A Comparison of Visual Servoing from Features Velocity and Acceleration Interaction Models

Franco Fusco[1], Olivier Kermorgant[1] and Philippe Martinet[1,2]

Abstract— Visual Servoing has been widely investigated in the last decades as it provides a powerful strategy for robot control. Thanks to the direct feed-back from a set of sensors, it allows to reduce the impact of some modeling errors and to perform tasks even in uncertain environments. The commonly exploited approach in this field is to use a model that expresses the rate of change of a set of features as a function of sensor twist. These schemes are commonly used to obtain a velocity command, which needs to be tracked by a low-level controller.

Another approach that can be exploited consists in going one step further and to consider an acceleration model for the features. This strategy allows also to obtain a natural and direct link with the dynamic model of the controlled system. This study aims at comparing the use of velocity and acceleration-based models in feed-back linearization for Visual Servoing. We consider the case of a redundant manipulator and discuss what this implies for both control techniques. By means of simulations, we show that controllers based on features acceleration give better results than those based on velocity in presence of noisy feedback signals.

I. Introduction

Visual Servoing is a control technique that allows precise relative positioning between a sensor and an observed object. It is robust to modeling errors and features a large convergence domain. The main research interests have been on modeling the interaction between descriptive features and sensor motion at the velocity level. Controllers are usually designed in two decoupled stages: a first high-level layer operates in the features space and produces a reference velocity [1], [2] which is then forwarded to a low-level robot controller actuating the motors to track the desired velocity profile [3], [4], [5].

In recent years, the idea of directly linking the dynamics of robotics systems to sensory feedback has been considered. This had already been proposed over than two decades ago [3], [6], with the appealing option of using a second-order model expressing the link between features acceleration and robot kinematics, altogether with the Dynamic model of the robot. This allows to define, *e.g.*, Computed Torque Controllers directly in the feature space. These models have now been "rediscovered" and used in different contexts, such as vision-based aerial grasping [7], stero-vision systems [8] and hybrid vision-force control for serial manipulators [9].

In this work, we contribute to the analysis of second-order Visual Servoing control schemes by performing a systematic comparison with classical first-order controllers. In the recent literature, second order models have been mainly derived *ad*

[1]Centrale Nantes, Laboratoire des Sciences du Numérique de Nantes LS2N, France. Mail: name.surname@ls2n.fr

[2]Inria Sophia Antipolis, France. Mail: philippe.martinet@inria.fr

hoc from the definition of considered features. We instead consider a more general approach, which does not rely on the selected set of descriptors. While this concept had been already described briefly in [3], we contribute with more explicit results in the development.

In addition, we consider the case of a redundant manipulator featuring 7 degrees of freedom. Our objective in this sense is to focus the attention on the fact that second-order schemes must explicitly handle the redundancy even when no additional tasks are specified, whereas first-order controllers naturally cope with this added complexity.

The remainder of this paper is organized as follows: in Section II we recall basic modeling exploited in classical Visual Servoing and illustrate how acceleration models can be obtained. We then review how to link robot kinematics and dynamics to the feature space, and how to use these models to design controllers. We also discuss the implications of redundancy in the proposed control laws. In Section III we compare in a simulated environment control laws based on the first and second-order models. Reported results show that models based on features acceleration outperform those exploiting the cascade of Visual and low-level controllers. In particular, they exhibit good convergence properties and smoother control inputs, with less sensitivity to noise.

II. Visual Servoing & Dynamics

A. First order Visual Servoing

We recall here the classical first-order modeling approach generally exploited in the context of Visual Servoing. To achieve the task of regulating the relative position between a sensor and an observed object, a set of descriptive features is firstly selected. Their numerical value is stored in the m-dimensional vector $\mathbf{s}$. This set might include coordinates extracted from an image, the relative pose between the sensor frame and the object frame, distance measurements, *etc*. We will focus in the following on the case of Image Based Visual Servoing (IBVS), although the models presented below are generic and can be applied to a broader class of features and sensors.

Once the features have been chosen, it is necessary to relate them to the kinematics of the robotic system. This is achieved via the following differential relation:

$$\dot{\mathbf{s}} = \mathbf{L_s}\,\mathbf{v} + \frac{\partial \mathbf{s}}{\partial t} \tag{1}$$

$\mathbf{L_s}$ being the *Interaction Matrix* of the set of features and $\mathbf{v}^T = \begin{bmatrix} \boldsymbol{v}^T & \boldsymbol{\omega}^T \end{bmatrix}$ representing the kinematic screw, or twist, of the camera with respect to the object (expressed in the

coordinate system of the sensor itself). The last term in (1) takes into account variations of the features that do not depend on the relative motion between the sensor and the object. It is generally possible to assume that additive component to be null.

It is worth remarking that $\mathbf{L_s}$ generally depends on both the observed features and a set of parameters. Among these, some can be assumed constant, such as intrinsic camera parameters. On the other side, there appear also quantities in $\mathbf{L_s}$ whose value changes depending on the relative motion between the camera and the object. Examples of such parameters are the depth of image points, or plane parameters when dealing with planar image moments. In the following, the vector containing such parameters is denoted as $\mathbf{z}$.

Starting from first-order models, it is shown in the next section how to obtain a relation that involves the second derivative of the features and the acceleration of the sensor.

B. Second order Visual Servoing

This class of models considers the second-order time derivative of the features and how it depends on the acceleration of sensors, in addition to their velocity. In related works the derivation of the second order model was achieved by considering the direct expression of the features and by evaluating explicitly $\ddot{\mathbf{s}}$ depending on its definition. We propose in the following a more general approach that we believe facilitates the derivation of these models. First of all, the derivative of (1) gives (assuming $\frac{\partial \mathbf{s}}{\partial t} = \mathbf{0}$):

$$\ddot{\mathbf{s}} = \mathbf{L_s} \frac{\mathrm{d}}{\mathrm{d}t} \mathbf{v} + \dot{\mathbf{L}}_\mathbf{s} \, \mathbf{v} \qquad (2)$$

The derivative of the kinematic screw writes as follows:

$$\frac{\mathrm{d}}{\mathrm{d}t} \mathbf{v} = \mathbf{a} + \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{\omega} \\ \mathbf{0} \end{bmatrix} \qquad (3)$$

which is obtained by considering the camera frame is not inertial with respect to the object. Regarding the differentiation of $\mathbf{L_s}$, it is convenient to consider this matrix as written in the form:

$$\mathbf{L_s} = \begin{bmatrix} \boldsymbol{\ell}_1 & \cdots & \boldsymbol{\ell}_m \end{bmatrix}^T \qquad (4)$$

$\boldsymbol{\ell}_i$ being the rows of $\mathbf{L_s}$, from which the derivative can be expressed as:

$$\dot{\mathbf{L}}_\mathbf{s} = \begin{bmatrix} \left(\frac{\partial \boldsymbol{\ell}_1}{\partial \mathbf{s}} \dot{\mathbf{s}} + \frac{\partial \boldsymbol{\ell}_1}{\partial \mathbf{z}} \dot{\mathbf{z}} \right)^T \\ \cdots \\ \left(\frac{\partial \boldsymbol{\ell}_m}{\partial \mathbf{s}} \dot{\mathbf{s}} + \frac{\partial \boldsymbol{\ell}_m}{\partial \mathbf{z}} \dot{\mathbf{z}} \right)^T \end{bmatrix} \qquad (5)$$

For simplicity we assume in the following that an expression for $\dot{\mathbf{z}}$ is available, and that furthermore it takes the form $\dot{\mathbf{z}} = \mathbf{L_z} \mathbf{v}$. Under this assumption and by introducing $\mathbf{H}_i \doteq \left(\frac{\partial \boldsymbol{\ell}_i}{\partial \mathbf{s}} \mathbf{L_s} + \frac{\partial \boldsymbol{\ell}_i}{\partial \mathbf{z}} \mathbf{L_z} \right)^T$:

$$\dot{\mathbf{L}}_\mathbf{s} = \begin{bmatrix} \mathbf{v}^T \mathbf{H}_1 \\ \cdots \\ \mathbf{v}^T \mathbf{H}_m \end{bmatrix} \qquad (6)$$

As a result, (2) becomes:

$$\ddot{\mathbf{s}} = \mathbf{L_s} \mathbf{a} + \mathbf{L_s} \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{\omega} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{v}^T \mathbf{H}_1 \\ \cdots \\ \mathbf{v}^T \mathbf{H}_m \end{bmatrix} \mathbf{v} \qquad (7)$$

The second term in the right-hand side can be rearranged in a nicer form that matches the structure of the last term. For this purpose, one can consider the first three elements in each row of $\mathbf{L_s}$ and place them in a set of vectors $\boldsymbol{\ell}_i^{(v)}$. After some simple manipulation, it is possible to rewrite (2) in the form:

$$\ddot{\mathbf{s}} = \mathbf{L_s} \mathbf{a} + \begin{bmatrix} \mathbf{v}^T \left(\mathbf{H}_1 + \boldsymbol{\mathcal{L}}_1 \right) \mathbf{v} \\ \cdots \\ \mathbf{v}^T \left(\mathbf{H}_m + \boldsymbol{\mathcal{L}}_m \right) \mathbf{v} \end{bmatrix} = \mathbf{L_s} \mathbf{a} + \mathbf{h_s} \qquad (8)$$

wherein matrices $\boldsymbol{\mathcal{L}}_i$ take the form:

$$\boldsymbol{\mathcal{L}}_i = \begin{bmatrix} \mathbf{O}_3 & \mathbf{O}_3 \\ \left[\boldsymbol{\ell}_i^{(v)} \right]_\times & \mathbf{O}_3 \end{bmatrix} \qquad (9)$$

having used the symbols $\mathbf{O}_3$ and $[\cdot]_\times$ to denote respectively the null 3-by-3 matrix and the skew-symmetric matrix corresponding to the cross-product.

The main interest in the given development is that to get the second order model of any feature, the only required work is to evaluate $\frac{\partial \boldsymbol{\ell}_i}{\partial \mathbf{s}}$ and $\frac{\partial \boldsymbol{\ell}_i}{\partial \mathbf{z}}$.

We conclude this section by showing how a second order model is obtained from another set of features for which an acceleration model has already been obtained. Consider a set of features $\mathbf{s}_a$ defined by some differentiable function of another set of features $\mathbf{s}_b$, *i.e.*, $\mathbf{s}_a = \mathbf{f}(\mathbf{s}_b)$. After development, one would get the following result:

$$\ddot{\mathbf{s}}_a = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{s}_b} \mathbf{L}_b \right) \mathbf{a} + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{s}_b} \mathbf{h}_b + \left(\frac{\mathrm{d}}{\mathrm{d}t} \frac{\partial \mathbf{f}}{\partial \mathbf{s}_b} \right) \mathbf{L}_b \mathbf{v} \right) \qquad (10)$$

in which it is easy to identify the Interaction Matrix, multiplying the acceleration screw $\mathbf{a}$, and the vector $\mathbf{h}_a$.

C. Robot Kinematics and Dynamics in Sensor Space

If we assume that the relative motion between the sensor and the object depends solely on the motion of the robot, the models presented in the previous section can be easily linked to the configuration space of the robotic system. Assuming an *eye-in-hand* setup, the velocity and acceleration of the sensor are given by:

$$\mathbf{v} = \mathbb{T} \mathbf{J} \dot{\mathbf{q}} \qquad (11a)$$
$$\mathbf{a} = \mathbb{T} \mathbf{J} \ddot{\mathbf{q}} + \mathbb{T} \dot{\mathbf{J}} \dot{\mathbf{q}} \qquad (11b)$$

wherein $\mathbf{J}$ is the 6-by-n Jacobian matrix of the robot, n being the number of degrees of freedom of the robot. The two vectors $\mathbf{q}, \dot{\mathbf{q}} \in \mathbb{R}^n$ collect respectively positions and velocities of the joints, while $\mathbb{T}$ is a twist-rotation matrix that projects the kinematic screw on the sensor frame.

By injecting (11a) and (11b) into the first and second order models respectively, the following kinematic relations are obtained[1]:

$$\dot{\mathbf{s}} = \mathbf{L_s} \mathbb{T} \mathbf{J} \dot{\mathbf{q}} = \mathbf{J}_\mathbf{s} \dot{\mathbf{q}} \qquad (12a)$$
$$\ddot{\mathbf{s}} = \mathbf{J}_\mathbf{s} \ddot{\mathbf{q}} + \mathbf{L_s} \mathbb{T} \dot{\mathbf{J}} \dot{\mathbf{q}} + \mathbf{h_s} = \mathbf{J}_\mathbf{s} \ddot{\mathbf{q}} + \mathbf{h_q} \qquad (12b)$$

where the matrix $\mathbf{J}_\mathbf{s} = \mathbf{L_s} \mathbb{T} \mathbf{J}$ is known as *feature Jacobian*.

We also recall here that the Inverse Dynamic Model (IDM) of a robotic system composed of a set of rigid bodies writes

---

[1]Obviously, (12b) can be obtained by differentiation of (12a) as well.

as:

$$\boldsymbol{\tau} = \mathbf{M}\ddot{\mathbf{q}} + \mathbf{b} \qquad (13)$$

wherein $\boldsymbol{\tau}$ represents joint efforts while $\mathbf{M}$ is the generalized inertia matrix of the system, grouping inertial effects due to masses and inertiae of rigid bodies and motors. Finally, $\mathbf{b}$ groups Coriolis and centripetal forces, the efforts exerted on the system by gravity and the forces and moments acting on the actuators due to friction.

Assuming that $\mathbf{M}$ is invertible, which is always true for a serial kinematic chain, the joint accelerations are given by $\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\boldsymbol{\tau} - \mathbf{b})$, thus leading to:

$$\mathbf{J_s}\mathbf{M}^{-1}\boldsymbol{\tau} = \ddot{\mathbf{s}} - \mathbf{h_q} + \mathbf{J_s}\mathbf{M}^{-1}\mathbf{b} \qquad (14)$$

This last equation provides a direct link between the commanded joint efforts and the acceleration of features.

D. Visual Servoing

Exploiting the illustrated kinematic and dynamic models, it is possible to build a variety of controllers that steer the observed features to a desired value $\mathbf{s}^\star$.

The objective of the control is to regulate the feature error $\mathbf{e_s} \doteq \mathbf{s} - \mathbf{s}^\star$ to zero. A proportional control with feed-forward can be applied when considering the first order model as:

$$\dot{\mathbf{q}}_1^\star = \widehat{\mathbf{J_s^+}}(\dot{\mathbf{s}}^\star - \lambda\mathbf{e_s}) \qquad (15)$$

whereas a proportional and derivative one can be exploited if the second order model is used:

$$\ddot{\mathbf{q}}_2^\star = \widehat{\mathbf{J_s^+}}\left(\ddot{\mathbf{s}}^\star - k_d\dot{\mathbf{e}}_\mathbf{s} - k_p\mathbf{e_s} - \widehat{\mathbf{h_q}}\right) \qquad (16)$$

In both cases, $\widehat{\mathbf{J_s^+}}$ represents an estimation of the Moore-Penrose pseudo-inverse of $\mathbf{J_s}$. Using (16) also requires an estimation of $\mathbf{h_q}$ and $\dot{\mathbf{s}}$. Finally, the desired set of features is often constant, *i.e.*, $\dot{\mathbf{s}}^\star = \mathbf{0}$ and $\ddot{\mathbf{s}}^\star = \mathbf{0}$, and therefore the controllers are slightly simplified.

The two reference commands correspond to the minimum norm-solutions of the respective minimization problems:

$$\arg\min_{\dot{\mathbf{q}}}\left\|\widehat{\mathbf{J_s}}\dot{\mathbf{q}} - \dot{\mathbf{s}}^\star + \lambda\mathbf{e_s}\right\| \qquad (17a)$$

$$\arg\min_{\ddot{\mathbf{q}}}\left\|\widehat{\mathbf{J_s}}\ddot{\mathbf{q}} - \ddot{\mathbf{s}}^\star + k_d\dot{\mathbf{e}}_\mathbf{s} + k_p\mathbf{e_s} + \widehat{\mathbf{h_q}}\right\| \qquad (17b)$$

They aim at enforcing an exponential decrease of the error according to the evolution of the autonomous linear systems $\dot{\mathbf{e}}_\mathbf{s} + \lambda\mathbf{e_s} = \mathbf{0}$ and $\ddot{\mathbf{e}}_\mathbf{s} + k_d\dot{\mathbf{e}}_\mathbf{s} + k_p\mathbf{e_s} = \mathbf{0}$, both asymptotically stable if positive gains are selected. Perfect linearization will not be achievable in general, but asymptotic stability is ensured for (15) provided that $\mathbf{J_s}\widehat{\mathbf{J_s^+}} > 0$ [1]. Additional conditions need to be satisfied for the stability of (16) [10].

The output obtained from these kinematic control laws can be finally used in conjunction with the IDM to evaluate a set of effort commands to be sent to the actuators. For the first kind of regulation, the provided reference is a joint velocity command. In order to exploit the dynamics, a possible choice is to add another control layer providing the auxiliary acceleration signal

$$\mathbf{w}_1 = \ddot{\mathbf{q}}_{ff} + \gamma(\dot{\mathbf{q}}_1^\star - \dot{\mathbf{q}}) \qquad (18)$$

given a positive proportional gain γ and the feed-forward term $\ddot{\mathbf{q}}_{ff}$ corresponding to the numerical derivative of (15). This auxiliary command is finally inserted in the IDM, thus giving:

$$\boldsymbol{\tau} = \mathbf{M}\mathbf{w}_1 + \mathbf{b} \qquad (19)$$

Regarding (16), the produced output can be used directly to compute the efforts from the IDM.

E. Dealing with Redundancy

We address separately here the case in which the robot has more degrees of freedom than those required to perform the servoing task. Assuming that no singularities are encountered, we can state that the robot is redundant if $n > \mathrm{rank}(\mathbf{J_s})$. In this case, as the kernel of the matrix is not null, infinite solutions can be found that are compatible with (17a) and (17b). As outlined before, (15) and (16) give minimum norm solutions for the joint velocity and acceleration respectively.

Other controls which are compatible with the second-order Visual Servoing task can be found by using directly (14) to compute the efforts from the desired task [9], [7]. Two possible choices in this sense are to minimize $\|\boldsymbol{\tau}\|$ or a weighted norm of the efforts $\boldsymbol{\tau}^T\mathbf{D}\boldsymbol{\tau}$ (with $\mathbf{D} > 0$):

$$\boldsymbol{\tau} = \left(\mathbf{J_s}\mathbf{M}^{-1}\right)^+ \left(\ddot{\mathbf{s}}^\star - k_d\dot{\mathbf{e}}_\mathbf{s} - k_p\mathbf{e_s} - \mathbf{h_q} + \mathbf{J_s}\mathbf{M}^{-1}\mathbf{b}\right) \qquad (20a)$$

$$\boldsymbol{\tau} = (\mathbf{J_s})_\mathbf{D}^+ \left(\ddot{\mathbf{s}}^\star - k_d\dot{\mathbf{e}}_\mathbf{s} - k_p\mathbf{e_s} - \mathbf{h_q} + \mathbf{J_s}\mathbf{M}^{-1}\mathbf{b}\right) \qquad (20b)$$

wherein $(\mathbf{J_s})_\mathbf{D}^+$ represents a weighted pseudo-inverse. Regarding these two possibilities, it must be noted that the latter should generally be preferred, as the former might lead to high uncontrolled velocities [11].

Even though in this work we are interested only in a single task, it must be noted that, when dealing with a redundant manipulator, schemes based on the second-order model will not be enough to stabilize the robot to a given configuration. A classical first-order control, without supplementary tasks, will return the minimum norm joint velocity required to reach an equilibrium configuration. Regardless this being a global or local minimum, after the convergence the computed command will be zero, and therefore the robot will not move anymore (we do not consider perturbations here). The same is not true for the other class of controllers. Indeed, at equilibrium the command corresponds to a null acceleration, which is compatible with a non-null instantaneous velocity lying in the kernel of $\mathbf{J_s}$. It is thus necessary to ensure the given control law damps the velocity of the robot.

A possible solution is to add the following secondary task:

$$\ddot{\mathbf{q}} = -k_v\dot{\mathbf{q}} \qquad (21)$$

for a positive gain k_v. The final control law would thus become:

$$\ddot{\mathbf{q}}_2^{\star\prime} = \ddot{\mathbf{q}}_2^\star - \mathbf{Z}\mathbf{Z}^T\left(k_v\dot{\mathbf{q}} + \ddot{\mathbf{q}}_2^\star\right) \qquad (22)$$

with $\ddot{\mathbf{q}}_2^\star$ given as before by (16). $\mathbf{Z}$ is an orthogonal basis of the kernel of $\mathbf{J_s}$, obtainable via, *e.g.*, Complete Orthogonal Decomposition, so that the orthogonal projector is given in

the efficient form $\mathbf{Z}\mathbf{Z}^T$ [12]. The efforts can then be obtained by injecting into (13) $\ddot{\mathbf{q}}_2^{\star\prime}$ instead of $\ddot{\mathbf{q}}_2^\star$.

As a concluding remark, it might be considered to inject the velocity damping task into (20a) or (20b), *i.e.*, solving the problem directly at the effort level. However, the Servoing task altogether with the velocity damping objective are fully constraining the degrees of freedom of the robot, and, as direct consequence, the final result would not change. On the contrary, as these models require more operations, *e.g.*, the inversion of $\mathbf{M}$, in all our experiments they are not considered.

III. Simulation Results

A comparison between classical first-order control schemes and second order ones has been performed in a simulated environment. We report in the following the details of our implementation and the results of this study.

Tests have been realized using Gazebo for physics simulation and ROS for the implementation of controllers and communication. ViSP [13] has been used for Visual Servoing related computations.

Our setup involves a positioning task from four coplanar points located at the vertices of a square. We consider a perspective camera mounted on the tip of a Kuka LWR4+ robot, a serial manipulator featuring 7 degrees of freedom. We assume a perfectly calibrated sensor with no distortion, and use directly the normalized coordinates of the observed points. We also do not consider uncertainties in the Geometric and Dynamic parameters of the robot.

The control period in simulation is set to $1\,\mathrm{ms}$. We consider that the visual feed-back comes from a mixture of image processing and state estimation algorithm, and that the unknown parameters $\mathbf{z}$ can be estimated as well [14], [15]. To take into account uncertainties due to these operations, we inject normally distributed noise in both $\mathbf{s}$ and $\mathbf{z}$. In addition, the presence of encoders on each motor is simulated by quantization of the readings. More details on noise are presented in the next section.

Three control strategies are analyzed. The first one, namely *First-Order* (FO), corresponds to a velocity-based scheme without the feed-forward signal $\ddot{\mathbf{q}}_{ff}$. The second strategy takes this term into account and will be referred to as *First-Order with Feed-Forward* (FOFF). Finally, a *Second-Order* (SO) scheme that evaluates joints acceleration via (16) is considered. Implementation and tuning of these controllers are discussed in Section III-B

A. Noise

Noise is introduced in the simulation to obtain a more realistic study-case. The first source of noise we consider is added to the current value of the features $\mathbf{s}$. In a real setup this might come from, *e.g.*, the image processing routine that detects the points of interest or estimators that produce a continuous feedback when an image is not available. We therefore add to each feature coordinate a random value, drawn from a truncated Gaussian distribution with zero mean. The truncation is performed symmetrically at $\pm 3\sigma$, σ

being the standard deviation of the unbounded distribution. In the experiments, we used $\sigma = 2 \cdot 10^{-3}$ (corresponding to about 1 pixel). A similar strategy has been used to simulate uncertainties in the unknown depths of the observed points, by adding to $\mathbf{z}$ a noise with $\sigma = 3 \cdot 10^{-3}$.

Regarding the measurement of joint positions and velocity, we assume that only encoders are available. These devices are simulated by discretizing the values of joints angles with a given resolution Δq. For this purpose we used $\Delta q = 10\,\mu\mathrm{rad}$.

Noisy signals are then filtered using low-pass Butterworth 4th-order filters and finally used in all the calculations performed by controllers.

B. Controllers Implementation and Parameters Tuning

1) First-Order (FO) Controller: One problem with the first order law (15) is that it presents a discontinuity in velocity at the very first iteration. In order to guarantee a "smooth start", we exploit the concepts presented in [16]. They showed that when a new task is activated, it is possible to achieve a smooth transition in the control input by adding a homogeneous term to the command. Applied to our case, the velocity reference $\dot{\mathbf{q}}_1^\star$ is now evaluated as:

$$\dot{\mathbf{q}}_1^\star = \mathbf{J}_\mathbf{s}^+ \left(\dot{\mathbf{s}}^\star - \lambda \mathbf{e}_\mathbf{s} + e^{-\mu t}\mathbf{e}_\mathbf{s}(0) \right) \qquad (23)$$

where $\mathbf{e}_\mathbf{s}(0)$ is the features error for $t = 0$, and we assume null desired features velocity at the beginning ($\dot{\mathbf{s}}^\star(0) = \mathbf{0}$). Due to this modification, the command at the first iteration is zero, thus enforcing a smooth transition of the desired velocity of the manipulator. The effect on the error dynamics is that it now behaves as an equivalent second order system – if perfect linearization was possible – according to the autonomous equation

$$\ddot{\mathbf{e}}_\mathbf{s} + (\mu + \lambda)\,\dot{\mathbf{e}}_\mathbf{s} + \mu\lambda\mathbf{e}_\mathbf{s} = \mathbf{0} \qquad (24)$$

One interesting consequence of this choice is that it is possible to enforce the same desired behavior of the features evolution with both a first and a second order control. Indeed, in a perfect world not affected by noise, the two control strategies are perfectly equivalent in terms of error evolution.

After computing the desired velocity (23), the effort command is computed as $\boldsymbol{\tau} = \gamma\mathbf{M}\,(\dot{\mathbf{q}}_1^\star - \dot{\mathbf{q}}) + \mathbf{b}$.

Regarding gains tuning, we decide to impose a critically damped behavior for the transient response, as this gives the fastest convergence without oscillations:

$$\lambda = \mu = \omega_n \qquad (25)$$

wherein $\omega_n > 0$ defines the roots of the characteristic polynomial of the second order system. The autonomous response of the error would be given by $\mathbf{e}_\mathbf{s}(t) = e^{-\omega_n t}\,(1 + \omega_n t)\,\mathbf{e}_\mathbf{s}(0)$, which is strictly decreasing and converges to zero. We finally select the value of ω_n so that after a time T the error is a fraction of its original value, *i.e.*, $e^{-\omega_n T}\,(1 + \omega_n T) = \alpha$, $\alpha \in (0, 1)$. Solving the equation numerically for $\alpha = 0.05$ and $T = 2\,\mathrm{s}$ gives $\omega_n \simeq 2.37\,\mathrm{s}^{-1}$.

Concerning the proportional velocity gain γ in (18), the higher its value the better will be the tracking of the desired

velocity profile. On the other hand, it will increase the control efforts $\boldsymbol{\tau}$, and therefore a trade-off has to be found. In our implementation, a value of $\gamma = 8\,\mathrm{s}^{-1}$ has been chosen, which allows a velocity error to decrease of $80\,\%$ in $0.2\,\mathrm{s}$.

2) First-Order with Feed-Forward (FOFF): The implementation of this controller is very similar to the one of FO. We use here as well the smooth velocity transition and the same set of gains. The feed-forward term $\ddot{\mathbf{q}}_{ff}$ is computed as the numerical derivative of the reference command $\dot{\mathbf{q}}_1^\star$. It is well-known that this highly amplifies noise, resulting in a generally unfeasible, if not just undesirable, control signal. To get a nicer behavior and reduce the impact of noise, (23) is thus filtered prior to differentiation. This considerably reduces the shakiness of the controller. The final efforts sent to the motors are evaluated using (18) and the IDM.

3) Second-Order (SO) Controller: This controller exploits the acceleration in the feature space. It computes joints accelerations using (22) to complete the visual task while damping the velocity of the manipulator, and subsequently exploits (13) to compute the actuation signal. To obtain an estimation of $\dot{\mathbf{e}}_{\mathbf{s}}$, (11a) was used. The control gains k_p and k_d are selected to obtain a critically damped evolution as in the FO case, with $k_p = \omega_n^2$, $k_d = 2\omega_n$, and $\omega_n = 2.37\,\mathrm{s}^{-1}$. The last parameter to be tuned is k_v, the gain appearing in the velocity damping task (21). As we assume the Servoing Task to be completed in about $2\,\mathrm{s}$, this parameter is tuned so that the convergence of velocity to zero is achieved after a longer time. In practice, we used the value $k_v = 1\,\mathrm{s}^{-1}$, which theoretically ensures that any non-null velocity will be reduced to about the $5\,\%$ of its initial value in $3\,\mathrm{s}$.

C. Regulation to a Fixed Configuration

The first set of tests involves positioning the end-effector in front of a set of four points at a distance of $0.5\,\mathrm{m}$, see Fig. 1. As the desired value of the features does not change during the tests, in the controllers the quantities $\dot{\mathbf{s}}^\star$ and $\ddot{\mathbf{s}}^\star$ are set to zero.
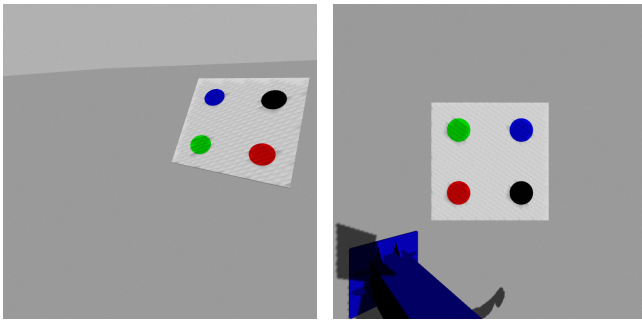

Fig. 1. Initial (left) and desired (right) configuration of the observed object. The points used for the task are the centers of the four circles.

The initial relative transformation from the camera to the object presents a rotation of about $90°$ around the optical axis of the sensor, leading to the well-known retraction problem. Since this affects all control laws, we did not try to solve the problem to optimize the 3D behavior of the camera. We

believe that a better choice of the descriptive features, such as almost decoupled image moments [17], would result in a trajectory of higher quality in all the cases.

We report results related to two cases, an ideal test run in absence of noise (Fig. 2) and a more realistic case with disturbances as discussed in Section III-A (Fig. 3). When signals are not influenced by any perturbation, all control strategies show nice convergence behavior and smooth control inputs. Compared to FOFF and SO, FO presents a deviation in convergence from the ideal evolution, justified by the intermediate velocity regulation not being able to perfectly track the desired command $\dot{\mathbf{q}}_1^\star$. FOFF and SO are instead able to better follow the desired behavior and the error evolves according to the expected second order exponential decay. The difference in these two controllers can be seen only in the control input, yet not significantly different.
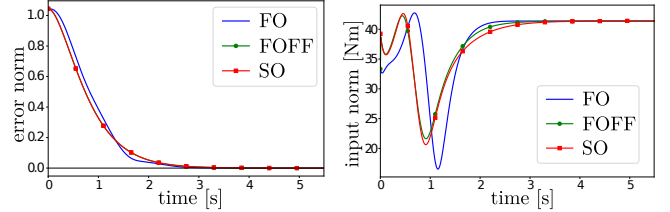

Fig. 2. Comparison of the visual error (left) and control input (right) using the different control strategies, in absence of noise. FOFF and SO are almost indistinguishable in terms of error evolution.

When noise is considered, more evident differences appear. FO's convergence becomes worse, with a little overshoot near to the equilibrium configuration, whereas FOFF and SO still feature a nice evolution of the error towards zero. In terms of control inputs, all strategies suffer from the added noise. In particular, the oscillations corresponding to FOFF are quite large, with peak-to-peak amplitude of about $30\,\mathrm{N\,m}$ even at equilibrium. Oscillations of FO and SO controllers are bounded instead to a reasonable amount, between $0.5\,\mathrm{N\,m}$ and $1\,\mathrm{N\,m}$ of amplitude in both cases.
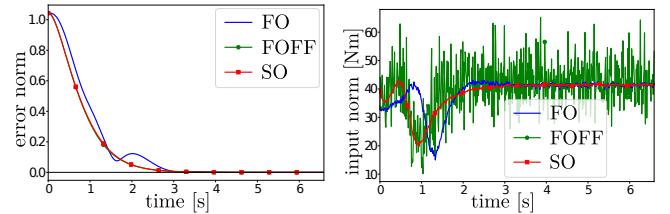

Fig. 3. Comparison of visual error (left) and control inputs (right) when noise is injected in simulations. FOFF and SO are still almost indistinguishable in terms of error evolution.

It is possible to enhance the convergence of FO by increasing the gain γ or selecting a faster behavior (higher ω_n), at the cost of higher oscillations in the commanded efforts. Similarly, to reduce the shakiness of the control signal in FOFF, the response has to be penalized with lower values of γ and/or ω_n.

It is also worth mentioning that when noises are of higher magnitude the relative performances of these controllers are not significantly altered. FOFF and SO still give satisfactory results in terms of error convergence, and the oscillations in the control commands produced by FO and SO are quite smaller compared to those introduced by FOFF.

D. Tracking of a Visual Trajectory

In this section, we consider the case of varying desired features, in which case $\dot{\mathbf{s}}^\star$ and $\ddot{\mathbf{s}}^\star$ are non-zero. For each normalized point $\mathbf{s}_i = [x_i, y_i]^T$, the motion is defined as a circular trajectory starting at the instant t_0 and ending after a time period T:

$$\begin{cases} x_i^\star(t) = x_i^\star(t_0) + R\left(\cos\rho(t - t_0) - 1\right) \\ y_i^\star(t) = y_i^\star(t_0) + R\sin\rho(t - t_0) \end{cases} \quad (26)$$

In this definition, $\rho(t)$ is a quintic polynomial whose value changes from 0 to 2π in $T = 2\,\mathrm{s}$, such that $\dot{\rho}(0) = \dot{\rho}(T) = 0$ and $\ddot{\rho}(0) = \ddot{\rho}(T) = 0$. We assume the desired initial position of each point $\mathbf{s}_i^\star(t_0)$ to be the one attained after the regulation task described in the previous section. The amplitude R is set to 0.2, and thus the radius of the physical trajectory is expected to be of $0.1\,\mathrm{m}$ (the sensor should be at $0.5\,\mathrm{m}$ from the four points, as explained before). In addition, we assume $t_0 \gg 0$, so that the homogeneous term in (23) does not affect the control, as the desired velocity is already null at t_0.

Results are depicted in Fig. 4, which shows that FO is the worst controller in terms of tracking error. FOFF again presents high oscillations in the input command during the motion, but still with good tracking performances. Finally, SO gives the best results, with tracking errors of the same magnitude of FOFF and reduced effort oscillations.
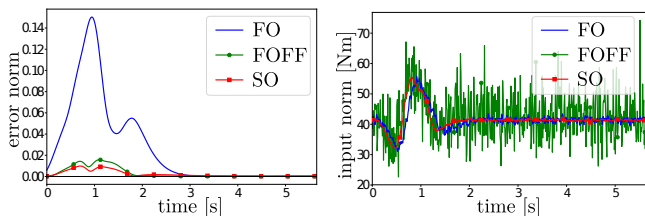

Fig. 4. Comparison of the visual error (left) and control input (right) during the execution of a trajectory. Time axes are relative to t_0.

IV. CONCLUSIONS

In this work, we reviewed and compared simple Visual Servoing control schemes based on both first and second order models. It has been shown that for a redundant manipulator it is desirable, if not necessary, to consider a secondary task in the case the Visual Task is formulated at the acceleration level in order to fully constrain the degrees of freedom and prevent uncontrolled motions spanning the kernel of the Feature Jacobian.

We showed via simulations that controllers can be designed both from velocity and acceleration commands to obtain a very similar evolution of visual features error. In particular, FOFF and SO were equally able to ensure good

performances in terms of error regulation. However, results suggest that schemes based on second-order models are preferable in order to obtain smoother control efforts.

In this analysis, redundancy was not exploited to truly achieve any secondary objective. We believe that with further improvements it will be possible to enhance existing strategies by, *e.g.*, minimizing the efforts along the trajectory while ensuring Visual Tasks to be properly executed. For this purpose, a possible future direction of work could consider the use of Model Predictive Controls.

We are working on a real implementation of the proposed strategies, in order to validate with experiments our simulated results. We are especially interested in performing Servoing tasks at higher speed, and to study more in depth the limits of FOFF and SO in such context.

REFERENCES

[1] F. Chaumette and S. Hutchinson, "Visual servo control. I. Basic approaches," *IEEE Robotics and Automation Magazine*, dec 2006.
[2] ——, "Visual servo control. II. Advanced approaches [Tutorial]," *IEEE Robotics and Automation Magazine*, mar 2007.
[3] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, jun 1992.
[4] Y. Fang, A. Behal, W. Dixon, and D. Dawson, "Adaptive 2.5D visual servoing of kinematically redundant robot manipulators," in *Proceedings of the 41st IEEE Conf. on Decision and Control, 2002*. IEEE, 2003.
[5] C. Wang, C. Y. Lin, and M. Tomizuka, "Visual servoing considering sensing dynamics and robot dynamics," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*. Elsevier, jan 2013.
[6] Hong Zhang and J. Ostrowski, "Visual servoing with dynamics: control of an unmanned blimp," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, 1999.
[7] J. Thomas, G. Loianno, K. Sreenath, and V. Kumar, "Toward image based visual servoing for aerial grasping and perching," in *Proceedings - IEEE Int. Conf. on Robotics and Automation*. IEEE, may 2014.
[8] A. Mohebbi, M. Keshmiri, and W. F. Xie, "An acceleration command approach to robotic stereo image-based visual servoing," in *IFAC Proceedings Volumes (IFAC-PapersOnline)*. Elsevier, jan 2014.
[9] S. Vandernotte, A. Chriette, P. Martinet, and A. S. Roos, "Dynamic sensor-based control," in *2016 14th Int. Conf. on Control, Automation, Robotics and Vision, ICARCV 2016*. IEEE, nov 2016.
[10] M. Keshmiri, W. F. Xie, and A. Mohebbi, "Augmented image-based visual servoing of a manipulator using acceleration command," *IEEE Transactions on Industrial Electronics*, oct 2014.
[11] A. De Luca and G. Oriolo, "Efficient Dynamic Resolution of Robot Redundancy," in *1990 American Control Conf.* IEEE, may 2018.
[12] A. Escande, N. Mansard, and P. B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *International Journal of Robotics Research*, jun 2014.
[13] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for Visual Servoing," *IEEE Robotics & Automation Magazine*, dec 2005.
[14] A. De Luca, G. Oriolo, and P. R. Giordano, "On-line estimation of feature depth for image-based visual servoing schemes," in *Proceedings - IEEE Int. Conf. on Robotics and Automation*. IEEE, apr 2007.
[15] R. Spica, P. R. Giordano, and F. Chaumette, "Plane estimation by active vision from point features and image moments," in *Proceedings - IEEE Int. Conf. on Robotics and Automation*. IEEE, may 2015.
[16] N. Mansard and F. Chaumette, "Task sequencing for high-level sensor-based control," *IEEE Transactions on Robotics*, feb 2007.
[17] F. Chaumette, "Image Moments: A General and Useful Set of Features for Visual Servoing," *IEEE Transactions on Robotics*, aug 2004.