IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

Second-Order Position-Based Visual Servoing of a Robot Manipulator

Eduardo G. Ribeiro[1], Raul Q. Mendes[2], Marco H. Terra[1] and Valdir Grassi Jr[1]

***Abstract*—Visual Servoing is an established approach for controlling robots using visual feedback. Most controllers in this domain generate velocity control signals to guide the cameras to desired positions and orientations. However, the dynamic characteristics of conventional visual servoing controllers may be unsatisfactory, and the velocity signal itself hinders the connection between the feature velocity model and the robot's dynamics. Consequently, research has explored models incorporating the second-order derivative of features and the robot's acceleration. The current state-of-the-art techniques mainly focus on image-based visual servoing, which deals with feature errors in the image domain. In this work, we propose an acceleration-based controller for the position-based visual servoing framework, which models the error in Cartesian space. Our approach involves extracting an acceleration control signal from the traditional velocity-based controller. To achieve this, we redefine the camera orientation using quaternions, generate new interaction matrices, and conduct comprehensive comparative experiments in simulated and real robot scenarios. We show that our method provides better dynamic properties in both image and Cartesian spaces, superior tracking performance, and less sensitivity to noise compared to velocity controllers.**

***Index Terms*—Visual Servoing, Motion Control, Dynamic Visual Servoing, Position-Based Manipulator Control.**

I. INTRODUCTION

VISUAL Servoing (VS) has been used in robotics for precise positioning control between a vision sensor and an object of interest [1], as well as to make machines more agile and dexterous [2]. The controller is usually designed at a high level, resulting in a velocity profile that is forwarded to the low-level robot controller, which then generates the required torque commands for the actuators [1], [2], [3].

Another less explored class of visual servo controllers removes the distinction between these levels and directly derives the relationship between image features and joint torques. In the literature, they are known as *Dynamic* or *Second-Order* (SO) visual servoing. The term *Dynamic* is typically used to refer to the inclusion of the robot's dynamic model in the controller design, and the term *Second-Order* can be found interchangeably. However, it carries its particular meaning as it defines the dynamics of the error, establishing a relationship between the features' acceleration and the acceleration of the vision sensor, which may be integrated into the torque control, as illustrated in Fig. 1.

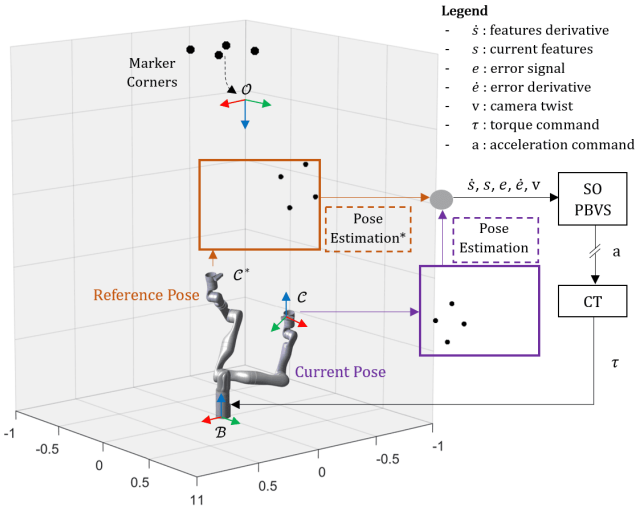

Fig. 1. SO-PBVS framework. The eye-in-hand mounted camera detects the marker corners (object of interest), which is used for pose estimation. The reference (*) and current poses are combined to get $\mathbf{s}$ and $\mathbf{e}$, whereas $\dot{\mathbf{s}}$, $\dot{\mathbf{e}}$, and $\mathbf{v}$ are further inferred. The error is regulated through an acceleration command, which may be sent to the robot's internal controller, or be integrated into a Computed Torque (CT) control to resolve the joint torques. $\mathcal{O}$, $\mathcal{B}$, and $\mathcal{C}$ are the object, base, and camera reference frames, respectively.

Second-order VS is recognized for its superior dynamic properties when compared to velocity controllers in both image and 3D spaces [2]. It naturally enables the use of PD and PID controllers and computed torque control directly in the feature space [1], which can deliver high speed, accurate trajectory tracking [2], and dynamic perturbations rejection while the robot is regulated at a fixed reference. Those features can be useful, for instance, in robotic endoscopy [4] and reactive robotic manipulation [5].

In addition to the classification between velocity controllers, from now on called First-Order (FO), and second-order controllers, VS schemes can be classified according to the nature of the visual information provided to the control system. Image-Based Visual Servoing (IBVS) regulates the error signal in image space, while Position-Based Visual Servoing (PBVS) regulates the error in Cartesian space [3].

Several works have focused on developing second-order controllers in the IBVS formulation [6], [2], [7], [1], [8], [9]. On the other hand, the majority of research in PBVS has been centered around obtaining dynamic controllers. Indeed, the concept of dynamic PBVS was considered over two decades ago [10] during a time when addressing the gap between the

This work was supported by the Brazilian National Council for Scientific and Technological Development (grant 465755/2014-3 and 380973/2020-0), by the Coordination of Improvement of Higher Education Personnel - Brazil (Finance Code 001, 88887.136349/2017-00), and the São Paulo Research Foundation (grant 2014/50851-0).

[1] Eduardo G. Ribeiro, Marco H. Terra and Valdir Grassi Jr are with the Department of Electrical and Computer Engineering, São Carlos School of Engineering, University of São Paulo, São Carlos, Brazil (eduardogr@usp.br, terra@sc.usp.br, vgrassi@usp.br).

[2] Raul Q. Mendes is with the Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands (r.de.queiroz.mendes@tue.nl).

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

image processing and the robot controller frequencies was of significant concern. While research in this area remains relevant, with alternative approaches being explored [11], [12], [13], dynamic PBVS primarily serves as an intermediate pose estimation step for feedback into the robot's torque control. Consequently, there are limited studies that delve into the concept of second-order PBVS [14].

The commonly adopted approach in SO-IBVS stems from the derivation of the classical VS law [3], which establishes a relationship between the temporal derivative of the features and the camera twist through an interaction matrix. This expression was derived by Pomares et al. [6], Keshmiri et al. [2], and Vandenotte et al. [7], offering broad generalization capabilities for features beyond image points. Pomares et al. [6] focused on obtaining a control law with chaos compensation, while Vandenotte et al. [7] dealt with a hybrid system requiring force control. In contrast, Keshmiri et al. [2] provide a more formal derivation of SO-IBVS, including a stability proof and comparative analyses between FO-IBVS and SO-IBVS on a real robot.

Fusco et al. [1] go further with this comparison by considering feature tracking scenarios and accounting for noisy measurements. They also formulate a general law that establishes the relationship between feature acceleration and joint torque. However, their findings were limited to simulations. In a subsequent study [8], the authors integrate the second-order feature dynamics into a model predictive control framework and conduct real experiments to compare different feature options.

In the PBVS scenario, Fakhry et al. [11] and Kuo et al. [13], despite being separated by more than two decades, employ the same concept of a position-based VS to estimate the pose of manipulators and subsequently derive a resolved acceleration controller (RAC) for the robot's joints. Dahmouche et al. [12] follows a similar approach but utilizes Computed Torque (CT) control instead of RAC.

Note that in such cases there is no high-level controller that generates an acceleration signal, which, as mentioned, defines a SO-PBVS. Zhang et al. [14] designed a SO-PBVS using dynamic surface control that generates an acceleration signal based on the estimated relative pose. This signal is then sent to the robot's internal controller. Interestingly, this work shows that it is possible to dissociate the VS from the robot controller and discusses the effects of sending the integrated signal (velocity) to obtain joint torque. In fact, few robotic systems allow direct torque control, and other SO works follow the same idea.

In this letter, we aim to develop a novel SO-PBVS based on the control law presented in the classical works of SO-IBVS [2], [1]. However, the conventional PBVS interaction matrices employ the axis-angle representation [3], which poses challenges in obtaining the temporal derivative of these matrices required for the second-order formulation. To tackle this, we propose to use unit quaternions to represent camera orientation. This not only simplifies the design process but also eliminates the singularities that arise from the trigonometric terms in the axis-angle representation [15].

Of course, quaternions have been considered before in visual servoing, both as unit quaternion [16], [17], and in the dual form [18], [19]. The representation adopted by Hu et al. [15] in a homography-based control is similar to ours since it relates the derivative of the quaternion and the angular velocity of the camera. However, here we develop the information in the camera frame and evaluate the second-order dynamics of the error.

To the best of our knowledge, no prior work has presented a SO-PBVS derived from the classical FO-PBVS formulation, nor has there been a comprehensive comparison between these two controllers. In addition to the controller design, we conduct comparative analyses between the SO-PBVS, FO-PBVS, and SO-IBVS, both in Cartesian and image spaces, considering both moving and fixed features, in both simulated and real robot scenarios.

The main contributions of this letter are as follows:

- A new second-order formulation for the classical PBVS;
- Redefinition of camera orientation from axis-angle representation to quaternion;
- Comprehensive comparative experiments in simulation considering the dynamic model of the robot;
- Comparative experiments with the PBVS formulations, in high-level, using a real robot.

The remainder of this letter is organized as follows: Section II presents the problem formulation and our proposed solution, Section III presents the simulation results, Section IV presents the experimental results, and Section V concludes the work.

II. PROBLEM FORMULATION

The error signal to be minimized during visual servoing is described as [3]

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

where $\mathbf{m}(t)$ is a vector of measurements in the image, such as the coordinates of some points of interest, $\mathbf{s}(\mathbf{m}(t), \mathbf{a})$ are j features obtained from the measurements, $\mathbf{a}$ is a set of additional information, such as the camera's intrinsic parameters, and $\mathbf{s}^*$ encodes the expected values of the features [3].

Once $\mathbf{s}$ is selected, it is usual to design a proportional velocity controller, which we refer to as First-Order (FO) controller. To achieve this, we need the relationship between the time variation of $\mathbf{s}$ and the twist of the camera $\mathbf{v}^T = [\boldsymbol{v}^T \ \boldsymbol{\omega}^T]$, expressed in its own frame. Assuming that variations of $\mathbf{s}$ come only from the relative motion between the camera and the object, we have

$$\dot{\mathbf{s}} = \mathbf{L}_\mathbf{s}\mathbf{v}, \tag{2}$$

where $\mathbf{L}_\mathbf{s} \in \mathbb{R}^{j\times 6}$ is the interaction matrix.

To define $\mathbf{s}$, PBVS utilizes the camera's position and orientation relative to a reference frame, which are obtained through pose estimation algorithms [3]. Consequently, $\mathbf{s}$ can be represented as $(\mathbf{t}, \mathbf{r})$, where $\mathbf{t}$ denotes the translation vector, and $\mathbf{r}$ represents a suitable parameterization of the rotation, as elaborated in the following section.

First, we establish the relevant coordinate frames. We define $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{O}$ as orthogonal frames attached to the robot's base, the camera mounted in the *eye-in-hand* configuration,

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2024, Yokohama, Japan. Cite as RA-L paper.

and the target object, respectively, as illustrated in Fig. 1. $\mathcal{C}$ corresponds to the current camera frame, while $\mathcal{C}^*$ represents the desired camera frame.

A. FO-PBVS with quaternion

Quaternions are an extension of complex numbers to a higher-dimensional space, forming a four-dimensional associative, non-commutative set [18]:

$$\mathbb{H} \triangleq \{q_0 + \mathbf{q} : \mathbf{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k},\ q_0, q_1, q_2, q_3 \in \mathbb{R}\}, \quad (3)$$

where, for any $q \in \mathbb{H}$, q_0 is the real part, $\mathbf{q}$ is the imaginary part, and $\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$ are unit vectors such that $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$.

A quaternion conjugate is defined as $\bar{q} = q_0 - \mathbf{q}$, and its norm is defined as $\|q\| = \sqrt{q \circ \bar{q}}$, where $\circ$ denotes the product (Hamilton product) of quaternions. In this work, we are particularly interested in the unit quaternion ($\|q\| = 1$), as it provides a global nonsingular parametrization of spatial rotations [15] and is useful for performing operations with spatial orientations.

We want to represent the current camera frame orientation, which composes the feature vector of the classic PBVS, $\mathbf{s} = (\mathbf{t}, \theta\mathbf{u})$, using quaternion, $\mathbf{s} = (\mathbf{t}, q)$. To achieve this, we rely on the well-known kinematic equation [20],

$$\dot{q} = \frac{1}{2}\omega^b \circ q, \quad (4)$$

which relates the quaternion derivative and the pure quaternion representation ($\omega = 0 + \boldsymbol{\omega}$) of the angular velocity of a body relative to a fixed frame ($\mathcal{B}$). Note that we are interested in the imaginary part of ω, which is the angular velocity vector $\boldsymbol{\omega}$.

The relationship between the fixed reference frame and the reference frame attached to the body ($\mathcal{C}$), is given by $\omega^b = q \circ \omega^c \circ \bar{q}$. Injecting this into (4), we have

$$\dot{q} = \frac{1}{2} q \circ \omega^c \circ \bar{q} \circ q, \quad (5)$$

and assuming $\bar{q} \circ q = (1, \mathbf{0})$, we obtain the angular velocity expressed in the camera frame:

$$\dot{q} = \frac{1}{2} q \circ \omega^c. \quad (6)$$

Rearranging, after multiplying both sides by $\bar{q}$, we have

$$\omega^c = 2\bar{q} \circ \dot{q}. \quad (7)$$

Opening the Hamilton product and using the definition of a quaternion conjugate, we have

$$\omega^c = 2\bar{q} \circ \dot{q} = 2(q_0\dot{q}_0 + \mathbf{q}^T\dot{\mathbf{q}},\ q_0\dot{\mathbf{q}} - \dot{q}_0\mathbf{q} - \mathbf{q} \times \dot{\mathbf{q}}), \quad (8)$$

which in matrix representation takes the form

$$\omega^c = 2\mathbf{E}(q)\dot{q}. \quad (9)$$

Since $\|q\| = 1$, we have that $\mathbf{E}(q)^T\mathbf{E}(q) = \mathbf{I}$, and (9) can be rewritten as

$$\dot{q} = \frac{1}{2}\mathbf{E}(q)^T\omega^c. \quad (10)$$

As ω^c is a pure quaternion, we can remove the first column of $\mathbf{E}(q)^T$ and obtain

$$\dot{q} = \mathbf{B}(q)\boldsymbol{\omega}^c, \quad (11)$$

with $\mathbf{B}(q) \in \mathbb{R}^{4\times3}$, such that

$$\mathbf{B}(q) = \frac{1}{2}\begin{bmatrix} -\mathbf{q}^T \\ q_0\mathbf{I_3} + [\mathbf{q}]_\times \end{bmatrix}, \quad (12)$$

which differs from the matrix that would be obtained for the relationship between $\dot{q}$ and ω^b, where the second row is $q_0\mathbf{I_3} - [\mathbf{q}]_\times$. Here, $\mathbf{I_3}$ is the 3×3 identity matrix, and $[.]_\times$ is the skew-symmetric representation of a vector.

Matrix $\mathbf{B}(q)$ corresponds to the component of the interaction matrix that connects the changing orientation of the camera frame to its angular velocity. Additionally, it is necessary to establish the connection between the variation in the camera's position and both its linear and angular velocity.

The conventional formulation of PBVS, as presented in Chaumette and Hutchinson [3], provides two alternative definitions for the camera's position $\mathbf{t}$. The orientation is held fixed as the rotation between the current and the desired camera frames, denoted by ${}^{c^*}\mathbf{R}_c$.

1) Translation relative to the desired camera frame (${}^{c^*}\mathbf{t}_c$)*:* By defining $\mathbf{s}$ as $({}^{c^*}\mathbf{t}_c, \mathbf{q})$, we can derive the corresponding interaction matrix as follows

$$\mathbf{L}_\mathbf{s}^\mathbf{1} = \begin{bmatrix} \mathbf{R}(q) & \mathbf{O_3} \\ \mathbf{O_3} & \mathbf{B_1}(q) \end{bmatrix}, \quad (13)$$

where $\mathbf{O_3}$ is the null 3×3 matrix and $\mathbf{R}(q)$ is the rotation matrix ${}^{c^*}\mathbf{R}_c$ obtained from q,

$$\mathbf{R}(q) = (q_0^2 - \mathbf{q}^T\mathbf{q})\mathbf{I_3} + 2\mathbf{q}\mathbf{q}^T + 2q_0[\mathbf{q}]_\times. \quad (14)$$

Note that we select only the imaginary part $\mathbf{q}$ of the quaternion q as features. Therefore, in (13) we use a submatrix $\mathbf{B_1}(q)$ of $\mathbf{B}(q)$ without its first row. This selection ensures that the interaction matrix remains square with dimensions 6×6, which is important to ensure the stability of the PBVS [3]. Moreover, the real term q_0 is not neglected, as it is still required for computing $\mathbf{R}(q)$ and $\mathbf{B_1}(q)$.

2) Translation of the object frame relative to the camera frame (${}^c\mathbf{t}_o$)*:* By defining $\mathbf{s}$ to be $({}^c\mathbf{t}_o, \mathbf{q})$, the following interaction matrix is achieved

$$\mathbf{L}_\mathbf{s}^\mathbf{2} = \begin{bmatrix} -\mathbf{I_3} & [{}^c\mathbf{t}_o]_\times \\ \mathbf{O_3} & \mathbf{B}_1(q) \end{bmatrix}, \quad (15)$$

which couples translation and rotation movements.

B. SO-PBVS

Given (2), the relationship between the second-order derivative of features and the camera acceleration $\mathbf{a}$ is [1]

$$\ddot{\mathbf{s}} = \mathbf{L_s}\left(\mathbf{a} + \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{\omega} \\ \mathbf{0} \end{bmatrix}\right) + \dot{\mathbf{L}}_\mathbf{s}\mathbf{v}, \quad (16)$$

where the term enclosed in parentheses represents the derivative of the camera twist, employing the principles of screw theory to calculate the derivative of a moving body relative to a fixed body.

The analytical derivative of the interaction matrix is computed using the chain rule for multivariate functions. As a result, similar to how $\mathbf{L_s}$ depends on $\mathbf{s}$, the derivative matrix $\dot{\mathbf{L}}_\mathbf{s}$ is also a function of both $\mathbf{s}$ and $\dot{\mathbf{s}}$ at each control iteration:

$$\dot{\mathbf{L}}_\mathbf{s} \triangleq \mathbf{L}_{\mathbf{s},\dot{\mathbf{s}}} = \mathbf{L}_{\mathbf{s},\dot{\mathbf{s}}}(\mathbf{s},\dot{\mathbf{s}}) = \mathbf{L}_{\mathbf{s},\dot{\mathbf{s}}}\left(\mathbf{s}, \mathbf{L_s v}\right). \tag{17}$$

By rearranging the terms in (16), we can access the camera acceleration as follows

$$\mathbf{a} = \widehat{\mathbf{L}_\mathbf{s}^+}\left(\ddot{\mathbf{s}} - \mathbf{L_s}\begin{bmatrix}\boldsymbol{v}\times\boldsymbol{\omega}\\ \mathbf{0}\end{bmatrix} - \mathbf{L}_{\mathbf{s},\dot{\mathbf{s}}}\mathbf{v}\right), \tag{18}$$

where $\widehat{\mathbf{L}_\mathbf{s}^+}$ is the estimation of the pseudo-inverse of $\mathbf{L_s}$.

The visual servoing control aims to reduce the error $\mathbf{e_s}$ in the feature domain to zero. Considering the second-order model, we can benefit from a proportional derivative controller, so that the exponential decrease of the error follows (19):

$$\ddot{\mathbf{e}}_\mathbf{s} + k_d\dot{\mathbf{e}}_\mathbf{s} + k_p\mathbf{e_s} = 0, \tag{19}$$

where k_p and k_d are the proportional and derivative control gains. By injecting (19) into (18), we derive the expression for the robot acceleration command in the task space:

$$\mathbf{a} = \widehat{\mathbf{L}_\mathbf{s}^+}\left(\ddot{\mathbf{s}}^* - k_d\dot{\mathbf{e}}_\mathbf{s} - k_p\mathbf{e_s} - \mathbf{L_s}\begin{bmatrix}\boldsymbol{v}\times\boldsymbol{\omega}\\ \mathbf{0}\end{bmatrix} - \mathbf{L}_{\mathbf{s},\dot{\mathbf{s}}}\mathbf{v}\right). \tag{20}$$

The proof and stability conditions of (20) are presented by Keshmiri et al. [2]. Unlike IBVS, PBVS has the desirable property that, given an ideal pose estimation, $\mathbf{L_s}\widehat{\mathbf{L}_\mathbf{s}^+} = \mathbf{I_6}$, which satisfies the initial condition $\mathbf{L_s}\widehat{\mathbf{L}_\mathbf{s}^+} > 0$ (sufficient for the first-order case). Certainly, errors in the pose estimation can affect stability, yet if $\mathbf{L_s}$ and $\widehat{\mathbf{L}_\mathbf{s}^+}$ are still full rank, the stability is guaranteed. Furthermore, $\mathbf{L_s}$ is non-singular as the orientation is calculated with quaternions, unlike in the axis-angle case, where $\mathbf{L_s}$ is singular when $\theta = 360n$ [3].

Since we are using an eye-in-hand setup, the camera twist can be linked to the robot joint space by

$$\mathbf{v} = \mathbb{A}\mathbf{J}\dot{\boldsymbol{\theta}} = \begin{bmatrix}{}^b\mathbf{R}_c^T & -{}^b\mathbf{R}_c^T[{}^b\mathbf{t}_c]_\times\\ \mathbf{O_3} & {}^b\mathbf{R}_c^T\end{bmatrix}\mathbf{J}\dot{\boldsymbol{\theta}}, \tag{21}$$

in which $\mathbb{A}$ is the adjoint matrix that maps the kinematic twist to the camera frame, $\mathbf{J}$ is the $6\times p$ Jacobian of the robot with p degrees of freedom, and $\dot{\boldsymbol{\theta}}$ refers to the velocities of the joints.

Given (21), the camera acceleration is related to the robot joint space through

$$\mathbf{a} = \mathbb{A}\mathbf{J}\ddot{\boldsymbol{\theta}} + \mathbb{A}\dot{\mathbf{J}}\dot{\boldsymbol{\theta}}, \tag{22}$$

where $\ddot{\boldsymbol{\theta}}$ is the robot's joint acceleration. It is possible to rearrange (22) to determine the control effort directly in the robot joints as follows,

$$\ddot{\boldsymbol{\theta}} = (\mathbb{A}\mathbf{J})^+\left(\mathbf{a} - \mathbb{A}\dot{\mathbf{J}}\dot{\boldsymbol{\theta}}\right). \tag{23}$$

Finally, (23) can be inserted in the inverse dynamic model of the robot manipulator to retrieve the joint torques,

$$\boldsymbol{\tau} = \mathbf{M}(\boldsymbol{\theta})\ddot{\boldsymbol{\theta}} + \mathbf{C}(\boldsymbol{\theta},\dot{\boldsymbol{\theta}})\dot{\boldsymbol{\theta}} + \mathbf{G}(\boldsymbol{\theta}) + \mathbf{F_v}\dot{\boldsymbol{\theta}} + \mathbf{F_c}\text{sign}(\dot{\boldsymbol{\theta}}), \tag{24}$$

where $\boldsymbol{\tau}$ is the vector of joint torques, $\mathbf{M}$ is the inertia matrix, $\mathbf{C}$ is the vector of Coriolis and centripetal forces, $\mathbf{G}$ are efforts exerted by gravity and $\mathbf{F_v}$ and $\mathbf{F_c}$ are viscous and Coulomb frictional effects, respectively. As a final note, although small dynamic uncertainties and perturbations in the robot model can be handled by the SO-PBVS alone, further Computed Torque (CT), *e.g.*, in [2], or adaptive CT may be necessary to improve robustness and performance.

III. SIMULATION RESULTS

To assess the controller's performance and compare it between the first and second-order formulations, we utilized a simulated environment that incorporated the robot's dynamic model. Building upon the comparison criteria between IBVS formulations established by Fusco et al. [1], we evaluated three control variations: FO-PBVS, SO-PBVS and a first-order controller with a feed-forward term (FOFF-PBVS).

The camera velocity control signal provided by the FO-PBVS variation is retrieved from (2):

$$\mathbf{v} = \widehat{\mathbf{L}_\mathbf{s}^+}\left(\dot{\mathbf{s}}^* - k_p\mathbf{e_s}\right), \tag{25}$$

where we leverage the dynamics of the error as $\dot{\mathbf{e}}_\mathbf{s} + k_p\mathbf{e_s}$=0.

Following [1], we include $e^{-\mu t}\mathbf{e_s}(0)$ to the error dynamics, and the camera velocity is rewritten as:

$$\mathbf{v} = \widehat{\mathbf{L}_\mathbf{s}^+}\left(\dot{\mathbf{s}}^* - k_p\mathbf{e_s} + e^{-\mu t}\mathbf{e_s}(0)\right). \tag{26}$$

This removes the discontinuity of the first control iteration and makes the error evolution behave close to the second-order one.

In order to exploit the dynamic model of the robot with the FO-PBVS, it is necessary to compute an acceleration signal from the velocity predicted by the controller. Thereby, the acceleration of FO-PBVS is retrieved by:

$$\mathbf{a} = \gamma(\mathbf{v} - \mathbf{v}_c), \tag{27}$$

where $\mathbf{v}_c$ is the actual current camera velocity, and γ is a proportional adjustment gain. Higher values of γ cause $\mathbf{v}_c$ to converge faster to $\mathbf{v}$.

The FOFF-PBVS, a variation of FO-PBVS with a feed-forward term, $\mathbf{a}_{ff}$, is also considered. This term corresponds to the numerical derivative of (26), with

$$\mathbf{a} = \mathbf{a}_{ff} + \gamma(\mathbf{v} - \mathbf{v}_c). \tag{28}$$

Regarding the SO-PBVS, the camera acceleration is calculated with (20). Once we have the acceleration signals for all three controllers, we can obtain the joint torques using (23) and (24).

The simulation environment is implemented in Matlab, with all calculations related to visual servoing developed from scratch. The robot's dynamics were extracted from the Robotics System Toolbox, and perspective projection operations were executed with the Epipolar Geometry Toolbox. We used the 7DoF robotic manipulator Kinova Gen3, whose dynamic parameters were identified in [21].

We define four coplanar points in the world as our object of interest, representing the vertices of a marker. The origin of the target object's coordinate system is located at $z = 2.5$m, and its x-axis is rotated by 180° with respect to the robot's base, as shown in Fig. 1.

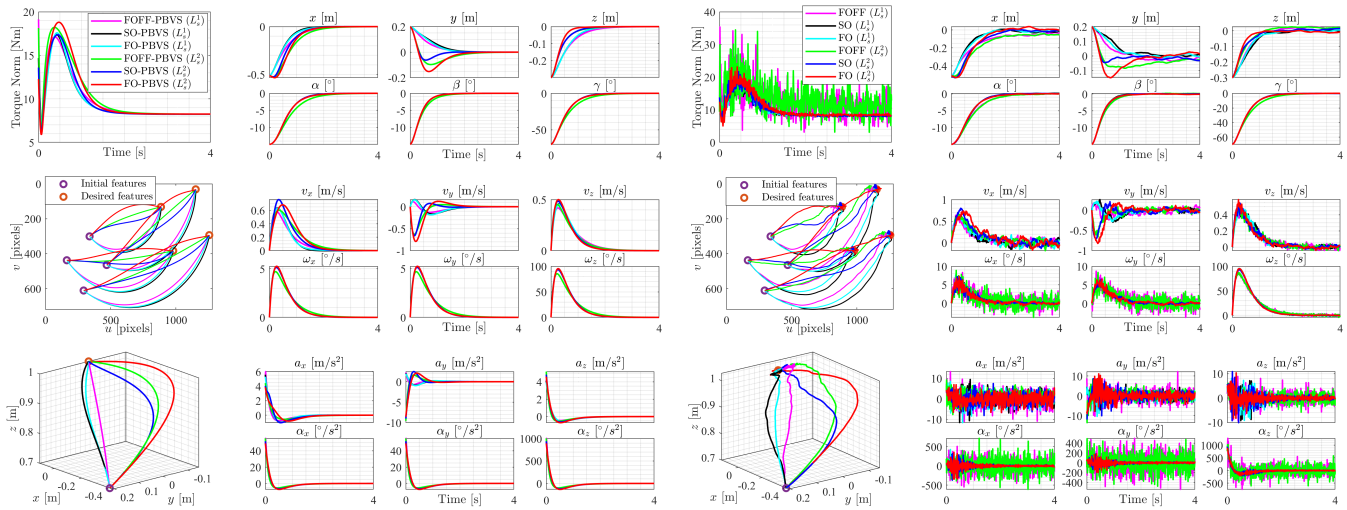

Fig. 2. Comparison of FO and SO formulations for PBVS. First column: torque norm (up), image plane (center) and 3D trajectory (bottom). Second column: error evolution (up), camera velocity (center) and camera acceleration (bottom). Third and fourth columns replicate results when noise is considered.

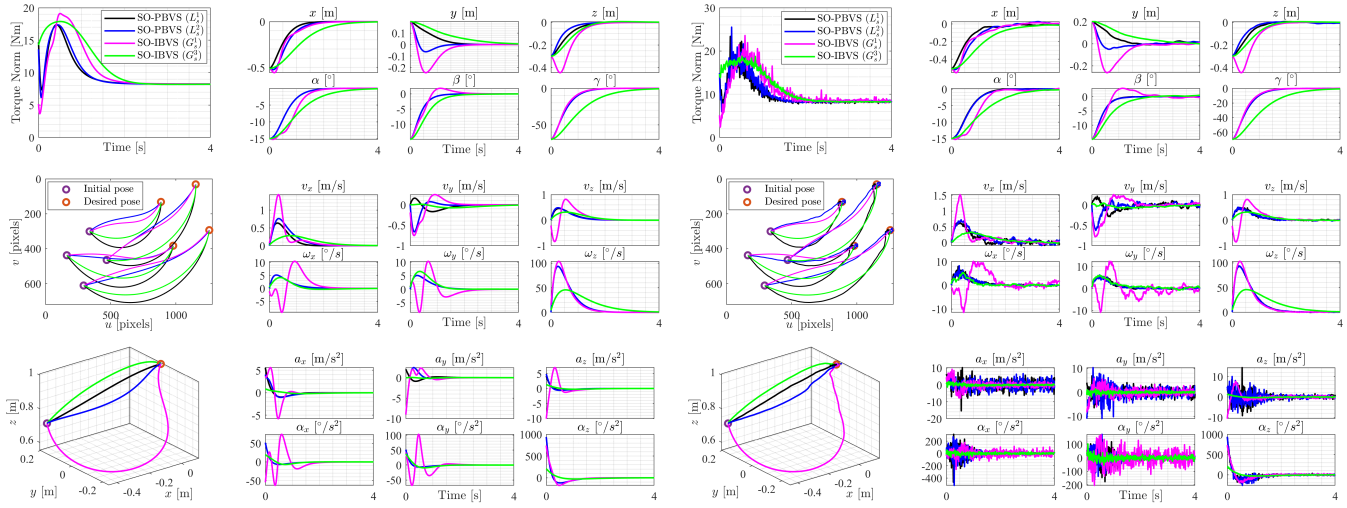

Fig. 3. Comparison of SO formulations for PBVS and IBVS. First column: torque norm (up), image plane (center) and 3D trajectory (bottom). Second column: error evolution (up), camera velocity (center) and camera acceleration (bottom). Third and fourth columns replicate results when noise is considered.

The simulated camera is defined by the intrinsic parameters of the Omnivision OV5640 embedded in the Gen3 robot: principal point o_x=620.9 and o_y=238.2 (in pixels) and focal length f_x=1297 and f_y=1298 (as multiple of pixel width and height). The resolution is set to 1280×720.

In simulation we assume the camera pose can be determined through encoder measurements on the robot and forward kinematics computations. To the calculated pose we add a random value drawn from a Gaussian distribution with zero mean and standard deviations σ_t=0.5cm for the position and σ_r=0.5° for the orientation. This is a good representation of the scenario where pose is estimated through an ArUco marker and the perspective-three-point (P3P) [22] algorithm, as leveraged with the real Gen3 robot.

For all three controllers, we require access to the current camera velocity $\mathbf{v_c}$, and for SO-PBVS, we also need access to $\dot{\mathbf{e}}_\mathbf{s}$ and $\dot{\mathbf{s}}$. We assume that $\mathbf{v_c}$ is readily available, and $\dot{\mathbf{e}}_\mathbf{s}$ and $\dot{\mathbf{s}}$ can be obtained through $\mathbf{L_s v_c}$.

The gains of the controllers were selected in order to achieve critically damped behavior and ensure that their error norm dynamics are as similar as possible. We set k_p and k_d of the SO-PBVS to 11.56 and 6.8 respectively, and set both k_p and μ of the FO-PBVS to 2.35, with $\gamma = 8\text{s}^{-1}$.

A. *Fixed configuration*

We first present the results for a fixed configuration, where $\dot{\mathbf{s}}^* = \ddot{\mathbf{s}}^* = 0$. As we choose gains that force controllers to behave similarly, the motivation for this experiment is to assess the influence of noise and show that acceleration command derived from velocity control can cause deviations in the error ideal evolution. The initial and final poses of the robot, as well as their representations as feature vectors, are presented in Table I and illustrated in Fig. 1. We consider the two variations of the interaction matrix, $\mathbf{L_s^1}$ (13) and $\mathbf{L_s^2}$ (15), and analyze the results separately. In the end, we compare the performances of SO-PBVS with variations of SO-IBVS.

1) FO-PBVS and SO-PBVS, using $\mathbf{L_s^1}$ *and* $\mathbf{L_s^2}$*:* Fig. 2 presents the results of this comparison in image and Cartesian spaces, the evolution of the camera position and orientation error, as well as the camera velocity and acceleration and robot joint torques norm. One notices little difference in performance

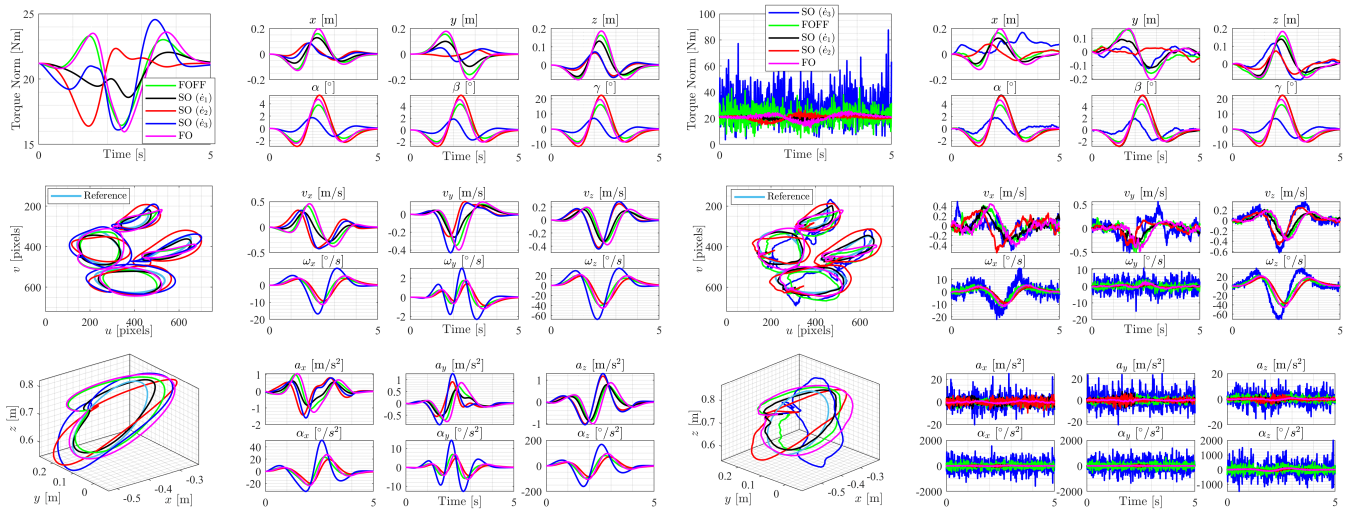

Fig. 4. Comparison for the tracking of a 3D trajectory with FO formulations and different estimations of $\dot{\mathbf{e}}$ for the SO case, using $\mathbf{L_s^2}$. First column: torque norm (up), image plane (center) and 3D trajectory (bottom). Second column: error evolution (up), camera velocity (center) and camera acceleration (bottom). Third and fourth columns replicate resuts when noise is considered.

TABLE I
INITIAL AND FINAL POSES CONSIDERED FOR SIMULATION. POSE AND $\Delta t|\Delta\theta$ ARE EXPRESSED IN $\mathcal{B}$

	Initial			Final			$\Delta t\|\Delta\theta$
	Pose	$\mathbf{s}$		Pose	$\mathbf{s}^*$		
		${}^{c^*}\mathbf{t}_c$	${}^{c}\mathbf{t}_{o}$		${}^{c^*}\mathbf{t}_c$	${}^{c^*}\mathbf{t}_{o}$	
Position (m)	-0.406	-0.321	0.418	0.106	0	-0.486	0.5
	0.206	0.363	-0.293	0.006	0	0.030	0.2
	0.700	-0.396	1.784	1.000	0	1.422	0.3
Orientation	-15°	q_1=-0.031		0°	q_1=0		15°
	0°	q_2=-0.031		15°	q_2=0		15°
	-50°	q_3=-0.577		20°	q_3=0		70°

between the controllers when using $\mathbf{L_s^1}$, although the FOFF maintains the straight-line trajectory in 3D space, guaranteed by this interaction matrix in the original PBVS.

When $\mathbf{L_s^2}$ is employed, the SO controller provided a more desirable trajectory in 3D space, while the FO had a worse performance. A deviation can be observed in the behavior of variable y for both FO formulations. While these deviations may arise from approximations in the derivative of the robot's Jacobian or in the computation of inverse kinematics, they provide insight into the impact of γ on the error dynamics, as indicated by the graph of v_y. This suggests that the artificial integration of the velocity command into torque control may not be optimal.

When noise is present, FOFF controllers suffer the most, and the torque norm is greatly affected, even at equilibrium, while FO and SO generate smoother control signals. Furthermore, the 3D trajectory and the image plane show that the FOFF fail to converge to the desired pose, displaying an error in the x component.

Noise is primarily influenced by pose estimation errors affecting the adjoint and Jacobian calculation in (23). This issue, however, is a common challenge in all VS systems that perform control in joint space. In our VS system, we generate camera acceleration in Cartesian space using (20), and this noise effect only arises when integrating the command into the robot's dynamic model for torque control. By sending the desired acceleration directly to the robot's internal controller, we can avoid this problem, as we demonstrated in the real experiments with Gen3.

2) SO-IBVS and SO-PBVS: For IBVS, we consider as features the image plane projections $\mathbf{s_i} = (x_i, y_i)$ of the four points of interest, with respect to the camera frame. In this case, the 2×6 interaction matrix of each point is given by

$$\mathbf{G_{s_i}} = \begin{bmatrix} \frac{-1}{Z} & 0 & \frac{x}{Z} & xy & -(1+x^2) & y \\ 0 & \frac{-1}{Z} & \frac{y}{Z} & 1+y^2 & -xy & -x \end{bmatrix}, \quad (29)$$

where Z is the depth of the point in camera frame. By stacking the four matrices we get the final 8×6 $\mathbf{G_s}$.

We consider three variations of $\mathbf{G_s}$, as defined in [3]. $\mathbf{G_s^1}$ is the matrix updated at each control iteration, with the current values of (x, y, Z); $\mathbf{G_s^2}$ is constant, where (x, y, Z) take the values of the desired camera pose; and $\mathbf{G_s^3}$ is such that $\mathbf{G_s^3}^+ = 1/2(\mathbf{G_s^1}+\mathbf{G_s^2})^+$. The acceleration is calculated with (20), using $\mathbf{L_s}=\mathbf{G_s}$ and $\mathbf{L_{s,\dot{s}}}=\mathbf{G_{s,\dot{s}}}$.

Fig. 3 shows the results obtained for this comparison. The best behaviors in the image domain are obtained with the SO-PBVS ($\mathbf{L_s^2}$) and the SO-IBVS ($\mathbf{G_s^1}$). However, the latter makes an undesirable 3D trajectory. In the 3D domain, SO-PBVS ($\mathbf{L_s^1}$) and SO-IBVS ($\mathbf{G_s^3}$) perform well, but in the latter, the convergence is slower, and using larger gains tends to deteriorate dynamic behavior. Results for the $\mathbf{G_s^2}$ deteriorates significantly and we do not present it. In the noisy scenario, with a 2-pixel standard deviation for IBVS, we can draw the same conclusions as in the noise-free case.

B. Tracking

Finally, we approach the scenario in which $\dot{\mathbf{s}}^*, \ddot{\mathbf{s}}^* \neq 0$. To achieve this, we take further inspiration from the comparison work by Fusco et al. [1] to define the features trajectory. Since we are dealing with 3D pose, we prefer to define this trajectory relative to the robot's base and not in the frame that defines $\mathbf{s}$, for ease of implementation. Therefore, we define a circular trajectory in 3D space, parameterized by:

$${}^b\mathbf{t}_c^*(t) = {}^b\mathbf{t}_c^*(0) + \lambda[\sin p(t) \quad \cos p(t) - 1 \quad \sin p(t)]^T, \quad (30)$$

$${}^b\boldsymbol{\theta}_c^*(t) = {}^b\boldsymbol{\theta}_c^*(0) + \sin p(t)[\alpha \quad \alpha \quad \beta]^T, \quad (31)$$

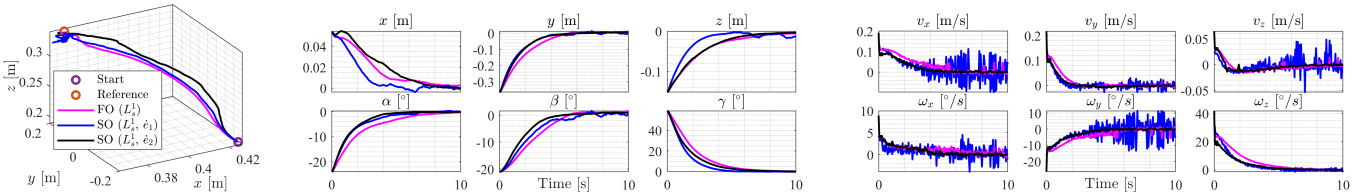

Fig. 5. Results for PBVS on the Kinova Gen3 with $\mathbf{L_s^1}$: Cartesian trajectory (left), error evolution (center) and velocity command (right).

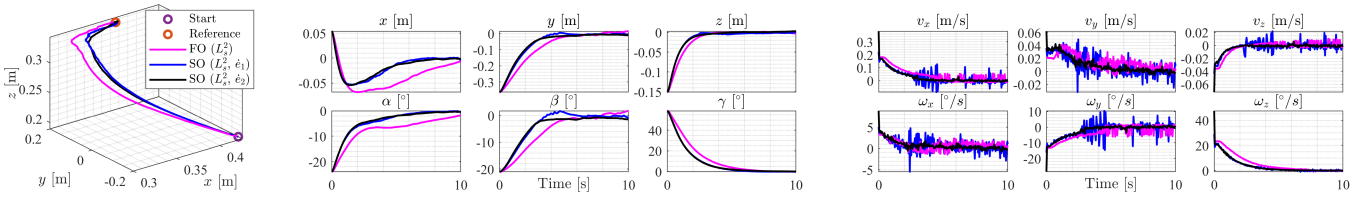

Fig. 6. Results for PBVS on the Kinova Gen3 with $\mathbf{L_s^2}$: Cartesian trajectory (left), error evolution (center) and velocity command (right).

where $p(t)$ is a quintic polynomial with $\dot{p}(0) = \dot{p}(t_f) = \ddot{p}(0) = \ddot{p}(t_f) = 0$, $t_f = 5$s, λ is 0.1m, α is 5º and β is 20º.

Considering $\mathbf{s}$ to be $({}^{c^*}\mathbf{t}_c, \mathbf{q})$, we would have $\mathbf{s}^*(t) = 0$ and then $\dot{\mathbf{s}}^*(t) = \ddot{\mathbf{s}}^*(t) = \mathbf{0}$, hence, we do not use $\mathbf{L_s^1}$ in this simulation. If we select $\mathbf{s} = ({}^c\mathbf{t}_o, \mathbf{q})$, then $\mathbf{s}^*(t) = ({}^{c^*}\mathbf{t}_o(t), \mathbf{0})$, therefore $\dot{\mathbf{s}}^*(t), \ddot{\mathbf{s}}^*(t) \neq 0$.

In addition, for the SO formulation, we examined three ways of measuring $\dot{\mathbf{e}}$, which is multiplied by k_d: $\dot{\mathbf{e}}_\mathbf{1} = \mathbf{L_s^2 v_c}$, $\dot{\mathbf{e}}_\mathbf{2} = \mathbf{L_s^2 v_c} - \dot{\mathbf{s}}^*$, and $\dot{\mathbf{e}}_\mathbf{3} = (\mathbf{e}_k - \mathbf{e}_{k-1})/\delta t$, with $\delta t = 0.05$s.

Results, presented in Fig. 4, reveal that the FOFF and particularly the FO formulations perform less effectively than all the SO variants for translation. There is only a slight advantage observed for FOFF and FO over the SO variations with $\dot{\mathbf{e}}_\mathbf{1}$ and $\dot{\mathbf{e}}_\mathbf{2}$ regarding orientation. For $\dot{\mathbf{e}}_\mathbf{1}$, where we employ the same $\dot{\mathbf{e}}$ estimation as for the fixed reference case, advantage of SO over FO and FOFF is minor, but it ensures a more accurate 3D trajectory as tracked by the camera.

When utilizing $\dot{\mathbf{e}}_\mathbf{2}$, the term $\dot{\mathbf{s}}^*$ anticipates changes in $\mathbf{s}^*$, leading to a significant drop in tracking errors for translational movement. The orientation error remains unchanged since $\mathbf{q}$ represents ${}^{c^*}\mathbf{R}_c$, *i.e.*, $\mathbf{q}^* = \dot{\mathbf{q}}^* = \mathbf{0}$. In this scenario, there is a deviation in the 3D trajectory, but the overall error is substantially reduced due to the reduction in velocity error.

Using numerical estimation of the error derivative, $\dot{\mathbf{e}}_\mathbf{3}$, guarantees the lowest tracking error because it directly addresses changes in orientation error. The introduction of this derivative generates undesirable behavior in the noisy conditions, as shown in the acceleration, velocity, torque and 3D trajectory graphs.

In summary, the SO formulation with $\dot{\mathbf{e}}_\mathbf{3}$ consistently delivers superior tracking performance in noise-free scenarios. On the other hand, the SO formulation with $\dot{\mathbf{e}}_\mathbf{2}$ stands out as the optimal compromise, performing well in scenarios both with and without noise.

IV. EXPERIMENTAL RESULTS

The simulation experiments included the dynamic model of the robot with fixed gains for the controllers. The experiments performed on the real robot are intended to evaluate the behavior in the image and 3D spaces when PBVS is used at a higher level. In this scenario, the FO-PBVS generates the velocity signal using (25), and the SO-PBVS generates an acceleration signal using (20), which is integrated and sent to the low-level internal controller of the robot as a velocity command. The same idea is explored in real-world testing in the context of SO-IBVS [14], [8].

TABLE II
INITIAL AND FINAL POSES CONSIDERED FOR REAL EXPERIMENTS. POSE AND $\Delta t|\Delta\theta$ ARE EXPRESSED IN $\mathcal{B}$

	Initial			Final			$\Delta t\|\Delta\theta$
	Pose	$\mathbf{s}$		Pose	$\mathbf{s}^*$		
		${}^{c^*}\mathbf{t}_c$	${}^{c}\mathbf{t}_o$		${}^{c^*}\mathbf{t}_c$	${}^{c^*}\mathbf{t}_o$	
Position (m)	0.426	-0.363	0.090	0.372	0	-0.083	-0.054
	-0.180	0.054	0.099	0.183	0	0.018	0.363
	0.189	0.151	0.321	0.340	0	0.355	0.151
Orientation	156°	q_1=-0.090		180°	q_1=0		24°
	-21°	q_2=0.255		0°	q_2=0		21°
	149°	q_3=-0.440		90°	q_3=0		-59°

We used the same robot, Kinova Gen3, with a Python interface through the Kortex API. We selected an ArUco marker as the object of interest, positioned at $(0.39\text{m}, 0.10\text{m}, -0.01\text{m}, 0°, 0°, 180°)$ with respect to the robot base, and pose estimation is performed using the ArUco library [23] from OpenCV [24]. The 6D pose is computed through the P3P algorithm [22] using the pixel coordinates of the marker's vertices, the camera's intrinsic parameters, and additional features of the marker, such as its width.

The camera's initial and desired 6D poses are presented in Table II. Note that, as in the simulation, the camera performs large linear and angular displacements. We do not add noise to the measurements as the visual pose estimation is already subject to various imaging conditions, and velocity estimation contains noise from the encoders. The error derivative for SO is estimated with both $\dot{\mathbf{e}}_\mathbf{1} = (\mathbf{e}_k - \mathbf{e}_{k-1})/\delta t$, with $\delta t \approx 0.05$s, and $\dot{\mathbf{e}}_\mathbf{2} = \mathbf{L_s^2 v_c}$.

Fig. 5 shows the camera trajectory in 3D space, the error evolution, and the velocity command for the control with $\mathbf{L_s^1}$, and Fig. 7 shows the trajectory of the ArUco vertices. We consider several gains and select those that lead to the fastest convergence, ensuring the ArUco remains in the camera's field of view. For FO, k_p=0.4, for SO ($\dot{\mathbf{e}}_\mathbf{1}$), k_p=6/k_d=0.5 and for SO ($\dot{\mathbf{e}}_\mathbf{2}$), k_p=4.5/k_d=0.5. Considering $\mathbf{L_s^2}$, results are shown in Fig. 6 and Fig. 8. We set k_p=0.4 for FO and k_p=5/k_d=0.5 for both SO ($\dot{\mathbf{e}}_\mathbf{1}$) and SO ($\dot{\mathbf{e}}_\mathbf{2}$).

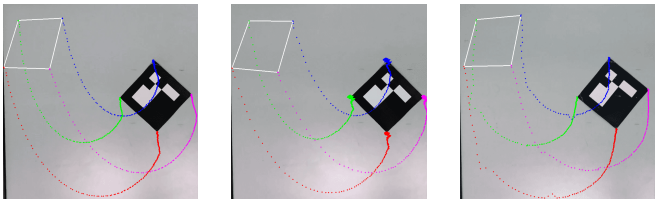

Fig. 7. Trajectory of ArUco vertices during control with $\mathbf{L_s^1}$. The white polygon shows how the camera perceived the marker in the initial pose. FO is depicted on the left, SO ($\dot{\mathbf{e}}_\mathbf{1}$) on the center, and SO ($\dot{\mathbf{e}}_\mathbf{2}$) on the right.

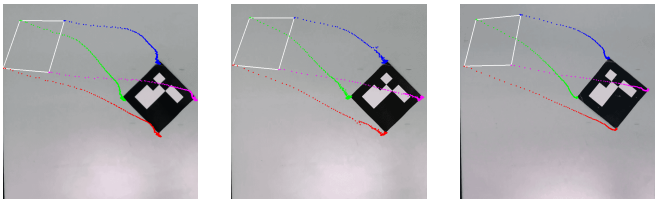

Fig. 8. Trajectory of ArUco vertices during control with $\mathbf{L_s^2}$. The white polygon shows how the camera perceived the marker in the initial pose. FO is depicted on the left, SO ($\dot{\mathbf{e}}_\mathbf{1}$) on the center, and SO ($\dot{\mathbf{e}}_\mathbf{2}$) on the right.

It can be seen that the SO-PBVS converges to the desired pose faster for both interaction matrices. For the $\mathbf{L_s^1}$ case, the combination of the PD controller decreases the overshoot in the image, enabling higher gains, while the FO controller is limited to a maximum k_p of 0.4. After convergence, however, the SO ($\dot{\mathbf{e}}_\mathbf{1}$) has oscillatory behavior in Cartesian space, due to pose estimation errors, which is magnified by the nature of $\dot{\mathbf{e}}_\mathbf{1}$. Considering $\mathbf{L_s^2}$, the ArUco trajectory does not limit gains tuning, so convergence time has minor relevance, but SO formulations do guarantee a better 3D trajectory.

Using SO ($\dot{\mathbf{e}}_\mathbf{2}$), we can strike the right balance between fast, precise convergence and robustness to noise. Since the robot's internal controller is finely tuned to exactly track velocity commands [2], it can lead to vibrations in the robot when confronted with sudden velocity changes caused by pose estimation noise. Second-order controllers effectively filter out noise because the robot's states are linked to the integral of the control signal. While SO ($\dot{\mathbf{e}}_\mathbf{1}$) does not adhere to this pattern due to extra differentiation, SO ($\dot{\mathbf{e}}_\mathbf{2}$) produces smooth velocity commands, ensuring smooth convergence.

V. CONCLUSIONS

In this letter, we presented a reformulation for the classical interaction matrices of PBVS, achieved by changing the orientation portion of the feature vector to quaternion. We further applied those matrices to define a second-order PBVS, which was broadly evaluated and compared against both first-order PBVS and second-order IBVS controllers.

The experiments demonstrate that the dynamic properties of SO-PBVS tend to be better than those of FO-PBVS regarding convergence time and Cartesian trajectories. We also lay some groundwork for the choice between SO-PBVS and SO-IBVS by comparing their behavior in Cartesian and image spaces. In future work, we aim to integrate the SO-PBVS into a model-based reinforcement learning framework to assess robustness and generalization in visual servoing.

REFERENCES

[1] F. Fusco, O. Kermorgant, and P. Martinet, "A comparison of visual servoing from features velocity and acceleration interaction models," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2019, pp. 4447–4452.

[2] M. Keshmiri, W.-F. Xie, and A. Mohebbi, "Augmented image-based visual servoing of a manipulator using acceleration command," *IEEE Trans. Ind. Electron.*, vol. 61, no. 10, pp. 5444–5452, 2014.

[3] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robot. Autom. Mag.*, vol. 13, pp. 82–90, 2006.

[4] A. V. Kudryavtsev, M. T. Chikhaoui, A. Liadov, P. Rougeot, F. Spindler, K. Rabenorosoa, J. Burgner-Kahrs, B. Tamadazte, and N. Andreff, "Eye-in-hand visual servoing of concentric tube robots," *IEEE Rob. and Automation Letters*, vol. 3, no. 3, pp. 2315–2321, 2018.

[5] A. Paolillo, K. Chappellet, A. Bolotnikova, and A. Kheddar, "Interlinked visual tracking and robotic manipulation of articulated objects," *IEEE Rob. and Automation Letters*, vol. 3, no. 4, pp. 2746–2753, 2018.

[6] J. Pomares, I. Perea, and F. Torres, "Dynamic visual servoing with chaos control for redundant robots," *IEEE/ASME Trans. Mechat.*, vol. 19, no. 2, pp. 423–431, 2013.

[7] S. Vandernotte, A. Chriette, P. Martinet, and A. S. Roos, "Dynamic sensor-based control," in *14th Int. Conf. Cont. Autom. Robot. Vision*, 2016, pp. 1–6.

[8] 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.

[9] T.-Y. Chang, W.-C. Chang, M.-Y. Cheng, and S.-S. Yang, "Dynamic visual servoing with kalman filter-based depth and velocity estimator," *Int. J. Adv. Robot. Syst.*, vol. 18, no. 3, p. 17298814211016674, 2021.

[10] H. Zhang and J. P. Ostrowski, "Visual servoing with dynamics: Control of an unmanned blimp," in *Proc. IEEE Int. Conf. Robot. Autom. (Cat. No. 99CH36288C)*, vol. 1, 1999, pp. 618–623.

[11] H. Fakhry and W. Wilson, "A modified resolved acceleration controller for position-based visual servoing," *Mathematical and computer modelling*, vol. 24, no. 5-6, pp. 1–9, 1996.

[12] R. Dahmouche, N. Andreff, Y. Mezouar, O. Ait-Aider, and P. Martinet, "Dynamic visual servoing from sequential regions of interest acquisition," *The Int. Journal Robot. Res.*, vol. 31, no. 4, pp. 520–537, 2012.

[13] Y.-L. Kuo and S.-C. Tang, "Dynamics and control of a 3-dof planar parallel manipulator using visual servoing resolved acceleration control," *J. Low Freq. Noise Vibr. Act. Cont.*, vol. 40, pp. 458–480, 2021.

[14] X. Zhang, R. Wang, Y. Fang, B. Li, and B. Ma, "Acceleration-level pseudo-dynamic visual servoing of mobile robots with backstepping and dynamic surface control," *IEEE Trans. Syst. Man Cyber.: Syst.*, vol. 49, no. 10, pp. 2071–2081, 2017.

[15] G. Hu, W. E. Dixon, S. Gupta, and N. Fitz-Coy, "A quaternion formulation for homography-based visual servo control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 2391–2396.

[16] X. Zhao, Z. Xie, H. Yang, and J. Liu, "Minimum base disturbance control of free-floating space robot during visual servoing pre-capturing process," *Robotica*, vol. 38, no. 4, pp. 652–668, 2020.

[17] F. Aghili, "Fault-tolerant and adaptive visual servoing for capturing moving objects," *IEEE/ASME Trans. Mechat.*, vol. 27, no. 3, pp. 1773–1783, 2022.

[18] C. De Farias, M. Adjigble, B. Tamadazte, R. Stolkin, and N. Marturi, "Dual quaternion-based visual servoing for grasping moving objects," in *IEEE 17th Int. Conf. Autom. Sci. Engin.*, 2021, pp. 151–158.

[19] R. Saltus, I. Salehi, G. Rotithor, and A. P. Dani, "Dual quaternion visual servo control," in *IEEE 59th Conference on Decision and Control*, 2020, pp. 5956–5961.

[20] W. E. Dixon, A. Behal, D. M. Dawson, and S. P. Nagarkatti, *Nonlinear control of engineering systems: a Lyapunov-based approach*. Boston, MA: Birkhäuser, 2003.

[21] E. G. Ribeiro, R. Q. Mendes, M. H. Terra, and V. Grassi, "Dynamic parameter identification of a 7-dof lightweight robot manipulator using probabilistic differential optimization," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2022, pp. 1917–1923.

[22] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Trans. Patt. Anal. Mach. Intel.*, vol. 25, no. 8, pp. 930–943, 2003.

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

[24] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.