Towards Dynamic Visual Servoing for Interaction Control and Moving Targets

Alexander Oliva, Erwin Aertbeliën, Joris de Schutter, Paolo Robuffo Giordano, François Chaumette

Towards Dynamic Visual Servoing for Interaction Control and Moving Targets

Alexander Antonio Oliva[1], Erwin Aertbeliën[2,3], Joris De Schutter[2,3], Paolo Robuffo Giordano[4] and François Chaumette[1]

***Abstract*— In this work we present our results on dynamic visual servoing for the case of moving targets while also exploring the possibility of using such a controller for interaction with the environment. We illustrate the derivation of a feature space impedance controller for tracking a moving object as well as an Extended Kalman Filter based on the visual servoing kinematics for increasing the rate of the visual information and estimating the target velocity for both the cases of PBVS and IBVS with image point features. Simulations are carried out to validate the estimator performance during a Peg-in-Hole insertion task with a moving part. Experiments are also conducted on a real redundant manipulator with a low-cost wrist-mounted camera. Details on several implementation issues encountered during implementation are also discussed.**

I. INTRODUCTION

Visual Servoing is a consolidated control technique that allows to precisely position a vision sensor with respect to an observed object in the scene. In the past decades, most research efforts focused on the modeling of the existing relationships between descriptive features in the image and the motion of the sensor at kinematic level [1], [2], and in the development of control strategies to guide the chosen visual features towards the desired ones as summarized in [3], [4]. Although much research has been conducted in this sense, little attention has been devoted to the use of second order models linking the features accelerations to the robot torques (via the robot dynamical model), with some early attempts dating back more than two decades [1], [5] up to some more recent contributions [6], [7]. As it is well known, explicitly taking into account the robot dynamics allows to design controllers with superior performance, especially for what concerns the regulation of forces and interaction with the environment. For instance, in many industrial applications the interaction tasks are executed in static or quasi-static conditions and can be solved at kinematic level. However if one wishes to reduce the execution time, faster motions are required making the control task essentially dynamic [8].

Most of the cited works mostly focused on the motion of the robot in free space without taking into account possible physical interactions with the environment. Indeed while [5] only compensates for the air drag forces acting on the blimp, in [6], dynamic visual-servoing (DVS) is used to control solely the unconstrained directions of an engraving task within a hybrid control scheme [9] while a pure force controller acts on the constrained directions. The goal of this paper is to study the possibility to use DVS for interaction control, since, as we have shown in previous work [10], a DVS controller under interaction with the environment is essentially an impedance controller in feature space, which is an indirect force control method [11]. In [10] we proposed an alternative to the impedance control in feature space - which we could not implement given the limitations discussed (and addressed) in this paper - by defining an admittance in feature space. The main limitation of this solution is however that, in addition to requiring an external sensor to measure the interaction forces, all collisions occurring upstream of the sensor cannot be detected and the manipulator cannot counteract for accommodating these forces.

A contribution of this work is, therefore, to overcome these difficulties and propose a true DVS in feature space that can also take into account the possible (unknown) motion of the observed target object. This is achieved by modeling the motion of the target in a DVS setting and by estimating its unknown motion via a suitable Extended Kalman Filter (EKF). The EKF is also instrumental for virtually increasing the data rate of the employed camera by providing high-rate estimations of the quantities needed by the employed torque-level controller. The derivations are specialized for the cases of Image-Based Visual Servoing (IBVS) and Position-Based Visual Servoing (PBVS) by pointing out the respective similarities and differences.

II. PRELIMINARIES

Let us consider the fixed world frame of reference, named Σ_w and with the z-axis pointing upwards, to be coincident with the robot base frame Σ_b. The manipulator consists of a succession of links and actuated joints whose configuration is given by the joint vector $\boldsymbol{q} \in \mathbb{R}^n$. The end-effector frame is denoted with Σ_e. A camera (frame Σ_c) in an *eye-in-hand* configuration is mounted on the robot wrist. Finally, an object (frame Σ_o) is visually tracked via the camera. For IBVS, the coordinates of image point features are directly extracted from the image and the associated depths are assumed provided by another pose reconstruction module. For PBVS, the object's pose in frame Σ_c is instead reconstructed from the visual input. Any required geometrical transformation between the above-defined reference frames is supposed to be known, except for those concerning the object frame.

The dynamic model of a robotic arm can be written using the *Lagrange* formulation in the joint space as [11]

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{\tau_e} = \boldsymbol{\tau} \tag{1}$$

This work was supported by the Région Bretagne, France.

[1]Inria, Univ Rennes, CNRS, IRISA, Campus de Beaulieu, 35042 Rennes, France {alexander.oliva, francois.chaumette}@inria.fr

[2]KU Leuven, Dept. of Mechanical Engineering, Leuven, Belgium, {joris.deschutter, erwin.aertbelien}@kuleuven.be

[3]Flanders Make, Core Lab ROB, Leuven, Belgium

[4]CNRS at IRISA and Inria, Rennes, France, prg@irisa.fr

where $\dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}} \in \mathbb{R}^n$ are respectively the generalized joint velocities and accelerations, $\boldsymbol{B}(\boldsymbol{q}) \in \mathbb{R}^{n\times n}$ is the symmetric and positive definite inertia matrix, $\boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \in \mathbb{R}^n$ is the vector of centrifugal and Coriolis effects, $\boldsymbol{g}(\boldsymbol{q}) \in \mathbb{R}^n$ is the configuration dependent vector of gravitational forces, $\boldsymbol{\tau_e} = {}^{\boldsymbol{e}}\boldsymbol{J_e}^\top {}^{\boldsymbol{e}}\boldsymbol{h_e} \in \mathbb{R}^n$ is the joint torque vector corresponding to the external *wrench* ${}^{\boldsymbol{e}}\boldsymbol{h_e} \in \mathbb{R}^6$ acting on the end-effector frame and ${}^{\boldsymbol{e}}\boldsymbol{J_e} \in \mathbb{R}^{6\times n}$ is the robot *Jacobian*, being both expressed in end-effector frame. Finally, $\boldsymbol{\tau} \in \mathbb{R}^n$ is the vector of joint actuation torques. For the sake of notation simplicity, we omit the time dependencies in the treatment when possible.

Let us now consider a vector $\boldsymbol{s} \in \mathbb{R}^k$ of k "features" which in the IBVS case can be, e.g., the coordinates of the points in the image plane and in the PBVS case the position and orientation of the observed object. The *kinematic* relationship between the motion of the visual features and the relative *twist* ${}^{\boldsymbol{c}}\mathbf{v} \in \mathbb{R}^6$, which is the difference between camera (${}^{\boldsymbol{c}}\mathbf{v_c}$) and object (${}^{\boldsymbol{c}}\mathbf{v_o}$) twists in the camera frame, is given by the following differential relation

$$\dot{\boldsymbol{s}} = \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbf{v} = \boldsymbol{L_s}({}^{\boldsymbol{c}}\mathbf{v_c} - {}^{\boldsymbol{c}}\mathbf{v_o}) = \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbf{v_c} - \dot{\boldsymbol{s}}_{\boldsymbol{o}} \tag{2}$$

being $\boldsymbol{L_s} \in \mathbb{R}^{k\times 6}$ the well-known interaction matrix [1] (in this work we will consider that it has full rank, thus constraining the 6 degrees of freedom (DoF) of the camera motion). The second term in the right-hand side represents the contribution to the feature velocity due to the (assumed unknown) target motion $\dot{\boldsymbol{s}}_{\boldsymbol{o}} = \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbf{v_o}$.

III. VISUAL-IMPEDANCE CONTROLLER IN FEATURE SPACE WITH MOVING TARGET

In the following we recall the derivation of the visual-servo (VS) dynamics and how the motion of visual features relates to the robot joint motion at acceleration level. This relationship will then be used, together with the VS kinematics, to express the dynamic model of the manipulator in feature space.

A classical VS task consists in regulating some visual features $\boldsymbol{s}$ to reach a desired value $\boldsymbol{s}^*$. In a previous work [10], we have detailed the derivation of an impedance controller in feature space for a static target. In this section we summarize the various steps of [10] but by considering the more general case of tracking a moving target with a non-negligible velocity. For the case of an eye-in-hand system, Eq. (2) can be rewritten to relate the velocity of the features with the velocities of the robot joints through its *Jacobian* as

$$\dot{\boldsymbol{s}} = \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\boldsymbol{J_e}\dot{\boldsymbol{q}} - \dot{\boldsymbol{s}}_{\boldsymbol{o}} \tag{3}$$

being ${}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}} \in \mathbb{R}^{6\times 6}$ the twist-transformation matrix that transforms the camera *twist* expressed in the end-effector frame into the camera frame. By time differentiation one obtains the expression of the features acceleration

$$\begin{aligned}\ddot{\boldsymbol{s}} =& \dot{\boldsymbol{L}}_{\boldsymbol{s}}\,{}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\boldsymbol{J_e}\dot{\boldsymbol{q}} + \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\dot{\mathbb{T}}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\boldsymbol{J_e}\dot{\boldsymbol{q}} + \\ & \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\dot{\boldsymbol{J}}_{\boldsymbol{e}}\dot{\boldsymbol{q}} + \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\boldsymbol{J_e}\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{s}}_{\boldsymbol{o}}\end{aligned} \tag{4}$$

in which a term relative to the target motion appears.

We can rewrite (4) in a more compact form as

$$\ddot{\boldsymbol{s}} = \boldsymbol{J_s}\ddot{\boldsymbol{q}} + \boldsymbol{h_q} - \ddot{\boldsymbol{s}}_{\boldsymbol{o}} \tag{5}$$

where $\boldsymbol{J_s} = \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\boldsymbol{J_e}$ denotes the so called *Feature Jacobian* [12], [7] and

$$\boldsymbol{h_q} = (\dot{\boldsymbol{L}}_{\boldsymbol{s}}\,{}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\boldsymbol{J_e} + \boldsymbol{L_s}\,{}^{\boldsymbol{c}}\mathbb{T}_{\boldsymbol{e}}\,{}^{\boldsymbol{e}}\dot{\boldsymbol{J}}_{\boldsymbol{e}})\dot{\boldsymbol{q}}.$$

Note that, being the transformation between the camera and the end-effector constant, for a mounted camera one has ${}^{\boldsymbol{c}}\dot{\mathbb{T}}_{\boldsymbol{e}} = 0$.

Now, rearranging (1) as

$$\ddot{\boldsymbol{q}} = \boldsymbol{B}(\boldsymbol{q})^{-1}(\boldsymbol{\tau} - \boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - {}^{\boldsymbol{e}}\boldsymbol{J}_{\boldsymbol{e}}^\top {}^{\boldsymbol{e}}\boldsymbol{h_e}) \tag{6}$$

in which we regroup in $\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{g}(\boldsymbol{q})$, and after replacing (6) into (5), we find the dynamic equation governing the features motion

$$\begin{aligned}\boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{\tau} = \ddot{\boldsymbol{s}} + \ddot{\boldsymbol{s}}_{\boldsymbol{o}} - \boldsymbol{h_q} + \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ + \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}{}^{\boldsymbol{e}}\boldsymbol{J}_{\boldsymbol{e}}^\top {}^{\boldsymbol{e}}\boldsymbol{h_e}.\end{aligned} \tag{7}$$

By renaming some terms in (7) as

$$\begin{aligned}\boldsymbol{f_s} &= \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{\tau} \\ \boldsymbol{b_s} &= \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \boldsymbol{f}_{\boldsymbol{s_{ext}}} &= \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}{}^{\boldsymbol{e}}\boldsymbol{J}_{\boldsymbol{e}}^\top {}^{\boldsymbol{e}}\boldsymbol{h_e}\end{aligned}$$

one obtains the manipulator dynamic model in feature space:

$$\ddot{\boldsymbol{s}} + \ddot{\boldsymbol{s}}_{\boldsymbol{o}} - \boldsymbol{h_q} + \boldsymbol{b_s} + \boldsymbol{f}_{\boldsymbol{s_{ext}}} = \boldsymbol{f_s}. \tag{8}$$

As can be seen from the previous equation, the system is not endowed of inertia (it behaves as a mechanical system with unitary mass/inertia). At this stage we suppose to not have any force measurement, but an estimate of the feature velocity and acceleration due to the target motion ($\dot{\boldsymbol{s}}_{\boldsymbol{o}}, \ddot{\boldsymbol{s}}_{\boldsymbol{o}}$) is supposed to be available (Sect. IV will detail how this estimation is obtained). We choose a feature space control input $\boldsymbol{u}$ able to compensate for any of the dynamic terms in (8)

$$\boldsymbol{u} = \boldsymbol{\alpha} - \boldsymbol{h_q} + \ddot{\boldsymbol{s}}_{\boldsymbol{o}} + \boldsymbol{b_s}$$

which re-injected into (8) results in

$$\boldsymbol{\alpha} = \ddot{\boldsymbol{s}} + \boldsymbol{f}_{\boldsymbol{s_{ext}}} \tag{9}$$

in which $\boldsymbol{\alpha} \in \mathbb{R}^k$ represents a resolved acceleration in feature space and is the new control input variable. A common choice for $\boldsymbol{\alpha}$ is a PD controller with acceleration *feed-forward*:

$$\boldsymbol{\alpha} = \ddot{\boldsymbol{s}}^* + \boldsymbol{D_s}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K_s}\boldsymbol{e_s}. \tag{10}$$

where $\boldsymbol{D_s}$ and $\boldsymbol{K_s}$ are positive definite $k \times k$ matrices representing the relative per unit of mass inertia (*p.u.m.i.*) virtual damping and stiffness of the impedance controller. The resulting joint space controller, recalling the error definition $\boldsymbol{e_s} = \boldsymbol{s}^* - \boldsymbol{s}$, $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = \dot{\boldsymbol{s}}^* - \dot{\boldsymbol{s}}$ and Eq. (3), is then

$$\begin{aligned}\boldsymbol{u_\tau} = (\boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1})^\dagger \Big(\ddot{\boldsymbol{s}}^* + \ddot{\boldsymbol{s}}_{\boldsymbol{o}} - \boldsymbol{h_q} + \boldsymbol{K_s}(\boldsymbol{s}^* - \boldsymbol{s}) \\ + \boldsymbol{D_s}(\dot{\boldsymbol{s}}^* - \boldsymbol{J_s}\dot{\boldsymbol{q}} + \dot{\boldsymbol{s}}_{\boldsymbol{o}}) + \boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1}\boldsymbol{b}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \Big)\end{aligned} \tag{11}$$

where the new terms $\ddot{\boldsymbol{s}}_{\boldsymbol{o}}$ and $\boldsymbol{D}_{\boldsymbol{s}}\dot{\boldsymbol{s}}_{\boldsymbol{o}}$ compensate for the target's motion.

Replacing Eq. (10) into (9) yields to the closed-loop system

$$\ddot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{D}_{\boldsymbol{s}}\dot{\boldsymbol{e}}_{\boldsymbol{s}} + \boldsymbol{K}_{\boldsymbol{s}}\boldsymbol{e}_{\boldsymbol{s}} = \boldsymbol{f}_{\boldsymbol{s}_{\boldsymbol{ext}}}$$

where $\boldsymbol{f}_{\boldsymbol{s}_{\boldsymbol{ext}}} \in \mathbb{R}^k$ is the vector of *p.u.m.i.* virtual forces (accelerations) acting on the features due to the external forces acting on the robot end-effector.

IV. EXTENDED KALMAN FILTER FOR TARGET TRACKING AND TARGET MOTION ESTIMATION

In the previous section, we derived a controller that linearizes and decouples the dynamics of the system via feedback linearization but, even though most of the terms in the controller can be easily computed, those related to the target's own motion are usually unknown and must thus be estimated. Since we do not assume, for the sake of generality, a particular model for the target motion, we treat it as a disturbance of the nominal motion of the features due to the motion of the camera as in (2), in which part of the feature velocity is given by the (unknown) target velocity $\dot{\boldsymbol{s}}_{\boldsymbol{o}} = \boldsymbol{L}_{\boldsymbol{s}}{}^{\boldsymbol{c}}\mathbf{v}_{\boldsymbol{o}}$.

To estimate the target motion we now describe two instances of an EKF used to estimate the target own motion for the two cases of image points and 3D pose error as visual features respectively. Note that both filters could be used in a classical VS control scheme to compensate for the target motion while here we exploit the entire state of the filters to implement the control law (11) for (i) virtually increasing the visual measurements frequency (*i.e.*, by using the state predictions as virtual measurements), and (ii) compensating for the target own motion and, thus, significantly reducing the tracking error.

A. EKF for IBVS with image points features

The goal of the filter is to both increase the data rate of the camera and to estimate the velocity of the moving target. The feature vector $\boldsymbol{s}$ will then be part of the filter state with a dynamics following the model equation (3). For each image point $f = (f_x, f_y)$, the associated Interaction matrix [1]

$$\boldsymbol{L}_{\boldsymbol{s}}(f, Z) = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{f_x}{Z} & f_x f_y & -(1+f_x^2) & f_y \\ 0 & -\frac{1}{Z} & \frac{f_y}{Z} & 1+f_y^2 & -f_x f_y & -f_x \end{bmatrix}$$

requires the knowledge of the depth of the feature Z which is also estimated by the filter as long as the trajectories are "exciting" enough. The depth dynamic is [13]

$$\dot{Z} = \boldsymbol{L}_Z(f, Z){}^{\boldsymbol{c}}\mathbf{v} = \begin{bmatrix} 0 & 0 & -1 & -f_y Z & f_x Z & 0 \end{bmatrix} {}^{\boldsymbol{c}}\mathbf{v}.$$

Furthermore, to implement controller (11), the feature acceleration contribution due to the motion of the target is also needed. The continuous-time state of the filter, stacking four image points is then $\boldsymbol{x}(t) = [\boldsymbol{s}(t)\ \boldsymbol{z}(t)\ \dot{\boldsymbol{s}}_{\boldsymbol{o}}(t)\ \ddot{\boldsymbol{s}}_{\boldsymbol{o}}(t)]^\top = [\boldsymbol{x_1}(t)\ \boldsymbol{x_2}(t)\ \boldsymbol{x_3}(t)\ \boldsymbol{x_4}(t)]^\top \in \mathbb{R}^{28}$, and expliciting the filter dynamics $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{w_t}$ one has:

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{L_s}(\boldsymbol{x_1}, \boldsymbol{x_2}){}^{c}\mathbb{T}_{e}{}^{e}\boldsymbol{J_e}\boldsymbol{u} - \boldsymbol{x_3} \\ \boldsymbol{L_Z}(\boldsymbol{x_1}, \boldsymbol{x_2})({}^{c}\mathbb{T}_{e}{}^{e}\boldsymbol{J_e}\boldsymbol{u} - \boldsymbol{L_s}^{\dagger}(\boldsymbol{x_1}, \boldsymbol{x_2})\boldsymbol{x_3}) \\ \boldsymbol{x_4} \\ \mathbf{0} \end{bmatrix} + \boldsymbol{w_t} \tag{12}$$

in which $\boldsymbol{u} = \dot{\boldsymbol{q}}$ is the plant input, $\boldsymbol{L_s}^{\dagger}$ is the Moore-Penrose pseudo-inverse of the Interaction matrix, and $\boldsymbol{w_t}$ is a vector of additive Gaussian noise $\boldsymbol{w_t} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_k)$. In the general case one does not have a model of the target motion, therefore we assume the dynamics of the acceleration of the features due to the target motion $\ddot{\boldsymbol{s}}_{\boldsymbol{o}}(t)$ to be approximately constant (corrupted by some added noise), i.e., $\dddot{\boldsymbol{s}}_{\boldsymbol{o}}(t) = \boldsymbol{w_{4_t}}$. Clearly if a motion model were available, it could be used, adjusting opportunely the filter dynamics.

To update the process state estimate with the (known) dynamics and its uncertainty, represented by the covariance matrix $\boldsymbol{P}$, we firstly need to linearize and discretize it at the control period ΔT:

$$\bar{\boldsymbol{A}} = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_k, \boldsymbol{u}=\boldsymbol{u}_k} \tag{13}$$

$$\begin{aligned} \boldsymbol{A}_k &= e^{\bar{\boldsymbol{A}}\Delta T} \approx \mathbb{I} + \bar{\boldsymbol{A}}\Delta T \\ \boldsymbol{x}_{k+1|k} &= \boldsymbol{x}_{k|k} + \boldsymbol{f}(\boldsymbol{x}_{k|k}, \boldsymbol{u}_k)\Delta T \\ \boldsymbol{P}_{k+1|k} &= \boldsymbol{A}_k \boldsymbol{P}_{k|k} \boldsymbol{A}_k^\top + \boldsymbol{Q}_k \end{aligned} \tag{14}$$

The state estimation improves during the *measurement update* phase based on the measurements $\boldsymbol{y}_k$. In our case, the measurement is the set of image point features in the image plane, leading then to a linear measurement equation in the filter state that is already in discrete time:

$$\boldsymbol{y}_k = \boldsymbol{s}_{k+1} + \boldsymbol{W}_{s_{k+1}} = \boldsymbol{C}_{k+1}\boldsymbol{x}_{k+1|k} + \boldsymbol{W}_{s_{k+1}}$$

with $\boldsymbol{W}_{s_{k+1}}$ a vector of discrete-time white noise with auto-correlation matrix $\boldsymbol{R}_{k+1}$, while $\boldsymbol{C}_{k+1} = [\mathbb{I}_8\ \mathbb{O}_{8\times 20}]$ is the measurement sensitivity matrix.

$$\boldsymbol{S}_{k+1} = (\boldsymbol{C}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{C}_{k+1}^\top + \boldsymbol{R}_{k+1}) \tag{15}$$

$$\begin{aligned} \boldsymbol{K}_{k+1} &= \boldsymbol{P}_{k+1|k}\boldsymbol{C}_{k+1}^\top \boldsymbol{S}_{k+1}^{-1} \\ \boldsymbol{x}_{k+1|k+1} &= \boldsymbol{x}_{k+1|k} + \boldsymbol{K}_{k+1}(\boldsymbol{y}_k - \boldsymbol{C}_{k+1}\boldsymbol{x}_{k+1|k}) \\ \boldsymbol{P}_{k+1|k+1} &= (\mathbb{I} - \boldsymbol{K}_{k+1}\boldsymbol{C}_{k+1})\boldsymbol{P}_{k+1|k} \end{aligned} \tag{16}$$

B. EKF for PBVS

The filter state for Pose-Based Visual Servoing (PBVS) is very similar to the previous case but without the depth dynamics. The feature $\boldsymbol{s}$ is a pose error vector and the state is $\boldsymbol{x}(t) = [\boldsymbol{s}(t)\ \dot{\boldsymbol{s}}_{\boldsymbol{o}}(t)\ \ddot{\boldsymbol{s}}_{\boldsymbol{o}}(t)]^\top = [\boldsymbol{x_1}(t)\ \boldsymbol{x_2}(t)\ \boldsymbol{x_3}(t)\ ]^\top \in \mathbb{R}^{18}$ with dynamics

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{L_s}(\boldsymbol{x_1}){}^{c}\mathbb{T}_{e}{}^{e}\boldsymbol{J_e}\boldsymbol{u} - \boldsymbol{x_2} \\ \boldsymbol{x_3} \\ \mathbf{0} \end{bmatrix} + \boldsymbol{w_t}.$$

In order to update the state estimate with the process equations, one can follow the same steps as in (13)–(14).

The "measurement" provided by the camera in the PBVS case is the relative pose of the observed object w.r.t. the camera frame, expressed as the homogeneous transformation

matrix ${}^{c}\boldsymbol{T}_{o}(\boldsymbol{y}_k)$. This leads to a discrete-time, non-linear and implicit measurement equation w.r.t. both the measure and filter state:

$$\boldsymbol{\eta}_{k+1} = \boldsymbol{h}(\boldsymbol{x}_{k+1|k}, \boldsymbol{y}_{k+1}). \tag{17}$$

The measurement equation (17) is a constraint equation ($\boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y}) = 0$) and to compute it, we first define the "closure" equation: ${}^{cd}\boldsymbol{T}_{c}(\boldsymbol{x}_{k+1|k})\ {}^{c}\boldsymbol{T}_{o}(\boldsymbol{y}_{k+1})\ {}^{cd}\boldsymbol{T}_{o}{}^{-1} = \mathbb{I}$, being ${}^{cd}\boldsymbol{T}_{o}$ the desired pose of the camera w.r.t. the object (which is of course known by design). The pose vector associated to the left-hand side of the *closure* equation must be zero, and this relationship represents the measurement equation of the form (17). The measurement update equations differ from those in (15)–(16) as follows:

$$\begin{aligned}
\boldsymbol{H}_{k+1} &= \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_{k+1|k}, \boldsymbol{y}=\boldsymbol{y}_{k+1}} \\
\boldsymbol{D}_{k+1} &= \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{y}}\right|_{\boldsymbol{x}=\boldsymbol{x}_{k+1|k}, \boldsymbol{y}=\boldsymbol{y}_{k+1}} \\
\boldsymbol{S}_{k+1} &= (\boldsymbol{H}_{k+1}\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top} + \boldsymbol{D}_{k+1}\boldsymbol{R}_{k+1}\boldsymbol{D}_{k+1}^{\top}) \\
\boldsymbol{K}_{k+1} &= -\boldsymbol{P}_{k+1|k}\boldsymbol{H}_{k+1}^{\top}\boldsymbol{S}_{k+1}^{-1} \\
\boldsymbol{x}_{k+1|k+1} &= \boldsymbol{x}_{k+1|k} + \boldsymbol{K}_{k+1}\boldsymbol{\eta}_{k+1} \\
\boldsymbol{P}_{k+1|k+1} &= \boldsymbol{P}_{k+1|k} + \boldsymbol{K}_{k}\boldsymbol{H}_{k}\boldsymbol{P}_{k+1|k}.
\end{aligned}$$

Once the filter converges, one can obtain the target velocity from the estimated disturbance state as ${}^{c}\mathbf{v}_{o}(t_k) = \boldsymbol{L_s}^{\dagger}\boldsymbol{x_3}(t_k)$ as we did in the depth dynamics of the EKF for IBVS (see second component of (12)).

V. SIMULATIONS

In order to validate the tracking performance of the proposed EKFs and study the feasibility of using DVS in a task involving contacts with the environment, we simulate a Peg-in-Hole task with a moving target using FrankaSim [1], a co-simulation environment implemented in ViSP [14] and Coppeliasim [15]. With FrankaSim it is possible to carry out dynamic simulations with a Panda robot, a 7-joint lightweight manipulator, and process different sensor information. The simulation was performed in synchronous mode with a time step of 1 ms using Vortex physics engine, since it is capable to deal with non-convex shapes. A vision sensor was mounted on the robot wrist and we programmatically drop and delay the grabbed frames in order to simulate a 30 fps (33 ms) camera rate with 10 ms of feature extraction delay. The visual features are extracted from an AprilTag of the family 36h11 using the ViSP detector (AprilTag pose or its corners). A 5×5 Gaussian mask is applied on the grabbed images to smooth the AprilTags' borders making the detection process more realistic. This is a convenient way to simulate noise because despite for points features it is easy to add Gaussian noise around the detected pixel, the uncertainties on the estimated pose depends on the distance of the object from the camera [16]. White noise of the magnitude of the real robot was also added to the measured joint velocities $\dot{\boldsymbol{q}}$ with standard deviation $\sigma = 0.021$ rad/s. The robot control loop runs at 1 kHz as for the real one.

[1]http://wiki.ros.org/visp_ros

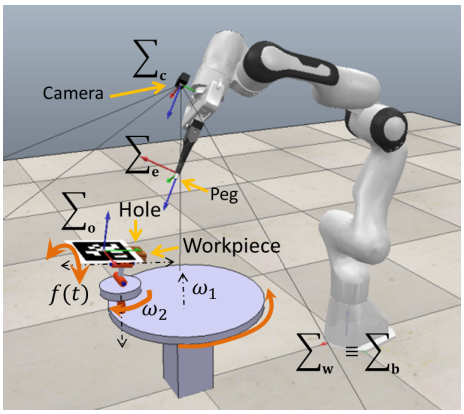

Fig. 1: Simulation setup: The Panda robot is in the initial configuration of the experiment. World and base $\Sigma_w \equiv \Sigma_b$, camera Σ_c, end-effector Σ_e and target Σ_o frames are drawn. An AprilTag is attached on the workpiece and tracked by the camera. The lower plate of the platform rotates at $\omega_1 = 0.5$ rad/s, the second plate at $\omega_2 = -\omega_1$ while the last joint motion is $f(t) = 0.2sin(2t)$.

A. Task Description and Controllers implementation

The simulated task consists in inserting a peg in the hole of a work-piece both cylindrical with diameters of 9 and 10 mm respectively. The piece is placed on a moving platform whose lowest joints turn at 0.5 rad/s in opposite directions allowing the upper part to undergo a purely translational motion, while the last joint of the platform performs a sinusoidal velocity motion of amplitude 0.2 rad/s and frequency $1/\pi$ Hz (see Fig. 1 and accompanying video).

Although the task constrains all the 6 degrees of freedom (DoF) of the operational space, our robot is a redundant manipulator and to prevent the robot joints to move along directions that lie in the kernel of the task Jacobian $\boldsymbol{J_s}$, the null-space directions must be damped (or a suitable secondary task must be considered). Controller (11) is then modified into the more computationally efficient form with the addition of a damping term in the null space of the task:

$$\begin{aligned}
\boldsymbol{u_\tau} = (\boldsymbol{J_s}\boldsymbol{B}(\boldsymbol{q})^{-1})^{\dagger}\Big(&\ddot{\boldsymbol{s}}^{*} + \boldsymbol{D_s}(\dot{\boldsymbol{s}}^{*} - \boldsymbol{L_s}\,{}^{c}\mathbb{T}_{e}\,{}^{e}\boldsymbol{J_e}\dot{\boldsymbol{q}} + \dot{\boldsymbol{s}}_{\boldsymbol{o}}) \\
&+ \boldsymbol{K_s}\boldsymbol{e_s} + \ddot{\boldsymbol{s}}_{\boldsymbol{o}} - \boldsymbol{h_q} + \int \boldsymbol{e_s}\Big) + \boldsymbol{c}(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{P}^{\perp}\boldsymbol{\tau}_N
\end{aligned} \tag{18}$$

in which the gravity term does not appear anymore since it is already compensated by the robot (in both the real and simulated cases), and $\boldsymbol{P}^{\perp}$ is a Null space projector that applies the secondary damping torques $\boldsymbol{\tau}_N = -k_d\dot{\boldsymbol{q}}$ on $\ker(\boldsymbol{J_s})$. Assuming the manipulator is in a non-singular configuration, in the PBVS case the task Jacobian has dimension 6×7 with $rank(\boldsymbol{J_s}) = 6$ and a Null space projector based on the dynamically consistent Inertia weighted pseudo-inverse $\boldsymbol{P}^{\perp} = (\mathbb{I} - \boldsymbol{J_s}^{\top}\bar{\boldsymbol{J}}_{\boldsymbol{s}}{}^{\top})$ can be computed [17], where $\bar{\boldsymbol{J}}_{\boldsymbol{s}} = \boldsymbol{B}^{-1}\boldsymbol{J_s}^{\top}(\boldsymbol{J_s}\boldsymbol{B}^{-1}\boldsymbol{J_s}^{\top})^{-1}$. On the other hand, in the IBVS case the task Jacobian is 8×7 with $rank(\boldsymbol{J_s}) = 6$ and more general methods must be used to find an orthogonal basis of $\ker(\boldsymbol{J_s})$, e.g., resorting to a Singular Value Decomposition (SVD) ($\boldsymbol{J} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}$) [18].

B. Simulation Results

We ran several simulations for both the proposed filters in which the estimates were used in the controller (18) to accomplish the task. The controller in the IBVS case exhibited large sensibility to depth errors and despite the good convergence observed for the depth estimation, we assumed for the experiments to have a "measurement" of the depth to overcome the persistency of excitation issue, since once the camera is above the target the trajectory does not excite the Z dynamics enough. The velocity estimation has a very good performance for both filters (see Fig. 2) but it is not perfect, and a small tracking error remains; an integral term was added to controller (18) when the feature error was sufficiently small to compensate for any final error and correctly realize the insertion task (see Figs. 3 and 4).

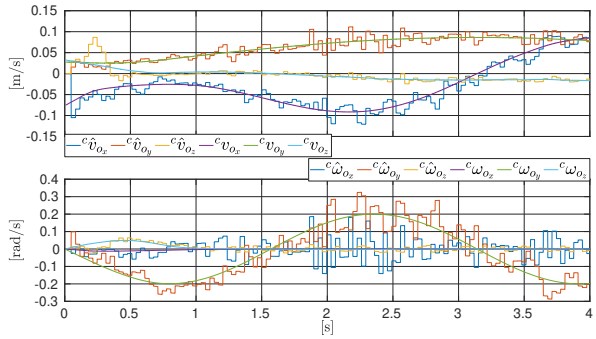

Fig. 2: Peg-in-Hole simulation: target linear (top) and angular (bottom) velocity estimates in camera frame vs ground-truth data collected from the simulation environment. The choice of a constant acceleration model for the target motion is well evident from the plotted curbs. Note how the angular velocity about the y axis ${}^{c}w_{o_y}$ follows the target motion about the same direction ($f(t) = 0.2 * sin(2t)$). This figure reports the results of the PBVS experiment, but similar results were achieved in the IBVS case (see accompanying video).

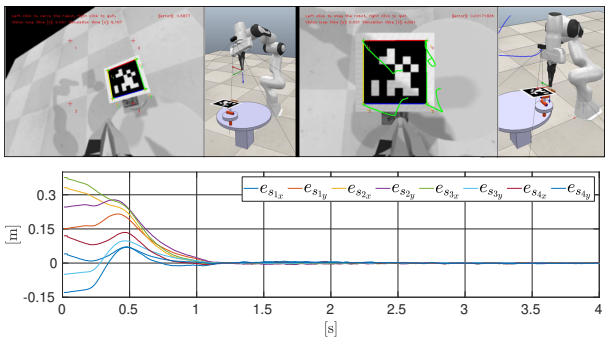

Fig. 3: IBVS Peg-in-Hole simulation: Initial (left) and final (right) camera views and robot configurations. The features errors are plotted. On the right image it is possible to appreciate, in green, the features trajectories in the image plane. Convergence is successfully attained.

With the proposed process model, the filter was able to properly reconstruct the disturbance (target velocity), thus succeeding in completing the task. Apart from the very beginning, which is affected by the transient behaviour of the filter convergence and the effects of a rapidly decaying

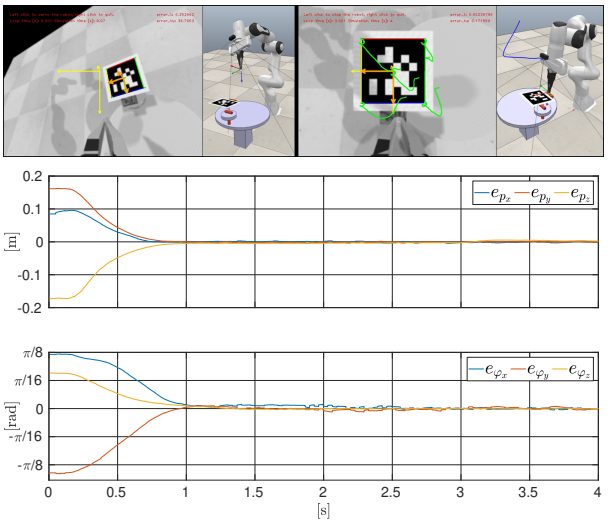

Fig. 4: PBVS Peg-in-Hole simulation: Initial (left) and final (right) camera views and robot configurations. Features position (top) and orientation (bottom) errors. On the right image it is possible to appreciate, in green, the trajectory of the target corners in the image plane. Convergence is successfully attained.

exponential $\boldsymbol{u}_\tau(0)e^{-\mu t}$ added to the torque command to guarantee a smooth start, the features trajectories show a nice exponential decrease. Simulations suggest that DVS controllers can be suitably used to deal with interactions tasks.

VI. EXPERIMENTS

An experiment for tracking a moving target on a turntable was performed on a Franka Emika Panda robot equipped with a wrist mounted RealSense D435 camera operating at 60 fps. Given the space limits, we report here only the results of the DVS for the IBVS case, which is more complex than its PBVS counterpart given both the sensitivity w.r.t. the feature depths and the difficulties related to the rank deficiency of the task Jacobian (more details below). For the PBVS case and some interaction experiments, please refer to the accompanying video in which the results of this work can be truly appreciated.

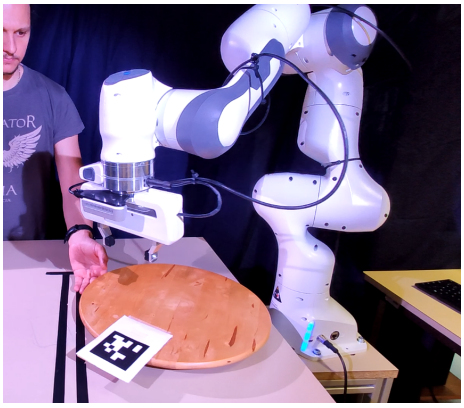

Fig. 5: Setup: Panda robot equipped with a RealSense D435 operating at 60 fps. The center of the target is located at 22 cm from the center of the turntable, which is manually operated.

A. Implementation issues

Most works have presented their results on DVS in simulation and tackling just few of the real world issues like measurement noise [6], [7], or instead of generating the torque command from the features acceleration, a velocity signal command is obtained from numerical integration and sent to the robot, as in [19] for an MPC controller. In this work we aimed at pushing forward the state of the art by tackling several implementation aspects of a real implementation not covered in previous works.

1) Data rate: The main bottleneck in a DVS controller implementation is certainly the low data rate coming from the vision sensor and the latency of the computer vision algorithms for the features extraction. The main contribution of this work are the proposed EKFs which increases the data rate and help to alleviate the negative effects of delayed measurements. If this problem is not addressed, it is practically impossible to implement such a control scheme without resorting to expensive high-speed cameras.

2) Ill-condition of the task Jacobian: The computation of the control command (18) requires to pseudo-invert the task Jacobian $\boldsymbol{J} = \boldsymbol{J_s}\boldsymbol{B}^{-1}$ which in practice turns out to be strongly ill-conditioned. This problem is due to the double effect of the ill-condition of the inertia matrix [20] which is then combined with the image Jacobian, resulting in a magnified singular values ratio. This issue is particularly pronounced when the disparity between the masses and inertia tensors of the individual links increases. Featherstone [21] has empirically found that the the condition number of the inertia matrix can asymptotically grow with $\mathcal{O}(n^4)$ of the number of bodies, which is our case as we are servoing a 7-joints manipulator. A common solution for dealing with ill-conditioned equations is through a regularization term such as in a Damped Least-squares (DLS) law [18]. However, we found that with a DLS-based regularization, rotations about x- and y-axis were far less "stiff" than the other directions while using an SVD-based regularization with a Gaussian function of the singular value as regularization term, $g(\epsilon_i) = m \exp(-\frac{\epsilon_i^2}{2\sigma^2})$ with ϵ_i the i-th singular value, the controller achieves better performance.

3) Joint friction: In [22], the dynamic friction for this robot has been identified and we used the released library to compensate for it. Unfortunately, we lack a model for dry/static friction, so small commanded torques do not result in actual robot motion, limiting the positioning performance close to the goal position; we have found this threshold to be ~ 0.5 Nm for our robot. This phenomenon particularly affects the last joint since the controlled torques are often below this value. The integral term on the features error we add before alleviate this issue, allowing to reduce the remaining steady state error. Further increasing the proportional gain could in fact leads to an unstable behaviour.

4) Illumination conditions: Despite evident when working with vision sensors, experiments are often conducted under poor illumination conditions with consequent performance degradation. In our case, it is of paramount importance to guarantee and maintain a continuous and constant feature detection rate to avoid the data fusion of too much delayed visual measurements, leading to catastrophic results. A good illuminated environment stabilizes the feature extraction with ViSP to ~ 8 ms while in regular conditions it can oscillate from 8 to 20 ms. Spotlights were then placed around the experimental platform also trying to avoid obscuring the target or blinding the camera with reflections.

B. Results

The proposed EKF proves to be effective both in increasing the data rate of the camera, providing an accurate estimate of the features (and depth) at higher frequency, and in alleviating the effects due to delays in the feature extraction process as well as in providing an accurate estimate of the target velocity. In Fig. 6 we can see how the feature error remains limited despite the highly dynamic motion (velocities up to 40 [cm/s] and almost 2 [rad/s] with high accelerations) manually applied to the turntable. Once the target is kept at rest, the feature errors slowly converges to zero. This is due to the high dry friction in the joints that the integral term has to overcome. Dry friction remains the main issue limiting the tracking accuracy of our implementation while SVD-based regularization managed to cope with the strong ill-conditioning of the task Jacobian.

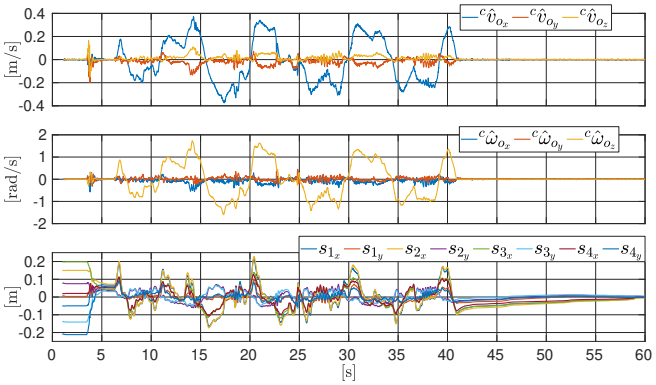

Fig. 6: Estimated target linear (Top) and angular (middle) velocities in camera frame compared to the feature errors (bottom). The circular motion produces a linear velocity component tangential to the circle and from a frame on top of the target it produces a linear and an angular velocity in camera frame along the y-axis and about the z-axis respectively.

VII. CONCLUSIONS

In this paper we presented our preliminary results on DVS for interaction control and moving targets. The proposed EKFs effectively track the target motion and provide high rate visual information allowing for the implementation on a real platform with very good performance for the motion tracking. Several real-world implementation issues have been discussed alongside the solutions adopted. We recognize in the dry friction the main cause limiting the tracking performances of our implementation, especially at low speeds. In future works we plan to integrate friction models to improve the performance and to consider passivity-based techniques to further increase the system stability to robustly use such controllers in dynamic interaction tasks.

REFERENCES

[1] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 3, pp. 313–326, 1992.

[2] F. Chaumette, "Image moments: a general and useful set of features for visual servoing," *IEEE Transactions on Robotics*, vol. 20, no. 4, pp. 713–723, 2004.

[3] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.

[4] ——, "Visual servo control. ii. advanced approaches [tutorial]," *IEEE Robotics Automation Magazine*, vol. 14, no. 1, pp. 109–118, 2007.

[5] Hong Zhang and J. P. Ostrowski, "Visual servoing with dynamics: control of an unmanned blimp," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, 1999, pp. 618–623 vol.1.

[6] S. Vandernotte, A. Chriette, P. Martinet, and A. S. Roos, "Dynamic sensor-based control," in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2016, pp. 1–6.

[7] F. Fusco, O. Kermorgant, and P. Martinet, "A comparison of visual servoing from features velocity and acceleration interaction models," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4447–4452.

[8] M. Vukobratović and D. Stokić, "Is dynamic control needed in robotic systems, and, if so, to what extent?" *The International Journal of Robotics Research*, vol. 2, no. 2, pp. 18–34, 1983. [Online]. Available: https://doi.org/10.1177/027836498300200202

[9] B. Nelson, J. Morrow, and P. Khosla, "Robotic manipulation using high bandwidth force and vision feedback," *Mathematical and Computer Modelling*, vol. 24, no. 5, pp. 11–29, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0895717796001136

[10] A. A. Oliva, P. Robuffo Giordano, and F. Chaumette, "A general visual-impedance framework for effectively combining vision and force sensing in feature space," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4441–4448, 2021.

[11] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modeling, Planning and Control*, 3rd ed. London: Springer, 2008.

[12] F. Chaumette, S. Hutchinson, and P. Corke, "Visual servoing," *Handbook of Robotics, 2nd edition*, pp. 841–866, 2016.

[13] A. D. Luca, G. Oriolo, and P. Robuffo Giordano, "On-line estimation of feature depth for image-based visual servoing schemes," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 2007, pp. 2823–2828.

[14] E. Marchand, F. Spindler, and F. Chaumette, "Visp for visual servoing: a generic software platform with a wide class of robot control skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, pp. 40–52, December 2005.

[15] E. Rohmer, S. P. N. Singh, and M. Freese, "Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. [Online]. Available: www.coppeliarobotics.com

[16] T. Larsen, N. Andersen, O. Ravn, and N. Poulsen, "Incorporation of time delayed measurements in a discrete-time kalman filter," in *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, vol. 4, 1998, pp. 3972–3977 vol.4.

[17] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, 1987.

[18] E. Simetti and G. Casalino, "A novel practical technique to integrate inequality control objectives and task transitions in priority based control," *Journal of Intelligent & Robotic Systems*, vol. 84, pp. 877–902, 2016.

[19] F. Fusco, O. Kermorgant, and P. Martinet, "Integrating features acceleration in visual predictive control," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5197–5204, 2020.

[20] A. Maciejewski, "Dealing with the ill-conditioned equations of motion for articulated figures," *IEEE Computer Graphics and Applications*, vol. 10, no. 3, pp. 63–71, 1990.

[21] R. Featherstone, "An empirical study of the joint space inertia matrix," *I. J. Robotic Res.*, vol. 23, pp. 859–871, 09 2004.

[22] C. Gaz, M. Cognetti, A. Oliva, P. Robuffo Giordano, and A. De Luca, "Dynamic identification of the Franka Emika Panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Lett.*, 2019.