf{0}^{3\times3} & [\mathbf{g}]_\times & [\mathbf{r}]_\times \\ \mathbf{0}^{3\times3} & \mathbf{0}^{3\times3} & \mathbf{I}^{3\times3} \\ \mathbf{0}^{3\times3} & \mathbf{0}^{3\times3} & \mathbf{0}^{3\times3} \end{bmatrix}, \text{ where}$$

$$\mathbf{r} = \begin{bmatrix} 0 \\ 0 \\ 0.0491 \end{bmatrix}, \text{ and} \tag{16}$$

$$\mathbf{B} = \int_0^h e^{\mathbf{A}_c v}\mathbf{B}_c dv = h \begin{bmatrix} \mathbf{H} & \frac{1}{m}\boldsymbol{\epsilon}_z \\ \frac{1}{2}\mathbf{M}^{-1} & \mathbf{0}^{3\times3} \\ \mathbf{M}^{-1} & \mathbf{0}^{3\times3} \end{bmatrix}, \text{ where}$$

$$\mathbf{H} = \begin{bmatrix} 0 & -\frac{0.016}{M_y} & 0 \\ \frac{0.016}{M_x} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{17}$$

and $\mathbf{x}(t+n|t)$ defines the estimate for time $t+n$ at time t, for $n = 1...N$. For $n = 0$, the estimation is the measurement at time t, that we represent as $\mathbf{x}(t)$.

III. MODEL PREDICTIVE IMAGE-BASED VISUAL SERVOING FOR QUADROTORS

In this section we present the proposed Image-Based Visual Servo control (IBVS) for quadrotors using Model Predictive Control (MPC). To combine both modules, we implemented the cascaded control system, as shown in Fig. 2. We start by modeling the IBVS with MPC. Then, we analyze its stability.

A. Model Predictive Control

The MPC module is responsible to track the desired velocity given by the IBVS module, considering the under-actuated constraints of the quadrotor.

At each step $i = 0...N-1$ where N is the horizon length of the MPC formulation, and from an initial error $\tilde{\mathbf{x}}(t)$, an optimal control input $\mathbf{u}(t|t)$ is obtained by minimizing a cost function $J(\cdot)$, yielding a predicted state $\tilde{\mathbf{x}}(t+1|t)$. Recursively performing these steps, yields a set of optimal predicted states $\{\tilde{\mathbf{x}}(t+1|t), \tilde{\mathbf{x}}(t+2|t), ..., \tilde{\mathbf{x}}(t+N|t)\}$ and $N-1$ optimal control inputs $\mathbf{u}(k+i|k) = \{\mathbf{u}(t|t), \mathbf{u}(t+1|t), ..., \mathbf{u}(t+N-1|t)\}, \forall i = 0...N-1$. Finally, at each sampling time of the MPC controller, the first optimal control input $\mathbf{u}(t|t)$ is selected to be applied to the system.

The desired state for system (6) is

$$\mathbf{x}^* = \begin{bmatrix} \mathbf{R}_B \mathbf{R}_c^B \boldsymbol{v}_c \\ \mathbf{0}^{3\times 1} \\ \boldsymbol{\omega}_B \end{bmatrix}, \tag{18}$$

with

$$\boldsymbol{\omega}_B = \mathbf{R}_c^B \boldsymbol{\omega}_c \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \tag{19}$$

where $\mathbf{v}_c$ and $\boldsymbol{\omega}_c$ are given in (4), and where $\mathbf{R}_c^B$ and $\mathbf{R}_B$ represent the camera to UAV body and UAV body to inertial rotation matrices, respectively.

With this choice of $\mathbf{x}^*$, we close the loop of the IBVS module, and we use MPC to track the received velocity and cope with the system dynamics to obtain an optimal tracking controller. Note that we disregard the angular velocity around the body z-axis - this is due to the selected system linearization state, which considers a fixed heading, as at this point we are interested on translating the UAV to desired visual navigation marks.

B. MPC Feasibility and Stability Analysis

In this subsection we define and analyze the proposed MPC framework. First, let the error state $\tilde{\mathbf{x}}$ be

$$\tilde{\mathbf{x}} \in \mathbb{R}^9 := \mathbf{x} - \mathbf{x}^*. \tag{20}$$

We start by defining the error set E as

$$E \triangleq \left\{\mathbf{x} \in \mathbb{R}^9 : X \oplus \{-\mathbf{x}^*\}\right\}, \tag{21}$$

where X is defined as

$$X \triangleq \{\mathbf{x} \in \mathbb{R}^9 : \theta \in (-\frac{\pi}{9}, \frac{\pi}{9}), \phi \in (-\frac{\pi}{9}, \frac{\pi}{9}, \psi \in (-\frac{\pi}{9}, \frac{\pi}{9})\}, \tag{22}$$

ensuring an operation close to the linearized equilibrium.

The error dynamics are represented by

$$\dot{\tilde{\mathbf{x}}}(\cdot) := \dot{\mathbf{x}}(\cdot) - \underbrace{\dot{\mathbf{x}}^*(\cdot)}_{=\mathbf{0}} \quad \Rightarrow \quad \dot{\tilde{\mathbf{x}}}(\cdot) = \dot{\mathbf{x}}(\cdot). \tag{23}$$

Remark 1. *We consider the velocity to be tracked by the MPC as constant through the whole receding horizon, as we sample the camera information only at the initial sampling time $\tilde{\mathbf{x}}(k)$, which renders $\dot{\mathbf{x}}^*(t+n|t) = \mathbf{0}, \forall n = 0, ..., N$.*

Accordingly, we write the cost functions for the running cost $J_l(\cdot)$ and final cost $J_f(\cdot)$ to be used in the MPC formulation as

$$\begin{aligned} J_l(\tilde{\mathbf{x}}(t+k|t), \mathbf{u}(t+k|t)) = \\ \tilde{\mathbf{x}}(t+k|t)^T \mathbf{Q}_x \tilde{\mathbf{x}}(t+k|t) + \\ \mathbf{u}(t+k|t)^T \mathbf{Q}_u \mathbf{u}(t+k|t), \end{aligned} \tag{24}$$

$$J_f(\tilde{\mathbf{x}}(t+N|t)) = \tilde{\mathbf{x}}(t+N|t)^T \mathbf{Q}_f \tilde{\mathbf{x}}(t+N|t). \tag{25}$$

where $\mathbf{Q}_x$ and $\mathbf{Q}_u$ are positive definite diagonal matrices that weight the state error and the control effort, respectively. These weights can be appropriately tuned to achieve a desired performance. Finally, the matrix $\mathbf{Q}_f$ is the final weight matrix for the last step of the receding horizon optimization, obtained through the Riccatti equation:

$$\begin{aligned} \mathbf{Q}_f = &\mathbf{Q}_x + \mathbf{A}^T \mathbf{Q}_f \mathbf{A} - \\ &(\mathbf{B}^T \mathbf{Q}_f \mathbf{A})^T (\mathbf{Q}_u + \mathbf{B}^T \mathbf{Q}_f \mathbf{B})^{-1} (\mathbf{B}^T \mathbf{Q}_f \mathbf{A}). \end{aligned} \tag{26}$$

The final cost is defined through the Riccatti equation as it provides the best estimate of the cost-to-go for a receding horizon controller [11], similarly to the infinite horizon cost for Linear Quadratic Regulators [32].

The cost function, referred to as $J(\cdot)$, is defined as

$$\begin{aligned} J(\tilde{\mathbf{x}}(t|t), \mathbf{u}(t|t)) = \\ \sum_{k=0}^{N-1} J_l(\tilde{\mathbf{x}}(t+k|t), \mathbf{u}(t+k|t)) + J_f(\tilde{\mathbf{x}}(t+N|t)). \end{aligned} \tag{27}$$

The optimal control input is then given by the following optimization problem

$$\begin{aligned} \underset{\mathbf{u}(k+i|k)}{\text{argmin}} \quad & J(\tilde{\mathbf{x}}(t|t), \mathbf{u}(t|t)) && \text{(28a)} \\ \text{subject to} \quad & \tilde{\mathbf{x}}(t+k+1|t) = \\ & \qquad \mathbf{A}\tilde{\mathbf{x}}(t+k|t) + \mathbf{B}\mathbf{u}(t+k|t), && \text{(28b)} \\ & \tilde{\mathbf{x}}(t+k|t) \in E, && \text{(28c)} \\ & \tilde{\mathbf{x}}(t+N|t) \in E_f, && \text{(28d)} \\ & \mathbf{u}(t+k|t) \in U && \text{(28e)} \end{aligned}$$

where E and U are given in (21), and (8), respectively. E_f is defined as

$$E_f \triangleq \{\tilde{\mathbf{x}} \in \mathbb{R}^9 : \tilde{\mathbf{x}} = \mathbf{0}^9\} \subset E, \tag{29}$$

representing the desired state final state.

We now state the theorem that yields the stability results of this work.

Theorem 1. *Under the control law in (28), then $\tilde{\mathbf{x}}(t)$ satisfies $\lim_{t\to\infty} \tilde{\mathbf{x}}(t) = 0$ from all initial values of $\tilde{\mathbf{x}}$ for which (29) admits a feasible solution.*

Proof. The proof of this theorem follows the corresponding analysis in [11] and is divided in three parts: at first, we prove that the terminal set E_f is a control invariant set; at the second, we demonstrate the boundedness of $J_N^0(\tilde{\mathbf{x}})$; and finally we guarantee that $J_N^0(\tilde{\mathbf{x}})$ is monotonically decreasing.

Following [11], let $J_N^0(\tilde{\mathbf{x}}) = J(\tilde{\mathbf{x}}(0|t), \mathbf{u}_N(\cdot|t))$, where $\tilde{\mathbf{x}}(0|t) \in E$ is any MPC starting state and $\mathbf{u}_N(\cdot|t) =$

$\{\mathbf{u}_N(0|t), \mathbf{u}_N(1|t), \ldots, \mathbf{u}_N(N-1|t)\}$ be a minimizing control sequence.

1) Control Invariant Sets: The controllability matrix for system (10) is of the form

$$\mathbf{C}_m = \begin{bmatrix}\mathbf{A}^n\mathbf{B} & \mathbf{A}^{n-1}\mathbf{B} & \ldots & \mathbf{A}\mathbf{B}\end{bmatrix} \in \mathbb{R}^{9\times 36}, \quad (30)$$

where $\mathbf{A}$ and $\mathbf{B}$ are defined in (16) and (17).

In our case, the rank of $\mathbf{C}_m$ is 9, and the condition number of $\mathbf{C}_m$ is approximately 388, for our system properties defined in Section IV. Therefore, we conclude that the system (10) is controllable, and therefore stabilizable. As so, there exists a terminal feedback controller $\mathbf{u}(t+N|t) = -\mathbf{\Gamma}\tilde{\mathbf{x}}(t+N|t)$ that renders E control invariant. Noting that $\{\tilde{\mathbf{x}}_{eq}\} = \{\mathbf{0}\} \in E_f \subset E$, E_f is consequently a control invariant set.

Remark 2. *The condition number of $\mathbf{C}_m$ is dependent on the system inertial parameters that affect the control matrix $\mathbf{B}$. Therefore, it is worth noting that these results are valid for the particular system in hand.*

2) Boundedness of $J_N^0(\tilde{\mathbf{x}})$: The bounds of $J_N^0(\tilde{\mathbf{x}})$ are derived from the choice of the running cost function, given by $\tilde{\mathbf{x}}(t+k|t)^T\mathbf{Q}_x\tilde{\mathbf{x}}(t+k|t) + \mathbf{u}(t+k|t)^T\mathbf{Q}_u\mathbf{u}(t+k|t)$, with $\mathbf{Q}_x$ and $\mathbf{Q}_u$ positive definite. Therefore, it holds that

$$\alpha_1(||\tilde{\mathbf{x}}||, ||\mathbf{u}||) \leq J_N^0(\tilde{\mathbf{x}}) \leq \alpha_2(||\tilde{\mathbf{x}}||, ||\mathbf{u}||) \quad (31)$$

where $\alpha_1(||\tilde{\mathbf{x}}||, ||\mathbf{u}||) = \underline{\lambda}(\mathbf{Q}_x)||\tilde{\mathbf{x}}|| + \underline{\lambda}(\mathbf{Q}_u)||\mathbf{u}||$ and $\alpha_2(||\tilde{\mathbf{x}}||, ||\mathbf{u}||) = \bar{\lambda}(\mathbf{Q}_x)||\tilde{\mathbf{x}}|| + \bar{\lambda}(\mathbf{Q}_u)||\mathbf{u}||$, and where $\underline{\lambda}(\mathbf{Q_i})$ and $\bar{\lambda}(\mathbf{Q_i})$ are, respectively, the minimum and maximum eigenvalues of matrix $\mathbf{Q_i}$. Noting that $\mathbf{Q}_x$ and $\mathbf{Q}_u$ are diagonal positive-definite matrices, then the eigenvalues are displayed on the diagonal of these matrices and correspond to the appropriate weights chosen for the cost function. For the experiments we chose $\lambda_{max}(\mathbf{Q}_x) = 100, \lambda_{min}(\mathbf{Q}_x) = 1$ and $\lambda_{max}(\mathbf{Q}_u) = 2, \lambda_{min}(\mathbf{Q}_x = 1)$, leading to

$$||\tilde{\mathbf{x}}|| + ||\mathbf{u}|| \leq J_N^0(\tilde{\mathbf{x}}) \leq 100||\tilde{\mathbf{x}}|| + 2||\mathbf{u}|| \quad (32)$$

3) Descent and Monotonicity Properties of the Cost Function: The remaining proof follows the same steps as in [11], Proposition 2.12, proof for Theorem 2.24.

4) Recursive Feasibility: To prove recursive feasibility, we recall that E is a control invariant set, that is, $\forall\tilde{\mathbf{x}} \in E$, there exists at least one $\mathbf{u} \in U$ such that $\mathbf{f}(\tilde{\mathbf{x}}, \mathbf{u}) \in E$. Note also that $\tilde{\mathbf{x}}(t) = \tilde{\mathbf{x}}(t|t)$.

The control law in (28) gives an optimal control signal for the system in (20), yielding $\tilde{\mathbf{x}}(t+n+1|t) = \mathbf{f}(\tilde{\mathbf{x}}(t+n|t), \mathbf{u}(t+n|t))$. fThe control inputs through the receding horizon, then, satisfy the constraints set by E and U if and only if the problem is initially feasible.

At step $n = N$, $\mathbf{f}(\tilde{\mathbf{x}}(t+N|t), \mathbf{u}_f(t+N|t)) = \tilde{\mathbf{x}}(t+N+1|t) \in E_f$, as there exists at least one control input $\mathbf{u}_f(t+N|t)$, that renders the set E_f control invariant, *i.e.*, $\tilde{\mathbf{x}}(t+N+1|t) \in E_f$ for at least one $\mathbf{u}_f(t+N|t)$ control input. Such control input can be a linear feedback controller $\mathbf{u}_f(t+N|t) = -\mathbf{\Gamma}\tilde{\mathbf{x}}(t+N|t)$, and $\mathbf{\Gamma}$ is an appropriately selected gain matrix. As so, if the optimization problem in (28) is feasible at (t), then it will be recursively feasible for $(t+n|t)$, $\forall n = 0 \ldots N$.

Finally, it can be concluded that the system in (20), under the control law in (28), converges to the origin, that is, the the error $\tilde{\mathbf{x}}(t)$ asymptotically converges to zero, and therefore the system achieves the desired tracking of the reference velocity from the IBVS module. □

C. Algorithm

In a succinct form, at each sampling time, we: i) sample four features $\mathbf{s}$; ii) solve (1) to obtain $\tilde{\mathbf{s}}$, followed by solving (4) to get $\boldsymbol{\upsilon}_c$; iii) obtain $\tilde{\mathbf{x}}$ from (18) and (20), and iv) solve the optimization problem in (28), yielding the control input $\mathbf{u}$, which is applied to the system.

IV. Experiments

We run two different type of experiments. We start with realistic simulation results using Gazebo and the RotorS simulator [33] (experiments presented in Sec. IV-A). No disturbances are considered and a good approximation of the system model is used. Then, in Sec. IV-B, we tested the proposed framework in a real experimental scenario using a UAV with very little knowledge of the system's model.

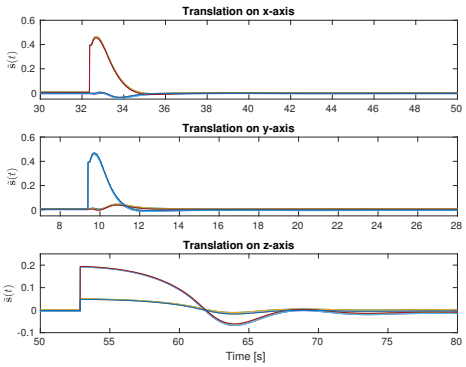
For both simulations and real experiments, we perform the same excitation, with desired features setpoints, to the system: sequential translations of $0.5[m]$ on x, y and z axis, achieved by changing the desired image features $\mathbf{s}^*$ accordingly. We have generated `C/C++` code using CVXGEN [34], with an horizon of $N = 10$, for the Linear MPC implementation of (28), which was wrapped around the ROS framework [35]. The code will be made available. Implementation-wise, both simulations and real experiments share the same ROS backend, apart from the vehicle interface. In simulation, we rely on RotorS, while for the real experiments we use a PX4-based UAV [31] and the MAVROS package[1] to interface with the flight controller. packages available.

A. Simulation Results

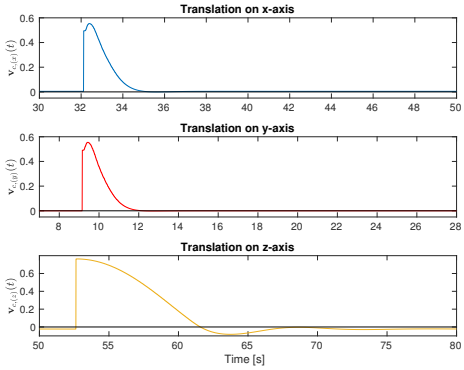
Figure 3 shows the results obtained in simulation. Figure 3(a) shows the feature error and 3(b) the IBVS velocity convergence. On the 3D setup, the UAV starts at an approximate distance to the target of $1.3[m]$, composed by 4 features. The simulated vehicle has a mass of $1.5[kg]$ and inertia of $M_x = 0.034, M_y = 0.046$ and $M_z = 0.098$.

As expected, with a good knowledge of the system parameters, our control method achieves a practically zero steady-state error, converging asymptotically to the desired reference with a small overshoot on the z axis. This result is important in assessing the proposed scheme, as it means that the connection of the IBVS module and the MPC works properly. The MPC solver took a minimum of $0.72[ms]$, a maximum of $5.8[ms]$, and an average of $1.1[ms]$ to obtain a solution on a $3.6[GHz]$ Intel Core-i7 CPU.

[1]Available at (on 12th September, 2019): https://github.com/mavlink/mavros

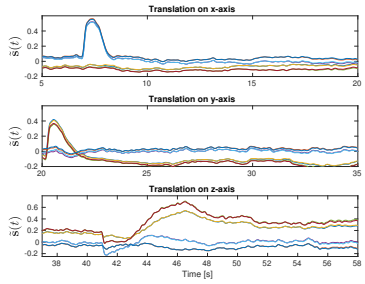


(a) Feature Error. Each curve represents a feature component error (x and y in the image plane).

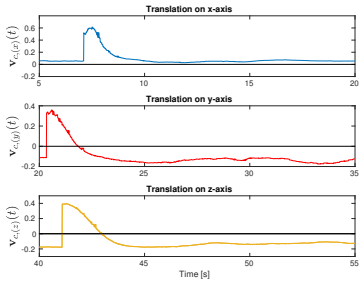


(b) IBVS Velocity Module. The curves in blue, red and yellow represent the desired velocity along the x, y and z axis.

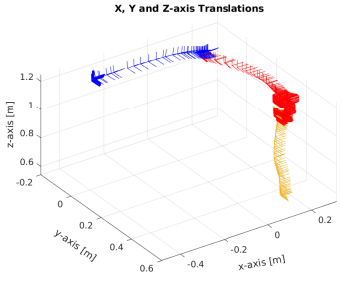
Fig. 3: Simulation results: We use a realistic physics engine simulator, namely Gazebo, and RotorS [33]. We observe that the feature errors converge to zero and, accordingly, so does the desired velocity for the camera/UAV. Note that convergence on the x and y axis is smaller (and similar among each other) than the convergence on the z axis, due to the system dynamics. These can be, also, tuned by adjusting weight matrices $\mathbf{Q}_x, \mathbf{Q}_u$ and $\mathbf{Q}_f$. On the left, each curve corresponds to a feature error. On the right, zero error is shown in black, while the velocity for x, y and z is shown in blue, red and yellow.



(a) Feature Error for x, y and z translations, top to bottom. Each curve represents a feature component error (x and y in the image plane).



(b) IBVS Velocity Module. The curves in blue, red and yellow represent the desired velocity along the x, y and z axis, respectively.



(c) 3D Trajectory of the UAV. Blue, red and yellow represent the translations on x, y and z respectively.

Fig. 4: Performance of the real system: We observe that, although the system converges to the correct desired feature position in the image plane, there is a small steady-state error. This error is due to the imperfect knowledge of the system inertial properties. This is observed on the error around z-axis, as a force-to-thrust mapping error causes a small decrease in the system performance. Nonetheless, we observe that the vehicle navigates stably through the desired feature sets, while maintaining small roll and pitch angles, as well as zero yaw, respecting the linearization limits.

B. Experimental Results

On what concerns the experimental test-bed, our UAV, built around a Hover 1 frame, a Nvidia Jetson TX2, a PX4 flight controller and, has an approximate mass of $1.73[kg]$ and a rough inertia approximation of $M_x = 0.04, M_y = 0.04$, and $M_z = 0.1$. To obtain the depth Z of each feature, we use a ZED Stereo camera. The system successfully converges to the desired setpoints, although a small steady-state error is present. The obtained 3D trajectory, as shown in Figure 4(c), provides an insight on the system's performance while navigating through the different setpoints. The accompanying video shows the UAV navigating through these setpoints. On the vehicle, running a Nvidia Jetson TX2, the solver took a minimum of $7.9[ms]$, a maximum of $10.8[ms]$ and an average of $8.4[ms]$ to obtain a sequence of control inputs, making it possible to run the controller at $100[Hz]$.

We observe a small decrease in the tracking performance on the real system. This results from (i) an imperfect knowledge of our system inertial parameters, and (ii) an inaccurate force-to-thrust mapping, dependent on motor properties and propellers performance. In typical PID implementations, the integrator gain compensates for these system errors, converging the steady-state error to zero. Nonetheless, a PID approach is subject to high gain tuning and input saturation that can render the system unstable, and it cannot deal with state constraints. At this stage, we did not add an integration action, as it is not relevant to evaluate our formulation.

The system coped with the state constraints that avoid states far from the linearization point, and kept the desired small attitude angles during the entire test sets. Moreover, the

computational benefits with respect to previous approaches of using MPC for quadrotors, e.g. [28], make this approach easily implementable on-board UAVs of different sizes and computational capabilities, ensuring an operation near the linearization point of the system.

Furthermore, the use of the MPC framework was crucial to limit the velocity of the vehicle and cope with motion blur that the used camera is subject to.

V. CONCLUSIONS

We present a new formulation to tackle the under-actuation issues of the image-based visual servoing of a quadrotor. The guarantees on the feasibility and stability of the MPC are presented. Using both synthetic and real data, we show our method works properly and is capable of running at a frequency of 100Hz on-board the vehicle. As future work, we plan to extend the formulation to a Linear Time-Varying (LTV) MPC to handle time-varying dynamics, and incorporate the visual servoing dynamics in the MPC framework.

REFERENCES

[1] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics Automation Magazine (RA-M)*, vol. 19, no. 3, pp. 46–56, 2012. 1

[2] Z. Li, Y. Liu, R. Walker, R. Hayward, and J. Zhang, "Towards automatic power line detection for a uav surveillance system using pulse coupled neural filter and an improved hough transform," *Machine Vision and Applications*, vol. 21, no. 5, pp. 677–686, 2010. 1

[3] M. Burri, J. Nikolic, C. Hrzeler, G. Caprari, and R. Siegwart, "Aerial service robots for visual inspection of thermal power plant boiler systems," in *Int'l Conf. Applied Robotics for the Power Industry (CARPI)*, 2012, pp. 70–75. 1

[4] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, L. Meier, P. Tanskanen, and M. Pollefeys, "Vision-based autonomous mapping and exploration using a quadrotor mav," in *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 2012, pp. 4557–4564. 1

[5] G. Loianno, G. Cross, C. Qu, Y. Mulgaonkar, J. A. Hesch, and V. Kumar, "Flying smartphones: Automated flight enabled by consumer electronics," *IEEE Robotics Automation Magazine (RA-M)*, vol. 22, no. 2, pp. 24–32, 2015. 1

[6] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," *IEEE Trans. Robotics (T-RO)*, vol. 32, no. 6, pp. 1498–1511, 2016. 1

[7] J. Willmann, F. Augugliaro, T. Cadalbert, R. D'Andrea, F. Gramazio, and M. Kohler, "Aerial robotic construction towards a new field of architectural research," *Int'l Journal of Architectural Computing*, vol. 10, no. 3, pp. 439–459, 2012. 1

[8] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine (RA-M)*, vol. 19, no. 3, pp. 20–32, 2012. 1, 3

[9] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Robotics Automation Magazine (RA-M)*, vol. 12, no. 5, pp. 651–670, 1996. 1

[10] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics Automation Magazine (RA-M)*, vol. 13, no. 4, pp. 82–90, 2006. 1, 2

[11] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, WI, 2017, vol. 2. 1, 4, 5

[12] 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 Transactions on Robotics*, vol. 25, no. 3, pp. 743–749, 2009. 1

[13] H. Sheng, E. Shi, and K. Zhang, "Image-based visual servoing of a quadrotor with improved visibility using model predictive control," in *IEEE Int'l Symposium on Industrial Electronics (ISIE)*, 2019, pp. 551–556. 1

[14] A. Mcfadyen, L. Mejias, P. Corke, and C. Pradalier, "Aircraft collision avoidance using spherical visual predictive control and single point features," in *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 2013, pp. 50–56. 1

[15] A. Mcfadyen, P. Corke, and L. Mejias, "Visual predictive control of spiral motion," *IEEE Trans. Robotics (T-RO)*, vol. 30, no. 6, pp. 1441–1454, 2014. 1

[16] T. Templeton, D. H. Shim, C. Geyer, and S. S. Sastry, "Autonomous vision-based landing and terrain mapping using an mpc-controlled unmanned rotorcraft," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2007, pp. 1349–1356. 1

[17] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2014, pp. 2113–2118. 1

[18] J. Thomas, G. Loianno, K. Daniilidis, and V. Kumar, "Visual servoing of quadrotors for perching by hanging from cylindrical objects," *IEEE Robotis and Automation Letters (RA-L)*, vol. 1, no. 1, pp. 57–64, 2015. 1

[19] G. Loianno, C. Brunner, G. McGrath, and V. Kumar, "Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu," *IEEE Robotis and Automation Letters (RA-L)*, vol. 2, no. 2, pp. 404–411, 2016. 1

[20] B. Penin, P. R. Giordano, and F. Chaumette, "Minimum-time trajectory planning under intermittent measurements," *IEEE Robotis and Automation Letters (RA-L)*, vol. 4, no. 1, pp. 153–160, 2018. 1

[21] ——, "Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions," *IEEE Robotis and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3725–3732, 2018. 1

[22] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, "Vision-based minimum-time trajectory generation for a quadrotor uav," in *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, 2017, pp. 6199–6206. 1

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

[24] M. Sauvee, P. Poignet, E. Dombre, and E. Courtial, "Image based visual servoing through nonlinear model predictive control," in *IEEE Conf. Decision and Control (CDC)*, 2006, pp. 1776–1781. 2

[25] T. Murao, T. Yamada, and M. Fujita, "Predictive visual feedback control with eye-in-hand system via stabilizing receding horizon approach," in *IEEE Conf. Decision and Control (CDC)*, 2006, pp. 1758–1763. 2

[26] G. Allibert, E. Courtial, and Y. Toure, "Visual predictive control for manipulators with catadioptric camera," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2008, pp. 510–515. 2

[27] A. McFadyen, M. Jabeur, and P. Corke, "Image-based visual servoing with unknown point feature correspondence," *IEEE Robotis and Automation Letters (RA-L)*, vol. 2, no. 2, pp. 601–607, 2017. 2

[28] D. Lee, H. Lim, and H. J. Kim, "Obstacle avoidance using image-based visual servoing integrated with nonlinear model predictive control," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 5689–5694. 2, 7

[29] S. S. Skiena, *The algorithm design manual: Text*. Springer Science & Business Media, 1998, vol. 1. 2

[30] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, 1st ed., ser. 26. Springer-Verlag New York, 2004. 2

[31] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 6235–6240. 3, 5

[32] A. Bemporad, M. Morari, V. Dua, and E. N. Pistikopoulos, "The explicit linear quadratic regulator for constrained systems," *Automatica*, vol. 38, no. 1, pp. 3–20, 2002. 4

[33] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Cham: Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-26054-9_23 5, 6

[34] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 12, no. 1, pp. 1–27, 2012. 5

[35] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009. 5