UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CENTRALE NANTES

THÈSE DE DOCTORAT DE

ÉCOLE CENTRALE DE NANTES
COMUE UNIVERSITÉ BRETAGNE LOIRE

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Robotique

Par

Franco FUSCO

Dynamic Visual Servoing for Fast Robotics Arms

Thèse présentée et soutenue à Nantes, le 27 November 2020
Unité de recherche : ARMEN LS2N
Thèse N° : XXX

Rapporteurs avant soutenance :

Jacques Gangloff	Professeur des Universités, Télécom Physique Strasbourg
Olivier Stasse	Directeur de recherches, LAAS-CNRS

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	Prénom Nom	Fonction, établissement *(à préciser après la soutenance)*
Examinateurs :	Guillaume Allibert	Maître de conférences, Université de Nice-Sophia Antipolis
	Abdelhamid Chriette	Maître de conférences, Centrale Nantes
	Estelle Courtial	Maître de conférences, Polytech Orléans
	Isabelle Fantoni	Directrice de recherches, LS2N
Dir. de thèse :	Philippe Martinet	Directeur de recherches, Inria Sophia-Antipolis Méditerranée
Co-dir. de thèse :	Olivier Kermorgant	Maître de Conférences, Centrale Nantes

A Sara e Lalle,

Siete stati la mia luce
in questi mesi di buio.

A Mariacarla,

"Voglio però ricordarti com'eri,
pensare che ancora vivi;
voglio pensare che come allora mi ascolti
e che come allora sorridi"

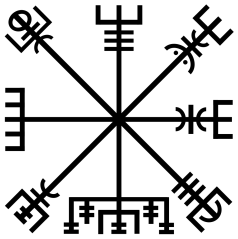


Table of Contents

INTRODUCTION

In order to manufacture an item, several operations are generally performed in succession, such as drilling, cutting, welding, assembly. Some of these tasks can be executed by automatic machines, possibly with several repetitions per second. However, such machines lack of flexibility, in the sense that they are usually designed to accomplish a very specific task in controlled conditions. As an example, components might need to be precisely positioned on a conveyor belt in order for a machine to properly perform its task. On the other hand, a human operator offers much more flexibility. A badly positioned component does not pose major issues for a human worker, who can, in addition, perform a variety of tasks according to the needs of the production line. Nonetheless, human operators are inadequate for the execution of tasks that require millimeter (or even micrometer) precision, and in addition they can experience health issues such as repetitive motion injuries due to execution of similar tasks over and over.

The use of robotic systems can improve the productivity since, like automatic machines, they can execute tasks with high repeatability and reliability. However, they can also be reprogrammed, and therefore, like human operators, can be assigned to different tasks depending on the production needs. Nonetheless, their productivity can still be limited. As mentioned before, one of the main advantages of having humans perform a given operation is their ability to adapt to mutable contexts, in which, as an example, objects might not be precisely positioned on a conveyor. To provide the same level of robustness and flexibility, robotized manufacturing systems must be equipped not only with proprioceptive sensors, such as motor encoders used in low-level joints control, but also with exteroceptive ones, allowing to gather information about both the objects to interact with and the surrounding environment. These additional sensors often correspond to vision systems composed of one or more cameras, allowing to gather information about the relative position between a target object and the actuated robot. Visual signals can then be used to evaluate control inputs that are robust with respect to several uncertainties, such as modeling and calibration errors, ultimately allowing an arm to precisely position itself with respect to the observed object.

Unfortunately, the motion that results from such control strategies is generally slow,

due to a combination of factors. As an example, images are affected by motion blur, making object detection and tracking harder and less reliable at higher speeds. Furthermore, the models that describe the interaction between the sensing and sensed systems are often formulated at a kinematic level, considering at most velocity information. With faster motions, however, the accelerations and the effects due to the dynamics become non-negligible and should be considered in the design of robot controllers by means of new, richer models. Even if these have not yet been used extensively in the research or industrial domain, few existing works suggest that they are indeed a fundamental ingredient required to attain high-speed performances. More specifically, the studies by Dahmouche [Dah+12] and Ozgur [Ozg+13] showed that including this "second-order" information to describe the evolution of the visual features allows to obtain a simultaneous estimation of the pose and velocity of the platform of parallel manipulators moving at high speed. Additional investigations in this field were conducted by Vandernotte [Van+16], who used acceleration models to establish a direct connection between vision and robot dynamics.

As for the problem of controlling the robot, almost all works that investigated the use of this new type of interaction focused on the use of feedback linearization techniques, usually in the form of a proportional and derivative control [Dah+12; KX12]. While this ensures desirable properties such as robustness with respect to calibration errors, it also presents some drawbacks. Just to provide some examples, motions can be longer than necessary, there are no guarantees of maintaining object visibility, joint limits could be violated. To deal with these issues, Model Predictive Control (MPC) can be employed [ACC10]. Not only it can take into account constraints imposed on the system, but it can also guide the robot through a path that is shorter, thus reducing the time to completion of a task.

These concepts are just some of the topics investigated by the PROMPT (Performances de la Robotique Manufacturière en Perception de Tâche) project, of which this thesis is part. It is one of the projects funded by RFI Atlanstic 2020 [1], and its overall goal is to define a set of new methodologies which allow to enhance the execution speed of sensor-based servoing tasks. This can be achieved by considering multiple, parallel, improvements in different fields. First of all, more advanced perception algorithms can be used to fusion altogether data gathered by different sensors, both exteroceptive and proprioceptive ones. As an example, information relative to an observed object can be acquired via a perspective camera, while an Inertial Measurement Unit measures the

1. https://atlanstic2020.fr/

velocity and acceleration of a moving sensor. Using filtering techniques, it is possible to better estimate the relative motion of the two entities. Moreover, by optimally placing the sensors within the environment and on the controlled system, it is possible to enhance the reliability of the estimation. Another mean to reduce the execution time of a task is the use of better models that provide a richer information about the evolution of the system with respect to its environment, in particular, allowing to properly express the dynamics relating them. Finally, advanced control laws should be considered that are capable of guiding the system from an initial to a desired configuration by following a better path and in a shorter amount of time.

The work presented in this manuscript focuses on these last topics. In particular, it studies the second-order interaction of eye-in-hand robotic configurations, potentially for any type of feature, and compares it to more classical strategies. Furthermore, it builds, on top of the acceleration models, advanced control schemes based on the MPC formulation. The goal is to use the acceleration information to define better predictors that can more adequately steer the sensor towards the desired configuration. The approach is validated both in simulation and with real experiments, and for different types of features. Finally, a last contribution aims at reducing the computational burden of the nonlinear optimization involved in predictive strategies, which in some cases might prevent real-time feasibility. It is based on the work by Alamir [Ala06] on input parameterizations, and can effectively reduce the time needed to evaluate control commands. A test involving a visual trajectory tracking task is proposed to underline the appeal of this approach.

Chapter 1 presents an overview of classical visual servoing modeling and Image Based Visual Servoing (IBVS), followed by details on few more advanced schemes that try to enhance the quality of the sensor motions and to ensure satisfaction of generic constraints along the path. The following chapter introduces second-order models, discussing their relation with the dynamics of the robot and detailing how feedback controllers can be defined on top of them. An brief review of MPC is then presented in Chapter 3, with a focus on predictive strategies applied to visual servoing, and the original work that combines MPC and second-order models is then detailed. Chapter 4 provides the fundamental ideas of “control parameterization” for MPC applications. It also describes several simulations that were performed to validate the concept. Finally, the general conclusions are reported, as well as the perspective for future works and improvements.

Publications

Journal Articles

Franco Fusco, Olivier Kermorgant and Philippe Martinet. *Integrating Features Acceleration in Visual Predictive Control*. IEEE Robotics and Automation Letters 5.4 (2020): 5197-5204.

Conference Proceedings

Franco Fusco, Olivier Kermorgant and Philippe Martinet. *Integrating Features Acceleration in Visual Predictive Control*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020.

Franco Fusco, Olivier Kermorgant and Philippe Martinet. *A Comparison of Visual Servoing from Features Velocity and Acceleration Interaction Models*. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2019.

Franco Fusco, Olivier Kermorgant and Philippe Martinet. *Constrained Path Planning using Quadratic Programming*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

Franco Fusco, Olivier Kermorgant and Philippe Martinet. *Improving Relaxation-based Constrained Path Planning via Quadratic Programming*. International Conference on Intelligent Autonomous Systems (IAS). Springer, 2018.

Chapter 1

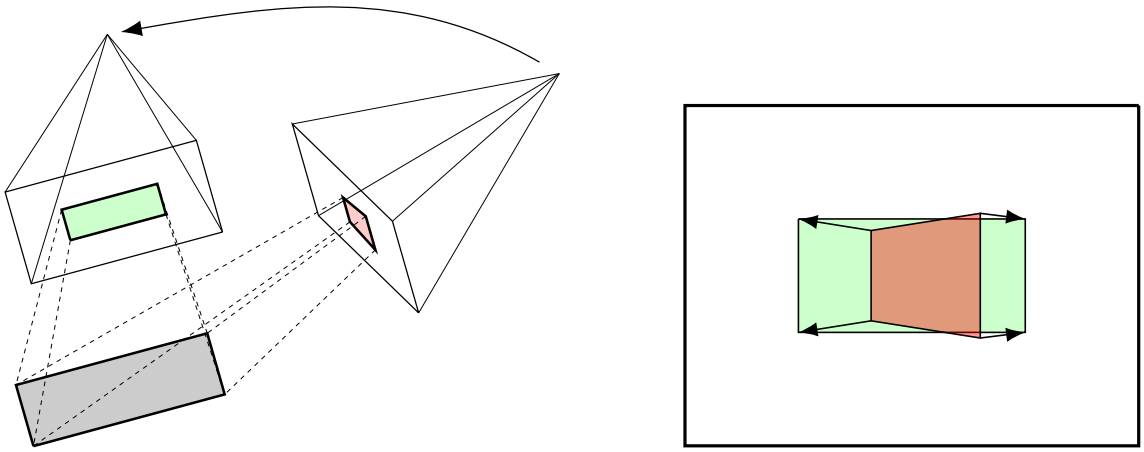
Visual Servoing

Visual Servoing is a robust control technique that allows precise positioning of a sensor with respect to an observed object. It is robust to modeling errors and features a large convergence domain thanks to the use of sensor measurements directly in the control loop.

The positioning task is achieved by actively regulating a set of descriptive features, which are extracted from the visual information. The key idea is that when the value of these features is changed, the relative pose of the sensor with respect to the object has to change accordingly. More specifically, the positioning task can be achieved by regulating the features to what the sensor would measure when properly positioned in front of the object, as sketched in Figure 1.1.

Several servoing schemes can be found in the literature, each one characterized by different kind of features and different control laws. In particular, a distinction is commonly made between 2D features, that can be extracted directly from images, and 3D ones, which need to be reconstructed from sensor measurements [WHB96; Thu+02]. Schemes that use the first set of features are commonly referred to as Image Based Visual Servoing (IBVS), while those based on 3D information are named Position Based Visual Servoing (PBVS). Hybrid approaches are also possible, in which both 2D and 3D features are used at the same time. Despite these differences, all schemes exploit the same formalism based on the interaction matrix, which provides a connection between the motion of a sensor and the resulting changes of the features. Using these models, a feedback control law can be established that steers the features to the desired configuration, and consequently moving the sensor in the proper location.

Simulations and experiments included in this thesis used only 2D features, and for this reason the second section of this chapter briefly illustrates IBVS control, focusing on the case in which normalized image points are used as features. More advanced approaches are then discussed, tackling the issues that come from the basic approach. These approaches attempt to solve some well-known issues arising from the use of normalized coordinates. The goal is to obtain better sensor motions in the operational space, which represents



(a) An object observed from different poses. (b) Object projection in the two views.

Figure 1.1 – A perspective camera observing a rectangular object from different views. Moving the camera from one pose to the other is equivalent to moving the corners of the projected shape from one image to the other.

indirectly a way to shorten motions and thus speedup task execution.

Finally, some works that focused on servoing under constraints are presented. Typical constraints that are to be enforced are the visibility of the observed object and keeping the joints within their mechanical limits, which were considered in the original contribution presented in Chapter 3.

1.1 Modeling and control

1.1.1 The Interaction Matrix

In order 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 values is gathered in a vector, denoted as $\boldsymbol{s} \in \mathbb{R}^{n_s}$, which can be extracted directly or by reconstruction from the measurements acquired by one or more sensors. The goal of the servoing task then becomes to regulate the value of the features to a desired one, denoted as $\boldsymbol{s}^\star$. For this purpose, it is first of all necessary to describe how the feature vector evolves as a function of the motion of the sensor and of the object. The most common approach in this sense is to use a kinematic relation that expresses the time derivative of the features,

$\dot{\boldsymbol{s}}$, in function of the relative sensor-object motion:

$$\dot{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}}\left(\boldsymbol{s}, \boldsymbol{z}\right)\mathbf{v} \tag{1.1}$$

The matrix $\mathbf{L}_{\boldsymbol{s}}\left(\boldsymbol{s}, \boldsymbol{z}\right) \in \mathbb{R}^{n_s \times 6}$ is known in the literature as interaction matrix, and is in general a function of the features themselves and of a set of n_z parameters $\boldsymbol{z}$. To shorten the notation in the sequel, the dependency on $\boldsymbol{s}$ and $\boldsymbol{z}$ will be implicitly assumed, and only $\mathbf{L}_{\boldsymbol{s}}$ will be written unless needed otherwise.

The vector $\mathbf{v}$ in (1.1) represents a kinematic screw, or twist, describing the instantaneous motion of the sensor frame $\mathcal{F}_c$ relative to the object frame $\mathcal{F}_o$. In practice, it consists in the concatenation of both the linear and angular velocities ($\boldsymbol{v}$ and $\boldsymbol{\omega}$ respectively) of the sensor with respect to the object, and expressed in the coordinate system of the sensor itself.

One interesting aspect of this formulation is that it unifies multiple setups at once, since in the kinematic relation defined above it is not specified which frame is the moving one. When the observed object is stationary and the features change due to the motion of the sensor, the setup is commonly named *eye-in-hand* configuration. In this case, the relative twist can be expressed as:

$$\mathbf{v} = \begin{bmatrix} {}^{w}\mathbf{R}_{c}^{T}\,{}^{w}\boldsymbol{v}_{c} \\ {}^{w}\mathbf{R}_{c}^{T}\,{}^{w}\boldsymbol{\omega}_{c} \end{bmatrix} = \begin{bmatrix} {}^{w}\mathbf{R}_{c}^{T} & \\ & {}^{w}\mathbf{R}_{c}^{T} \end{bmatrix} \begin{bmatrix} {}^{w}\boldsymbol{v}_{c} \\ {}^{w}\boldsymbol{\omega}_{c} \end{bmatrix} = {}^{c}\mathbb{T}_{w}\,{}^{w}\mathbf{v}_{c} \tag{1.2}$$

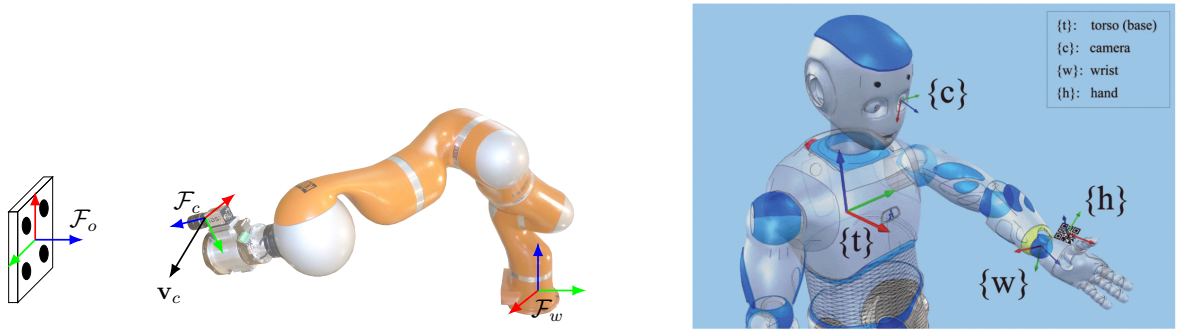
wherein ${}^{w}\mathbf{v}_{c}$ is the twist of the sensor with respect to a fixed "world" frame $\mathcal{F}_w$, while ${}^{w}\mathbf{R}_{c} \in \mathrm{SO}(3)$ is the rotation matrix representing the orientation of $\mathcal{F}_c$ with respect to $\mathcal{F}_w$.

Conversely, a *eye-to-hand* configuration is obtained whenever a fixed sensor observes a moving object. In this case, denoting with ${}^{c}\boldsymbol{t}_{o/c}$ the translation vector from the origin of $\mathcal{F}_c$ to that of $\mathcal{F}_o$ and expressed in the sensor frame, the expression of the kinematic screw $\mathbf{v}$ writes as:

$$\mathbf{v} = -\begin{bmatrix} \mathbf{I} & \left[{}^{c}\boldsymbol{t}_{o/c}\right]_{\times} \\ & \mathbf{I} \end{bmatrix} {}^{c}\mathbb{T}_{w}\,{}^{w}\mathbf{v}_{o} = -{}^{c}\mathbb{V}_{o}\,{}^{c}\mathbb{T}_{w}\,{}^{w}\mathbf{v}_{o} \tag{1.3}$$

in which ${}^{c}\mathbb{T}_{w}$ has the same meaning as before, and ${}^{w}\mathbf{v}_{o}$ represents the linear and angular velocity of the object in the fixed frame $\mathcal{F}_w$.

It is also possible to consider hybrid cases with multiple sensors in different configurations [FCM00; LSV05], or in which both the sensor and the object are moving [MKF08]. However, it usually the case that only one of the two motions is caused by the actuation



(a) Eye-in-hand (b) Eye-to-hand

Figure 1.2 – Examples of (a) eye-in-hand and (b) eye-to-hand [CSC16] configurations. On the left, the relevant frames are named as proposed in the text, while on the right the naming used in the original paper has been kept. In practice, $\mathcal{F}_w$ corresponds to `{t}`, $\mathcal{F}_c$ to `{c}` and $\mathcal{F}_o$ to `{h}`.

of the robotic system, while the other is assumed to be a disturbance. In these cases, it is rather common to rewrite the interaction model as:

$$\dot{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}}\mathbf{v} + \frac{\partial \boldsymbol{s}}{\partial t} \tag{1.4}$$

wherein $\mathbf{v}$ corresponds to the relative kinematic screw that would be obtained solely due to the actuation. The second term takes into account the contribution due to the uncontrolled motion, and usually has to be estimated and compensated for [BC95; Gin+05; Sha+13].

The interaction model (1.1) provides a link between the rate of change and the relative sensor motion in Cartesian space. However, it can also be rewritten in terms of the generalized coordinates of the controlled system. In particular, for a serial kinematic chain featuring n_q joints, the time derivative of $\boldsymbol{s}$ can be expressed as a function of the joint velocities $\dot{\boldsymbol{q}} \in \mathbb{R}^{n_q}$:

$$\dot{\boldsymbol{s}} = \mathbf{J}_{\boldsymbol{s}}\dot{\boldsymbol{q}} \tag{1.5}$$

$\mathbf{J}_{\boldsymbol{s}}$ is sometimes referred to as feature Jacobian [CH07], and takes a different form depending on the type of servoing configuration:

$$\mathbf{J}_{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}}\,{}^{c}\mathbb{T}_{w}\mathbf{J} \qquad \text{(eye-in-hand)} \tag{1.6a}$$

$$\mathbf{J}_{\boldsymbol{s}} = -\mathbf{L}_{\boldsymbol{s}}\,{}^{c}\mathbb{V}_{o}\,{}^{c}\mathbb{T}_{w}\mathbf{J} \qquad \text{(eye-to-hand)} \tag{1.6b}$$

In both cases, $\mathbf{J}$ is the robot Jacobian providing the end-effector velocity expressed in the base frame of the robot [KD04], and is a function of the current joint coordinates $\boldsymbol{q}$. Moreover, it is assumed that $\mathcal{F}_w$ coincides with the base of the robot, while the end-effector frame $\mathcal{F}_{ee}$ coincides with $\mathcal{F}_c$ in the case of an eye-in-hand configuration, and with $\mathcal{F}_o$ for a eye-to-hand setup.

1.1.2 Control via Visual Feedback

To control the relative positioning of the sensor with respect to an object it suffices to regulate the values of the descriptive features to a given desired configuration $\boldsymbol{s}^\star$. This is commonly achieved using the task function formalism [SE91], in which the objective is to regulate the feature error

$$\boldsymbol{e}_{\boldsymbol{s}} \doteq \boldsymbol{s} - \boldsymbol{s}^\star \tag{1.7}$$

to zero. The most common approach in this sense is to attempt to enforce an exponential decay of the error by imposing the desired profile $\dot{\boldsymbol{e}}_{\boldsymbol{s}} = -\lambda \boldsymbol{e}_{\boldsymbol{s}}$, in which λ is a positive proportional gain. Considering (1.5), and assuming the joint velocity of the manipulator to be the control input of the system, the problem is thus equivalent to solve the linear system

$$\mathbf{J}_{\boldsymbol{s}} \dot{\boldsymbol{q}} = \dot{\boldsymbol{s}}^\star - \lambda \boldsymbol{e}_{\boldsymbol{s}} \tag{1.8}$$

in which $\dot{\boldsymbol{s}}^\star$ provides a feed-forward derivative term, which can be assumed null if a fixed configuration has to be attained. The system admits a unique solution in $\dot{\boldsymbol{q}}$ only if $\mathbf{J}_{\boldsymbol{s}}$ is squared and non-singular, which is not necessarily always true, and thus only an approximate solution will be possible in general. The system is thus solved in the least squares sense, obtaining the following proportional control law:

$$\dot{\boldsymbol{q}} = \widehat{\mathbf{J}_{\boldsymbol{s}}^{+}} \left(\dot{\boldsymbol{s}}^\star - \lambda \boldsymbol{e}_{\boldsymbol{s}} \right) \tag{1.9}$$

where $\widehat{\mathbf{J}_{\boldsymbol{s}}^{+}}$ represents an estimation of the pseudo-inverse of the feature Jacobian. This control law can be proven locally stable in a neighborhood of $\boldsymbol{s}^\star$ [CH06] given that a sufficiently good approximation of $\mathbf{J}_{\boldsymbol{s}}$ is used. There are different factors that can influence the stability of such control law, such as geometric calibration errors, and uncertainty in the parameters $\boldsymbol{z}$. Nonetheless, the advantage of visual servoing is that it is rather robust

to such modeling uncertainties, and a coarse estimation of the parameters is often sufficient to ensure local stability.

1.2 Image-Based Visual Servoing

IBVS schemes focus on the case of features that can be extracted directly from image measurements, as opposed to PBVS [WHB96; Thu+02] and hybrid approaches [MCB99; MC02a] in which at least some of the features need to be reconstructed via pose estimation algorithms [DD95; Com+06]. One of the advantages of regulating quantities that come directly from the vision sensor is thus that little if no knowledge at all of the observed object is required.

Arguably the most common servoing scheme is based on the use of normalized points as features, even though other geometric primitives can be used such as lines or ellipses [ECR92; CRE93]. Given a set of 3D points, each with coordinates X_i, Y_i and Z_i expressed in the sensor frame $\mathcal{F}_c$, their normalized coordinates can be used as features, *i.e.*,

$$\boldsymbol{s} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \dots \end{bmatrix} = \begin{bmatrix} X_1/Z_1 \\ Y_1/Z_1 \\ X_2/Z_2 \\ Y_2/Z_2 \\ \dots \end{bmatrix} \tag{1.10}$$

The extraction and tracking of these coordinates from an image can be achieved using several well-established techniques, such as those based on SIFT (Scale Invariant Feature Transform) [Low99] or SURF (Speeded Up Robust Features) [BTV06] descriptors, and a conversion from pixel to normalized coordinates can be obtained from the sensor projection model.

The interaction matrix of the i-th point can be obtained easily using the law of composition of velocities from classical mechanics, and takes the following well known form:

$$\mathbf{L}_{\boldsymbol{s}_i} = \begin{bmatrix} -\frac{1}{Z_i} & 0 & \frac{x_i}{Z_i} & x_i y_i & -x_i^2 - 1 & y_i \\ 0 & -\frac{1}{Z_i} & \frac{y_i}{Z_i} & y_i^2 + 1 & -x_i y_i & -x_i \end{bmatrix} \tag{1.11}$$

while the interaction matrix of the whole set of points is obtained by vertically stacking the individual matrices. The vector of parameters $\boldsymbol{z}$ consists in this case in the collection of the depths Z_i of each point. These values need to be estimated [DOG07; SG13] or

approximated, as an example using the depths at the desired configuration, *i.e.*, $Z_i = Z_i^\star$. Finally, at least four points should be used [MR93] to avoid singularities in the interaction matrix.

Schemes based on image points as features are simple to implement, but can lead to unsatisfactory results due to the structure of the interaction matrix. First of all, since the total number of feature coordinates n_s is usually at least 8, there exist configurations known as local minima [Cha98], in which the proportional control law presented before leads to $\mathbf{v} = \mathbf{0}$ despite having $\boldsymbol{e_s} \neq \mathbf{0}$. Furthermore, the components of the relative twist $\mathbf{v}$ are not decoupled. This fact is particularly important when the sensor has to perform a large rotation in order to reach the desired configuration. Due to the high coupling of the degrees of freedom, the motion of the sensor presents not only the desired rotation, but also an unwanted translation, usually in the form of a possibly large retraction from the observed object. This is obviously an issue when the sensor is attached on the end-effector of a manipulator, since the tip of the serial chain can only move within a limited workspace.

1.2.1 Features with higher decoupling

One way to solve the issues that appear when using image points is to consider alternative features which provide better decoupling. In particular, it would be desirable to find a set of features partitioned as:

$$\begin{bmatrix} \dot{\boldsymbol{s}}_1 \\ \dot{\boldsymbol{s}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{L}_1 \\ \mathbf{L}_2 \end{bmatrix} \mathbf{v} \tag{1.12}$$

whose interaction matrices have orthogonal rows, *i.e.*, $\mathbf{L}_1\mathbf{L}_2^T = \mathbf{O}$. From this orthogonality condition, it would follow that the proportional control law would be written in the form

$$\mathbf{v} = \underbrace{-\lambda \mathbf{L}_1^+ \boldsymbol{e}_1}_{\mathbf{v}_1} \underbrace{-\lambda \mathbf{L}_2^+ \boldsymbol{e}_2}_{\mathbf{v}_2} \tag{1.13}$$

in which $\mathbf{v}_1 \perp \mathbf{v}_2$. In other words, each of the two features would govern a selected number of degrees of freedom, and the two components would not influence each other. This is the principle at the base of partitioned visual servoing, in which the goal is usually to isolate the motions related to the focal axis of the camera or to decouple translational and rotational motions. More commonly, only a partial partitioning is used, in which the

interaction model writes as

$$\begin{bmatrix} \dot{\boldsymbol{s}}_a \\ \dot{\boldsymbol{s}}_b \end{bmatrix} = \begin{bmatrix} \mathbf{L}_a & \\ \mathbf{L}_{ab} & \mathbf{L}_b \end{bmatrix} \begin{bmatrix} \mathbf{v}_a \\ \mathbf{v}_b \end{bmatrix} \tag{1.14}$$

in which $\mathbf{v}_a$ and $\mathbf{v}_b$ represent a partition of the coordinates of $\mathbf{v}$ (not necessarily with the same ordering). As a consequence of this specific structure, the value of $\mathbf{v}_a$ is dependent solely on that of $\boldsymbol{e}_a$, while that of $\mathbf{v}_b$ is evaluated as

$$\mathbf{v}_b = -\mathbf{L}_b^+ \left(-\lambda \boldsymbol{e}_b + \mathbf{L}_{ab}\mathbf{v}_a\right) \tag{1.15}$$

and thus depends not only on $\boldsymbol{e}_b$ but also on $\boldsymbol{e}_a$, with the term $\mathbf{L}_{ab}\mathbf{v}_a$ acting as a disturbance compensation. This approach is similar to that of 2.5D visual servoing [MCB99], in which the rotational motion is decoupled from the translational one. However, partitioned IBVS schemes are based solely on image measurements and no 3D reconstruction is needed.

As an example, in [CH01] two features are introduced in addition to image point coordinates. The first one, being the orientation of a line joining two of the observed points, is strictly linked to the rotational motion around the optical axis. The second feature is related to the area of the polygon defined by the image points, which is invariant to translation and to rotation around the focal axis. Instead, it changes mainly with the distance of the sensor from the points, and is thus closely linked to the translation.

Another way to solve the issues that appear when using image points is to consider alternative features which provide different motions in the Cartesian space. As a first example, polar coordinates can be used [IO05], in which the interaction matrix of the i-th point is given by

$$\mathbf{L}_{\boldsymbol{s}_i} = \begin{bmatrix} -\frac{\cos\theta_i}{Z_i} & -\frac{\sin\theta_i}{Z_i} & \frac{\rho_i}{Z_i} & (\rho_i^2+1)\sin\theta_i & -(\rho_i^2+1)\cos\theta_i & 0 \\ \frac{\sin\theta_i}{Z_i\rho_i} & -\frac{\cos\theta_i}{Z_i\rho_i} & 0 & \frac{\cos\theta_i}{\rho_i} & \frac{\sin\theta_i}{\rho_i} & -1 \end{bmatrix} \tag{1.16}$$

with $\rho_i = \sqrt{x_i^2 + y_i^2}$ and $\theta_i = \operatorname{atan2}(y_i, x_i)$. The advantage of this formulation is that when large rotations around the optical axis are to be performed, retraction issues are not present since the translation along the focal axis is now decoupled from the rotation normal to the plane (third and last columns of the interaction matrix).

Still with the objective of determining a set of features which guarantees good decoupling properties, planar image moments can also be considered [Cha04; TC05]. Given an

object whose projection on the image plane is described by the dense set $\mathcal{O}$, its moments can be defined as:

$$m_{i/j} = \iint_{\mathcal{O}} x^i y^j \, dx \, dy \tag{1.17}$$

Similarly, for a finite set of N points, the discrete moments of order $i + j$ are expressed as:

$$m_{i/j} = \sum_{k=1}^{N} x_k^i \, y_k^j \tag{1.18}$$

The interaction matrix can be obtained after differentiation by considering the feature model of each point (x_k, y_k). In addition, under the planar assumption it is possible to express the inverse depth as a function of the image coordinates as in $\frac{1}{Z_k} = Ax_k + By_k + C$, wherein A, B and C are the parameters that characterize the 3D plane to which the observed shape belongs, which can be estimated online [SGC15] or otherwise approximated. On top of these features, more complex ones can be obtained, such as a normalized area of the shape or the main axis of orientation. Interestingly, it was shown that when the object plane is parallel to the projection plane of the sensor ($A = B = 0$) it is possible to build a feature set based on moments whose interaction matrix features good decoupling properties, in which each feature almost controls a separate degree of freedom and thus leading to satisfactory sensor motions in 3D.

1.2.2 Efficient Second Order Minimization

The retraction issues obtained when servoing from image points can be explained by the highly nonlinear map between the operational and sensor space. In particular, the proportional control law (1.9) will attempt to move each feature along a straight line from the initial to the target position. Despite this being the shortest path in the feature space, it can lead to relative motions of the sensor in SE(3) which do not necessarily correspond to the geodesic connecting the initial and goal poses.

A nice explanation for this issue was provided by [Mal04], which compares different control strategies to minimization algorithms. More specifically, the proportional control law based on the pseudo-inverse of the interaction matrix was compared to the Gauss-Newton minimization algorithm, which provides good performances in presence of mild nonlinearities or in proximity of the desired configuration.

Additionally, still based on the parallelism control/minimization, new schemes were proposed with the objective of reaching the desired configuration by following a shorter path. They consist in using, inside the proportional control law (1.9), one of the following:

$$\widehat{\mathbf{J}_s^+} = \frac{\left(\hat{\mathbf{J}}_s\right)^+ + \left(\hat{\mathbf{J}}_{s^\star}\right)^+}{2} \tag{1.19a}$$

$$\widehat{\mathbf{J}_s^+} = \left(\frac{\hat{\mathbf{J}}_s + \hat{\mathbf{J}}_{s^\star}}{2}\right)^+ \tag{1.19b}$$

in which $\hat{\mathbf{J}}_s$ represents an estimation of the feature Jacobian evaluated from the current measured features, while $\hat{\mathbf{J}}_{s^\star}$ is calculated by replacing the desired features in the interaction matrix.

A similar philosophy led to the development of a Hessian-based control scheme [LM04], while an extension of (1.19b) was proposed in [MC08]. Finally, in [TM08; TM10] some limitations of the approaches proposed in [Mal04] were discussed, in particular focusing on the case of large translations and rotations. In particular, it was shown that the control laws have to be adjusted by integrating information related to the displacement to be performed by the sensor from the current to the desired location.

1.3 Dealing with Constraints

1.3.1 Weighted control

One first way that can be used to deal with constraints is to consider them as additional tasks in the control law. As an example, if visibility of an image point is to be ensured, the task could consist in moving the feature towards the center of the image. More generically, the idea is to add a task in the form $\mathbf{J}_c\dot{\boldsymbol{q}} = \dot{\boldsymbol{e}}_c$ in addition to the original servoing task (1.8). It must be noted that this additional task cannot be permanently active since it could otherwise interfere with the main one, *e.g.*, by introducing local minima even when the constraint would be satisfied at the goal configuration. One solution is thus to dynamically activate or deactivate the task by means of a proper weighting matrix. In other words,

the new control law would become:

$$\dot{\boldsymbol{q}} = \mathbf{J}_a^+ \dot{\boldsymbol{e}}_a = \begin{bmatrix} \mathbf{J}_s \\ \mathbf{W}_c \mathbf{J}_c \end{bmatrix}^+ \begin{bmatrix} \dot{\boldsymbol{s}}^\star - \lambda \boldsymbol{e}_s \\ \mathbf{W}_c \dot{\boldsymbol{e}}_c \end{bmatrix} \tag{1.20}$$

in which the pair $\mathbf{J}_a$, $\dot{\boldsymbol{e}}_a$ represents the "augmented task" and $\mathbf{W}_c$ is a diagonal weighting matrix with non-negative entries. In principle, when far away from the constraints it suffices to select $\mathbf{W}_c = \mathbf{O}$, leading to the original control law (1.9) Increasing the weights in $\mathbf{W}_c$ allows instead to activate the corresponding constraints.

In [RMC06] it was proposed to define the weights using the activation function:

$$w(e) = \begin{cases} 0 & e \leq 0 \\ \frac{1}{2}\left(1 + \tanh\left(\frac{1}{1-e/\beta} - \frac{\beta}{e}\right)\right) & 0 < e < \beta \\ 1 & e \geq \beta \end{cases} \tag{1.21}$$

in which β is a scaling parameter that regulates how sharply constraints activate. Additionally, a set of weights were also added to the main task. These value were chosen complementary to those associated to constraints, in such a way that when a constraint is fully activated then the weight associated to the corresponding feature is reduced to zero.

A similar concept is used in [KC14]. The weight of the main task is kept equal to 1, while the weights associated to the constraints are modulated according to a hyperbolic profile:

$$w(e) = \begin{cases} 0 & e \leq e_+ \\ \frac{e-e_+}{e_{max}-e} & e_+ < e < e_{max} \end{cases} \tag{1.22}$$

wherein e_{max} is the maximum allowed value for the quantity e, while e_+ represents a "soft maximum" in which the constraint should activate. In addition, means to escape from local minima are provided, as well as strategies to smooth out the behavior of the controlled system when moving between multiple active constraints.

Finally, it must be noted that these approaches can introduce discontinuities in the control laws. In fact, the pseudo-inverse operation is not continuous under rank changes, and thus irregular control signals will be obtained if $\mathbf{J}_a$ changes rank due to the (de)activation of one or more constraints. This can be a major problem in systems which are redundant with respect to the main task, in which the additional degrees of freedom are often used

precisely to satisfy constraints such as joint limits avoidance. Nonetheless, the inversion operator proposed in [MRC09] can be used to solve (1.20) while ensuring continuity. Such operator behaves exactly like the pseudo-inverse when a constraint is either fully active or inactive. Furthermore, it can guarantee local asymptotic stability and the resulting controller can remove oscillation that arise when near to constraint boundaries due to repeated activation and deactivation of the weights.

1.3.2 Hierarchical control

If the system presents redundancy with respect to the target task, constraints can be enforced using the additional degrees of freedom. In fact, one can use the redundancy framework [SS91] to execute a secondary task of lower priority while ensuring that the main task is executed as expected without any perturbation. In practice, an extension of (1.9) can be written as:

$$\dot{\boldsymbol{q}} = \widehat{\mathbf{J}_{\boldsymbol{s}}^{+}} \left(\dot{\boldsymbol{s}}^{\star} - \lambda \boldsymbol{e}_{\boldsymbol{s}} \right) + \mathbf{P}_{\mathbf{J}_{\boldsymbol{s}}} \, \dot{\boldsymbol{q}}' \tag{1.23}$$

in which $\mathbf{P}_{\mathbf{J}_{\boldsymbol{s}}} = \mathbf{I} - \mathbf{J}_{\boldsymbol{s}}^{+} \mathbf{J}_{\boldsymbol{s}}$ is the projector into the kernel of $\mathbf{J}_{\boldsymbol{s}}$, *i.e.*, such that $\mathbf{J}_{\boldsymbol{s}} \mathbf{P}_{\mathbf{J}_{\boldsymbol{s}}} = \mathbf{O}$, and $\dot{\boldsymbol{q}}'$ is an arbitrary vector. If the kernel of $\mathbf{J}_{\boldsymbol{s}}$ is not empty, then $\dot{\boldsymbol{q}}'$ can be chosen in order to enforce existing constraints without affecting the desired features dynamics.

Oe possible application of such hierarchical approach is joint limits avoidance [CM01], using the so-called gradient projection method [SEB91]. In this context, a cost function $V(\boldsymbol{q})$ is defined, which increases when approaching the constraint boundary and stationary while sufficiently far away from it. Denoting with q_i each coordinate of the joint position vector, such a function can be written as:

$$V(\boldsymbol{q}) = \sum_i \alpha_i w_i^2 \qquad w_i = \begin{cases} q_i^- - q_i & q_i < q_i^- \\ 0 & q_i^- \leq q_i \leq q_i^+ \\ q_i - q_i^+ & q_i > q_i^+ \end{cases} \tag{1.24}$$

with q_i^- and q_i^+ are soft activation limits for the i-th joint, while the coefficients α_i can be modulated online by a proper algorithm. The gradient of the cost function, $\frac{\partial V}{\partial \boldsymbol{q}}^T$, can then be used as an artificial repulsive potential to steer the joint configuration away from the limits by setting $\dot{\boldsymbol{q}}' = -\frac{\partial V}{\partial \boldsymbol{q}}^T$.

The main issue with the gradient projection method is that it allows constraints to be

included only as low-priority tasks, and thus there is no guarantee that they can always be satisfied during the control. To overcome this issue, the idea of using a hierarchy of tasks was expanded in [MC07]. The main task is decomposed in a multitude of sub-tasks, each one thus requiring a lower number of degrees of freedom, and constraints are still handled at the lowest priority using the gradient projection method. However, contrarily to existing approaches, two supervisory controllers are added as well, which can respectively remove or add tasks in the stack. The first one is required to check if, due to the current set of active tasks, the produced control law will lead to violation of any constraint. If this is the case, the auxiliary controller selects the active task which is responsible for the predicted constraint infringement and deactivates it. The second controller is in charge of determining when it is safe to re-activate a task that had been previously removed from the stack. Since each task is responsible for controlling only few degrees of freedom, the system will still tend to evolve towards the goal configuration thanks to the remaining active tasks. In addition, by proper modification of the control law (1.9) it is possible to make the computed command continuous even during insertion or removal of a task.

Even though [MC07] introduced substantial improvements in handling of constraints, they are still managed at the lowest priority level. It is thus essential to properly decompose the main task in smaller ones so as to be able to dynamically change the stack and allow constraints to be taken into account. The method proposed in [MKK09] is based on the continuous inverse [MRC09] detailed before, extending its use to a stack of tasks. It allows to add unilateral constraints at arbitrary priority levels, and not necessarily at the lowest ones.

A different approach was instead considered in [MC10], in which a new projection operator was proposed. Constraints are still handled using the gradient projection method, and are thus still located at a lower priority with respect to the main task. However, the main task is modified: rather than acting directly on the feature error $\boldsymbol{e_s}$, the control objective becomes to steer its norm $\|\boldsymbol{e_s}\|$ to zero. This task requires a single degree of freedom, thus leaving much more flexibility to satisfy the secondary task, *i.e.*, constraints satisfaction. The new norm nullification task is characterized by a new Jacobian $\mathbf{J}_{\|\boldsymbol{e_s}\|}$ whose pseudo-inverse takes the following expression:

$$\mathbf{J}^{+}_{\|\boldsymbol{e_s}\|} = \frac{\|\boldsymbol{e_s}\|}{\boldsymbol{e_s}^T \mathbf{J}_s \mathbf{J}_s^T \boldsymbol{e_s}} \mathbf{J}_s^T \boldsymbol{e_s} \tag{1.25}$$

It must be noted that this matrix approaches a singularity when $\mathbf{J}_s^T \boldsymbol{e_s} \to \mathbf{0}$, which hap-

pens either when the main task is completed ($\boldsymbol{e_s} = \mathbf{0}$) or when the system reaches a local minimum for the servoing task, *i.e.*, when $\boldsymbol{e_s} \in \ker\left(\mathbf{J}_s^+\right) = \ker\left(\mathbf{J}_s^T\right)$. It is therefore proposed to use the new projection strategy in the beginning of the task (when the main task error is still significant), and to smoothly switch to the classical projection method once a neighborhood of the target configuration has been reached.

1.3.3 Quadratic Programming

The classical control law based on the pseudo-inverse of the interaction matrix can be viewed as an unconstrained, multi-objective, quadratic minimization. More specifically, it can be shown that (1.9) is equivalent to solving:

$$\min_{\dot{\boldsymbol{q}}} \left\|\mathbf{J}_s\dot{\boldsymbol{q}} - (\dot{\boldsymbol{s}}^\star - \lambda \boldsymbol{e_s})\right\|^2 \quad \succ \quad \min_{\dot{\boldsymbol{q}}} \left\|\dot{\boldsymbol{q}}\right\|^2 \tag{1.26}$$

in which $a \succ b$ means that the minimization problem b is of secondary importance with respect to a. The controller thus evaluates the command that achieves at best the main task (1.8), also featuring the least magnitude if possible.

Starting from this parallelism, it would thus be possible to consider the control problem explicitly as a quadratic minimization. This opens to the possibility of inserting constraints in the minimization, as long as they can be described as a set of inequalities that are linear in the control input, and to obtain the commands using any Quadratic Programming (QP) algorithm. As an example, in [Agr+17] constraints of this type are introduced to ensure visibility and avoid occlusion on humanoid robots, while in [PM+18] they are used to ensure collision avoidance for an autonomous vehicle performing a parking maneuver.

The illustrated concept allows to consider a single task subject to one set of constraints, which are all to be satisfied at the solution. However, it is possible to combine the concept of a stack of tasks and the quadratic minimization formulation to create a more complex control scheme. More specifically, the methodology proposed in [KLW11] extends the notion of task to include inequalities relations. At each priority level, a new quadratic problem is formed, in which the objective is to satisfy a set of (in)equality tasks as much as possible, while not influencing the tasks coming from higher priority levels. Constraints can thus be incorporated in the control scheme with a given priority, an not necessarily at the last level. The method is further refined in [EMW14], in which subsequent quadratic problems are formulated efficiently using a change of variable. The basic idea is that the projector $\mathbf{P}_{\mathbf{J}_s}$ in (1.23) can be replaced with an orthogonal basis of the kernel of $\mathbf{J}_s$,

denoted as $\mathbf{Z}_{\mathbf{J}_s}$, and such that $\mathbf{J}_s\mathbf{Z}_{\mathbf{J}_s}$. However, while the projector is a square matrix, the orthogonal basis has a smaller number of columns in comparison. The new hierarchical control law can be written using the orthogonal basis as:

$$\dot{\boldsymbol{q}} = \widehat{\mathbf{J}_s^+} \left(\dot{\boldsymbol{s}}^\star - \lambda \boldsymbol{e}_s \right) + \mathbf{Z}_{\mathbf{J}_s} \boldsymbol{y} \tag{1.27}$$

The difference with respect to (1.23) is that, while $\dim(\dot{\boldsymbol{q}}') = \dim(\dot{\boldsymbol{q}})$, $\boldsymbol{y}$ is a lower-dimensional vector with only $\dim(\dot{\boldsymbol{q}}) - \operatorname{rank}(\mathbf{J}_s)$ components. Therefore, after a priority level is solved in the stack, the following problem can be parameterized in a lower-dimensional variable. Starting from this simple concept, the complete hierarchical approach is developed and it shown that it greatly improves the performances of the controller.

1.4 Conclusions

In this chapter, the fundamental notions of visual servo control have been introduced, with a focus on Image Based Visual Servoing. As discussed, servoing schemes that use image points as features are simple to implement, but can lead to few problems. In particular, when large displacements are involved the sensor motion can be longer than necessary, thus leading to larger time to convergence. One way to improve the performances can be found in the use of better features that lead to higher decoupling in the control degrees of freedom, or by using efficient laws which attempt to steer the sensor along geodesic paths.

Two issues are however still open. First of all, the control laws detailed so far work under the assumption that the evaluated joint velocity commands can be readily tracked by the low-level controllers of the manipulator. This hypothesis is reasonable when the robot moves at relatively low speed. However, in order to obtain good performances at high speed it might be necessary to employ high gains to achieve the required velocity tracking performances. This can be problematic, since higher gains will lead to noise amplification. This problem is thus briefly investigated in the next chapter.

A second issue is related to the exponential decrease in velocity imposed by feedback control laws. As the sensor approaches the desired configuration, the speed of the manipulator can decrease significantly. Faster convergence can be obtained using larger gains, but at the cost of having very large velocities and accelerations in the beginning of the task. One way to tackle this problem, as proposed in Chapter 3, can be found in the use

of Model Predictive Control, in which the control inputs can be optimized in order to achieve high velocities for a longer period, but also with the ability of decelerating in a controlled manner once the servoing task is completed. Furthermore, the advantage of considering a predictive control strategy is that it allows to accomplish two further goals using a unified approach: first of all, not only the motions can be faster, but also shorter thanks to the ability of predictive controllers to foresee future states and thus optimize the sensor motions. Secondly, unilateral constraints can readily be included in the optimization process. It is thus possible to ensure not only a faster convergence, but also to guarantee feasibility of the motion.

Chapter 2

Visual Servoing using Second-Order Models

The control laws classically used in visual servoing are formulated at the velocity level. It is generally assumed that low-level controllers are able to properly track velocity references generated by the visual controllers, thus assuming that dynamic effects are negligible. This is a fair assumption when the task to be performed is executed at medium-low speed. However, dynamic effects increase with the velocity [CG96] and therefore, as faster motions are involved, new and more accurate models have to be considered.

One way to enhance the performances of the servoing schemes is to establish a direct link between the evolution of the features in the image and the dynamic parameters of the controlled system. This naturally leads to the use of second-order models which express the acceleration of the features as a function of both the velocity and the acceleration of the sensor. The advantage is that no low-level velocity controllers are needed anymore, and the desired features evolution can be obtained by computing directly the required motor torques by taking into account the dynamic model of the manipulator.

This chapter focuses precisely on this topic, in particular on the study of second-order acceleration models. First of all, the next section includes several related works that explored the use of features acceleration and attempted to exploit it in the context of dynamic control. Afterwards, in Section 2.2 a procedure to derive features acceleration models is detailed, and it is shown how they can be used to directly link the feature space with the dynamics of manipulators. Section 2.2.3 proposes different dynamic controls. The first one does not need the use of acceleration models and relies instead only on velocity information. It will be used later for comparison with another type of controller that is instead based on the proposed second-order models. The comparison, performed in simulation, is presented in Section 2.3. In particular two variations of the velocity-based controller are tested against the control law based on second-order models, and the benefits of using this approach are discussed.

2.1 Related Works

The idea of using acceleration information in a sensor-based control scheme can be backtracked to almost the origins of visual servoing itself. Indeed, few early works can be found in the literature which propose controllers that are not based solely on the classical interaction model. The idea of computing commands using a PD controller based on second-order models of image features and incorporating it into a CTC was already mentioned in [ECR92], even though it was then proposed to consider a kinematic control under the assumption of low accelerations and the use of high-gain joint velocity controllers.

Even when the dynamics of the manipulator is disregarded, integrating higher order information in kinematic control laws can still be beneficial, as illustrated in [MBG96]. The study shows that using both the position and velocity of a point as feature can enhance the performances of the controller in presence of relatively large processing delays.

A series of works by Dahmouche *et al.* [Dah+09; Dah+10; Dah+12] focuses on the use of spatio-temporal point features whose evolution depends not only on the current relative twist $\mathbf{v}$ but also on its derivative. More precisely, these works aim at performing simultaneous pose and velocity estimation of the end-effector of a parallel robot using a eye-to-hand setup. To decrease significantly the cost of image acquisition and processing, features are detected by analyzing specific ROIs (Region of Interests) in a sequential manner, and thus at different time instants. As a consequence, the value of each feature is dependent not only on the position of the corresponding point, but also on the time at which the ROI is captured and the object velocity. In particular, using a first order linearization, [Dah+09] estimates the position of a measured feature as $\boldsymbol{s}_i = \boldsymbol{p}_i + \Delta t_i \mathbf{L}_i \mathbf{v}$, in which $\boldsymbol{p}_i$ is the point that is being measured and the second component represents the displacement of the point due to the time delay of the sequential acquisition. The time derivative of each feature thus depends both on the relative velocity and acceleration of the sensor with respect to the object. In [Dah+10] a virtual visual servoing approach [MC02b] is used to obtain an estimation of the pose and velocity of the end effector, under the assumption of piece-wise constant acceleration. This estimation is then used in a CTC scheme with the objective of tracking a given trajectory at high speed, with the experiments showing that the vision-based approach outperforms model-based schemes based only on proprioceptive information. Furthermore, in [Dah+12] a further scheme was proposed consisting in a Image Based Visual Servoing (IBVS) scheme formulated directly

at the dynamic level, and in [Ozg+13] the same approach is extended to control a parallel mechanism from legs observations rather than using image points.

An alternative approach to directly express the dynamics of the features at the dynamic level can be found in [HO99]. By considering a bijective map between image features and robot state, a Lagrangian is written directly in terms of the feature vector $\boldsymbol{s}$ and its derivative $\dot{\boldsymbol{s}}$. In particular, the center and radius of a circular shape are used to control the motion of a simplified miniaturized blimp.

Yet another approach is used in [Tho+14], which considers a simplified drone model moving in the sagittal plane. The position of the drone is controlled using two features projected on a virtual image plane parallel to the ground. The authors show that these features constitute a flat output for the system, allowing to evaluate the thrust and torque needed to pilot the drone directly from the feature error and its derivatives.

A second-order IBVS scheme using the joint acceleration as control input was proposed in [KX12] for a non-redundant manipulator observing four image points, and was later extended to take into account the dynamic model of the robot in a CTC fashion [KXM14]. Using the same formalism, [MKX14] proposes an acceleration control for a serial robot equipped with a stereo-vision system on its end-effector.

A hierarchical multi-objective control scheme was proposed in [Van+16], in which a 7-dof manipulator performs a hybrid force-vision control. The tasks are formulated directly at the dynamic level and the priorities in the hierarchy are respected using a projection strategy similar to that of [MC07] and [SS91].

Finally, whole-body control of a humanoid robot was proposed in [Agr+17], which has been already mentioned in the previous chapter. In this case, an acceleration control command is computed using QP to minimize three weighted objectives under multiple constraints.

2.2 Second-Order Visual Servoing & Dynamics

This section focuses on the theory behind modeling and feedback control via second-order model. In particular, starting from the kinematic relation described by (1.1), it is shown in the next section how to obtain an expression that provides the second derivative of the features in function of the spatial acceleration of the sensor. It is then detailed how these relations can be linked with the dynamic model of a serial manipulator, and how the new models can be used for visual servo control.

Since the final objective is to control a redundant manipulator, for which the visual task can constrain at most 6 dof, it is shown how redundancy can (and should) be resolved in the new schemes.

2.2.1 Second-Order Visual Servoing

The class of models presented in the sequel considers the second-order time derivative of the features and how it depends on the acceleration of sensors, in addition to their velocity. In some related works the derivation of the second-order model was achieved by considering the geometric meaning of the considered features and by evaluating $\ddot{\boldsymbol{s}}$ from such geometric definition. As an example, let's consider a stationary 3D point ${}^c\boldsymbol{p}$ observed from a moving camera, for which the interaction matrix writes as [CH06]:

$$\mathbf{L}_{{}^c\boldsymbol{p}} = \begin{bmatrix} -\mathbf{I} & [{}^c\boldsymbol{p}]_\times \end{bmatrix} \tag{2.1}$$

It is possible to express the second derivative of ${}^c\boldsymbol{p}$ from the laws of classical mechanics, *e.g.*, as done in [KX12]. In particular, using the well-known rule of composition of accelerations in non-inertial frames [Sic+10]:

$${}^c\ddot{\boldsymbol{p}} = -\boldsymbol{a} + {}^c\boldsymbol{p} \times \dot{\boldsymbol{\omega}} + 2\boldsymbol{\omega} \times \boldsymbol{v} + ({}^c\boldsymbol{p} \times \boldsymbol{\omega}) \times \boldsymbol{\omega} \tag{2.2}$$

in which $\boldsymbol{a}$ and $\dot{\boldsymbol{\omega}}$ are the linear and angular acceleration of the sensor expressed in the coordinate system of the sensor itself. Rearranging the terms, one can realize that ${}^c\ddot{\boldsymbol{p}}$ can be expressed using the interaction matrix of the 3D point as

$${}^c\ddot{\boldsymbol{p}} = \begin{bmatrix} -\mathbf{I} & [{}^c\boldsymbol{p}]_\times \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + 2\boldsymbol{\omega} \times \boldsymbol{v} + ({}^c\boldsymbol{p} \times \boldsymbol{\omega}) \times \boldsymbol{\omega} = \mathbf{L}_{{}^c\boldsymbol{p}}\mathbf{a} + \boldsymbol{h}_{{}^c\boldsymbol{p}} \tag{2.3}$$

In the formula, $\mathbf{a}$ is the concatenation of the linear and angular sensor accelerations, *i.e.*, $\mathbf{a}^T = \begin{bmatrix} \boldsymbol{a}^T & \dot{\boldsymbol{\omega}}^T \end{bmatrix}$, and the affine term $\boldsymbol{h}_{{}^c\boldsymbol{p}}$ groups the quantities that depend solely on the kinematic screw $\mathbf{v}$ and on the feature itself. As it will be soon shown, this structure is common to other features as well.

A similar procedure could be followed to evaluate the second-order models of any feature, such as lines and planes [Van16]. However, the procedure can become rather long and error-prone. Instead, we propose in the following a general approach that we believe facilitates the derivation of these models, and in addition provides a specific structure for

the affine term $\boldsymbol{h}$. First of all, the interaction model (1.1) is directly differentiated:

$$\ddot{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}} \frac{d}{dt}\mathbf{v} + \dot{\mathbf{L}}_{\boldsymbol{s}}\, \mathbf{v} \tag{2.4}$$

Assuming an eye-in-hand setup, the derivative of the kinematic screw writes as follows:

$$\frac{d}{dt}\mathbf{v} = \mathbf{a} + \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{\omega} \\ \mathbf{0} \end{bmatrix} \tag{2.5}$$

The additional term is a consequence of the fact that the camera frame is not inertial with respect to the object. In order to proceed, it is now necessary to differentiate $\mathbf{L}_{\boldsymbol{s}}$. For this purpose, it is convenient to consider the interaction matrix as a stack of vectors $\boldsymbol{\ell}_i \in \mathbb{R}^6$ $(i = 1, \cdots, n_s)$. In particular, each of these vectors represents one the rows of $\mathbf{L}_{\boldsymbol{s}}$:

$$\mathbf{L}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{\ell}_1^T \\ \cdots \\ \boldsymbol{\ell}_{n_s}^T \end{bmatrix} \tag{2.6}$$

Given this structure of the interaction matrix, the required quantity $\dot{\mathbf{L}}_{\boldsymbol{s}}$ readily follows from the total differentiation rule:

$$\dot{\mathbf{L}}_{\boldsymbol{s}} = \begin{bmatrix} \left(\frac{\partial \boldsymbol{\ell}_1}{\partial \boldsymbol{s}}\dot{\boldsymbol{s}} + \frac{\partial \boldsymbol{\ell}_1}{\partial \boldsymbol{z}}\dot{\boldsymbol{z}}\right)^T \\ \cdots \\ \left(\frac{\partial \boldsymbol{\ell}_{n_s}}{\partial \boldsymbol{s}}\dot{\boldsymbol{s}} + \frac{\partial \boldsymbol{\ell}_{n_s}}{\partial \boldsymbol{z}}\dot{\boldsymbol{z}}\right)^T \end{bmatrix} \tag{2.7}$$

The derivative $\dot{\boldsymbol{s}}$ can be obtained from the classical interaction model (1.1). In the following it is assumed for simplicity that an expression for $\dot{\boldsymbol{z}}$ is available, and that furthermore it takes the form $\dot{\boldsymbol{z}} = \mathbf{L}_{\boldsymbol{z}}\boldsymbol{v}$. Under this assumption it is thus possible to write:

$$\dot{\mathbf{L}}_{\boldsymbol{s}} = \begin{bmatrix} \left(\frac{\partial \boldsymbol{\ell}_1}{\partial \boldsymbol{s}}\mathbf{L}_{\boldsymbol{s}}\mathbf{v} + \frac{\partial \boldsymbol{\ell}_1}{\partial \boldsymbol{z}}\mathbf{L}_{\boldsymbol{z}}\mathbf{v}\right)^T \\ \cdots \\ \left(\frac{\partial \boldsymbol{\ell}_{n_s}}{\partial \boldsymbol{s}}\mathbf{L}_{\boldsymbol{s}}\mathbf{v} + \frac{\partial \boldsymbol{\ell}_{n_s}}{\partial \boldsymbol{z}}\mathbf{L}_{\boldsymbol{z}}\mathbf{v}\right)^T \end{bmatrix} = \begin{bmatrix} \mathbf{v}^T\mathbf{H}_1 \\ \cdots \\ \mathbf{v}^T\mathbf{H}_{n_s} \end{bmatrix} \tag{2.8}$$

in which the matrices $\mathbf{H}_i \doteq \left(\frac{\partial \boldsymbol{\ell}_i}{\partial \boldsymbol{s}}\mathbf{L}_{\boldsymbol{s}} + \frac{\partial \boldsymbol{\ell}_i}{\partial \boldsymbol{z}}\mathbf{L}_{\boldsymbol{z}}\right)^T$ are a function of the features $\boldsymbol{s}$ and the

parameters $\boldsymbol{z}$ only. Injecting this result and (2.5) into (2.4) leads to:

$$\ddot{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}}\mathbf{a} + \mathbf{L}_{\boldsymbol{s}} \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{\omega} \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{v}^T\mathbf{H}_1 \\ \cdots \\ \mathbf{v}^T\mathbf{H}_{n_s} \end{bmatrix} \mathbf{v} \tag{2.9}$$

The second term in the right-hand side can be rearranged in a nicer form that matches the structure of the last term. By denoting with $\boldsymbol{\ell}_i^{(v)} \in \mathbb{R}^3$ the vector containing the first three elements of $\boldsymbol{\ell}_i$, the second term in the last equation can be written as:

$$\mathbf{L}_{\boldsymbol{s}} \begin{bmatrix} \boldsymbol{v} \times \boldsymbol{\omega} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\ell}_1^T (\boldsymbol{v} \times \boldsymbol{\omega}) \\ \cdots \\ \boldsymbol{\ell}_{n_s}^T (\boldsymbol{v} \times \boldsymbol{\omega}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\omega}^T (\boldsymbol{\ell}_1 \times \boldsymbol{v}) \\ \cdots \\ \boldsymbol{\omega}^T (\boldsymbol{\ell}_{n_s} \times \boldsymbol{v}) \end{bmatrix} = \begin{bmatrix} \mathbf{v}^T\boldsymbol{\mathcal{L}}_1\mathbf{v} \\ \cdots \\ \mathbf{v}^T\boldsymbol{\mathcal{L}}_{n_s}\mathbf{v} \end{bmatrix} \tag{2.10}$$

in which the circular-shift invariance of the mixed product has been used in the second passage, and the matrices $\boldsymbol{\mathcal{L}}_i \in \mathbb{R}^{6\times6}$ in the last expression are defined as:

$$\boldsymbol{\mathcal{L}}_i \doteq \begin{bmatrix} \mathbf{O} & \mathbf{O} \\ \left[\boldsymbol{\ell}_i^{(v)}\right]_\times & \mathbf{O} \end{bmatrix} \tag{2.11}$$

Finally, using (2.9), (2.10) and by defining

$$\mathbf{G}_i \doteq \frac{(\mathbf{H}_i + \boldsymbol{\mathcal{L}}_i) + (\mathbf{H}_i + \boldsymbol{\mathcal{L}}_i)^T}{2} \tag{2.12}$$

leads to the sought expression of second-order models:

$$\ddot{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}}\mathbf{a} + \begin{bmatrix} \mathbf{v}^T\mathbf{G}_1\mathbf{v} \\ \cdots \\ \mathbf{v}^T\mathbf{G}_{n_s}\mathbf{v} \end{bmatrix} = \mathbf{L}_{\boldsymbol{s}}\mathbf{a} + \boldsymbol{h}_{\boldsymbol{s}} \tag{2.13}$$

It must be noted that, as already mentioned, the second-order model of the 3D point developed in the beginning of this section takes a similar form, the only difference being the $\boldsymbol{h}_{^c\boldsymbol{p}}$ vector which was not expressed as a set of quadratic forms in (2.3). Appendix A.1 includes the second-order model of 3D points rewritten according to (2.13), which, after some development, can be shown to be equivalent to (2.3). More in general, Appendix A includes the second-order models of few other features, such as normalized image points.

The main interest in the illustrated procedure is that, in order to get the second-order model of any feature, the only required work is the evaluation of $\frac{\partial \boldsymbol{\ell}_i}{\partial \boldsymbol{s}}$ and $\frac{\partial \boldsymbol{\ell}_i}{\partial \boldsymbol{z}}$. In addition,

the constructive procedure shows that the affine term $\boldsymbol{h_s}$ consists in a series of quadratic forms, with $\mathbf{G}_i \in \mathbb{R}^{6\times6}$ depending on $\boldsymbol{s}$ and $\boldsymbol{z}$. This can be useful, *e.g.*, if the partial derivatives of $\ddot{\boldsymbol{s}}$ are to be computed.

To conclude this section, it is shown how a second-order model can be obtained from another set of features for which an acceleration model has already been obtained. Let's consider a set of features $\boldsymbol{s}_b$ defined by some twice differentiable function of another set $\boldsymbol{s}_a$, *i.e.*, $\boldsymbol{s}_b = \boldsymbol{\psi}(\boldsymbol{s}_a)$. Using the chain rule of differentiation, the second time derivative of $\boldsymbol{s}_b$ can be expressed as:

$$\ddot{\boldsymbol{s}}_b = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{s}_a}\ddot{\boldsymbol{s}}_a + \left(\frac{d}{dt}\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{s}_a}\right)\dot{\boldsymbol{s}}_a \tag{2.14}$$

The first and second derivatives of $\boldsymbol{s}_a$ can be expressed by their respective models (1.1) and (2.13). Substituting these expressions leads to

$$\ddot{\boldsymbol{s}}_b = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{s}_a}\mathbf{L}_a\mathbf{a} + \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{s}_a}\boldsymbol{h}_a + \left(\frac{d}{dt}\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{s}_a}\right)\mathbf{L}_a\mathbf{v} \tag{2.15}$$

in which, by comparison with (2.13), it is easy to identify $\boldsymbol{h}_b$ as:

$$\boldsymbol{h}_b = \frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{s}_a}\boldsymbol{h}_a + \left(\frac{d}{dt}\frac{\partial \boldsymbol{\psi}}{\partial \boldsymbol{s}_a}\right)\mathbf{L}_a\mathbf{v} \tag{2.16}$$

However, directly using this expression does not provide explicitly the form of the **G**-matrices. To obtain them it is necessary to perform few additional steps, leading to the final result:

$$\mathbf{G}_{b,i} = \mathbf{L}_a^T\frac{\partial^2\psi_i}{\partial \boldsymbol{s}_a^2}\mathbf{L}_a + \sum\nolimits_j \frac{\partial \psi_i}{\partial s_{a,j}}\mathbf{G}_{a,j} \tag{2.17}$$

wherein $\mathbf{G}_{b,i}$ is the i-th **G**-matrix of the feature set $\boldsymbol{s}_b$ and $\mathbf{G}_{a,j}$ follows the same notation. Finally, ψ_i is the i-th entry of the vector-valued function $\boldsymbol{\psi}$ and $s_{a,j}$ is the j-th coordinate of the feature set $\boldsymbol{s}_a$.

2.2.2 Robot Kinematics and Dynamics in Sensor Space

As previously mentioned, the obtained second-order model (2.13) is valid under the assumption that the relative motion between the sensor and the object depends solely on the motion of the robot (eye-in-hand setup). In this case, the velocity and acceleration of

the sensor, expressed in the base coordinate system, are given by:

$$^{w}\mathbf{v}_{c} = \mathbf{J}\dot{\boldsymbol{q}} \tag{2.18a}$$

$$^{w}\mathbf{a}_{c} = \mathbf{J}\ddot{\boldsymbol{q}} + \dot{\mathbf{J}}\dot{\boldsymbol{q}} \tag{2.18b}$$

wherein $\mathbf{J}$ and $\dot{\mathbf{J}}$ are the robot Jacobian (as used in (1.6a)) and its derivative. By projecting (2.18a) and (2.18b) on the sensor frame and injecting the result into the second-order model, the feature acceleration can be obtained as a function of the joint velocities and accelerations:

$$\ddot{\boldsymbol{s}} = \mathbf{J}_{\boldsymbol{s}}\ddot{\boldsymbol{q}} + \mathbf{L}_{\boldsymbol{s}}\,{}^{c}\mathbb{T}_{w}\dot{\mathbf{J}}\dot{\boldsymbol{q}} + \boldsymbol{h}_{\boldsymbol{s}} = \mathbf{J}_{\boldsymbol{s}}\ddot{\boldsymbol{q}} + \boldsymbol{h}_{\boldsymbol{q}} \tag{2.19}$$

where $\mathbf{J}_{\boldsymbol{s}}$ is the feature Jacobian (1.6a) defined in the previous chapter.

A further step consists in expressing a direct link between the features and the dynamic model of the robot. First of all, it is recalled here that the Inverse Dynamic Model (IDM) of a robotic system composed of a set of rigid bodies writes as [KD04]:

$$\begin{aligned}\boldsymbol{\tau} &= \mathbf{A}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \mathbf{C}(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \mathbf{F}_{v}\dot{\boldsymbol{q}} + \mathbf{F}_{s}\mathrm{sign}(\dot{\boldsymbol{q}}) + \mathbf{J}^{T}\boldsymbol{f}_{e} \\ &= \mathbf{A}\ddot{\boldsymbol{q}} + \boldsymbol{b} + \mathbf{J}^{T}\boldsymbol{f}_{e}\end{aligned} \tag{2.20}$$

wherein $\boldsymbol{\tau}$ represents joint efforts while $\mathbf{A}$ is the generalized inertia matrix of the system, grouping inertial effects due to masses and inertiae of both rigid bodies and motors. $\mathbf{C}\dot{\boldsymbol{q}}$ represents Coriolis and centripetal efforts, while $\boldsymbol{g}$ gravitational ones. $\mathbf{F}_{v}$ and $\mathbf{F}_{s}$ are diagonal matrices used to model friction, and $\boldsymbol{f}_{e}$ represents an external wrench applied on the end-effector. For brevity, $\boldsymbol{b}$ 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 $\mathbf{A}$ is invertible, which is always true for a serial kinematic chain, the joint accelerations are given by $\ddot{\boldsymbol{q}} = \mathbf{A}^{-1}\left(\boldsymbol{\tau} - \boldsymbol{b} - \mathbf{J}^{T}\boldsymbol{f}_{e}\right)$. In conjunction with (2.19), it leads to:

$$\mathbf{J}_{\boldsymbol{s}}\mathbf{A}^{-1}\boldsymbol{\tau} = \ddot{\boldsymbol{s}} - \boldsymbol{h}_{\boldsymbol{q}} + \mathbf{J}_{\boldsymbol{s}}\mathbf{A}^{-1}\left(\boldsymbol{b} + \mathbf{J}^{T}\boldsymbol{f}_{e}\right) \tag{2.21}$$

which provides the sought relation between motor effort commands and features evolution.

2.2.3 Dynamics based Feedback Control

In this section, few controllers are discussed, which try to evaluate effort commands by considering the dynamics of the maneuvered manipulator with the objective of regulating the feature error $\boldsymbol{e_s} \doteq \boldsymbol{s} - \boldsymbol{s}^\star$ to zero.

A first solution would be to disregard second-order models and to separate the control law in two distinct loops. An outer visual servoing loop evaluates a reference joint velocity using the proportional control law (1.9):

$$\dot{\boldsymbol{q}}^\star = \widehat{\mathbf{J}_s^+} \left(\dot{\boldsymbol{s}}^\star - \lambda \boldsymbol{e_s} \right) \tag{2.22}$$

The goal of the second control loop is then to track such velocity profile. An auxiliary acceleration pseudo-control can be designed for this purpose as:

$$\boldsymbol{w} = \dot{\boldsymbol{q}}_{ff} + \gamma \left(\dot{\boldsymbol{q}}^\star - \dot{\boldsymbol{q}} \right) \tag{2.23}$$

given a positive proportional gain γ. The feed-forward term $\ddot{\boldsymbol{q}}_{ff}$ is added in order to remove tracking delays, and should ideally correspond to the derivative of $\dot{\boldsymbol{q}}^\star$. Since it cannot be obtained analytically, it has to be estimated numerically from (2.22). This auxiliary command can be injected in the IDM, thus giving the control:

$$\boldsymbol{\tau} = \hat{\mathbf{A}} \boldsymbol{w} + \widehat{\boldsymbol{b}} \tag{2.24}$$

in which $\hat{\mathbf{A}}$ and $\widehat{\boldsymbol{b}}$ provide an estimation of $\mathbf{A}$ and $\boldsymbol{b}$ respectively, and $\boldsymbol{f}_e$ was assumed to be null. The main drawback of this control scheme, as it will be shown later in this chapter, is that it requires the feed-forward to be effective. However, since the signal has to be estimated via numerical differentiation, it is a source of noise amplification.

An alternative approach that does not require the use of second order models consists in using the following control law, corresponding to a proportional and derivative feedback with gravity compensation [Kel+00]:

$$\boldsymbol{\tau} = \boldsymbol{g} - \gamma \dot{\boldsymbol{q}} - \lambda \mathbf{J}_s^T \boldsymbol{e_s} \tag{2.25}$$

Intuitively, this control law moves the end-effector of the robot under the action of a virtual force that equals $-\mathbf{L}_s^T \boldsymbol{e_s}$. Moreover, asymptotic stability can be proved rigorously considering the Lyapunov function $\frac{1}{2} \dot{\boldsymbol{q}}^T \mathbf{A} \dot{\boldsymbol{q}} + \frac{\lambda}{2} \boldsymbol{e_s}^T \boldsymbol{e_s}$. However, the resulting sensor trajectories seem less satisfactory and harder to anticipate than those obtained with other

methods.

Finally, control laws based on second-order models can be considered. The rationale behind them is similar to that of first-order proportional controllers, the goal being to enforce an exponential decrease of the error according to the evolution of the autonomous linear system $\ddot{\boldsymbol{e}}_{\boldsymbol{s}} + k_d\dot{\boldsymbol{e}}_{\boldsymbol{s}} + k_p\boldsymbol{e}_{\boldsymbol{s}} = \mathbf{0}$, which is asymptotically stable if positive gains are selected. Perfect linearization will not be achievable in general, but asymptotic stability can nonetheless be proved [KXM14] when using the auxiliary acceleration control

$$\ddot{\boldsymbol{q}}^\star = \widehat{\mathbf{J}_{\boldsymbol{s}}^+}\left(\ddot{\boldsymbol{s}}^\star - k_d\dot{\boldsymbol{e}}_{\boldsymbol{s}} - k_p\boldsymbol{e}_{\boldsymbol{s}} - \widehat{\boldsymbol{h}_{\boldsymbol{q}}}\right) \tag{2.26}$$

Note that in addition to an estimation of the pseudo-inverse of $\mathbf{J}_{\boldsymbol{s}}$, the two quantities $\boldsymbol{h}_{\boldsymbol{q}}$ and $\dot{\boldsymbol{e}}_{\boldsymbol{s}}$ are to be estimated as well. Similarly to the case of the first controller presented in this section, the acceleration command $\ddot{\boldsymbol{q}}^\star$ can be used in conjunction with the IDM to evaluate a set of effort commands to be sent to the actuators.

2.2.4 Dealing with Redundancy

The case in which the robot has more degrees of freedom than those required to perform the servoing task is address separately here. First of all, it is important to recall that the controls returned by the feedback control laws (2.22) and (2.26) are nothing but particular solutions to the following minimization problems:

$$\min_{\dot{\boldsymbol{q}}} \left\| \widehat{\mathbf{J}_{\boldsymbol{s}}}\dot{\boldsymbol{q}} - \dot{\boldsymbol{s}}^\star + \lambda\boldsymbol{e}_{\boldsymbol{s}} \right\| \tag{2.27a}$$

$$\min_{\ddot{\boldsymbol{q}}} \left\| \widehat{\mathbf{J}_{\boldsymbol{s}}}\ddot{\boldsymbol{q}} - \ddot{\boldsymbol{s}}^\star + k_d\dot{\boldsymbol{e}}_{\boldsymbol{s}} + k_p\boldsymbol{e}_{\boldsymbol{s}} + \widehat{\boldsymbol{h}_{\boldsymbol{q}}} \right\| \tag{2.27b}$$

In particular, as mentioned in the previous chapter, they correspond to the solutions that, in addition to be minimizers of the objectives, have the least norm possible. Assuming that no singularities are encountered, a visual task can constrain up to six degrees of freedom. If the manipulator features $n_q > 6$ joints, it follows that the kernel of $\mathbf{J}_{\boldsymbol{s}}$ is not null, and thus infinite solutions can be found that are compatible with (2.27a) and (2.27b).

Assuming that no secondary task needs to be performed, the control law based on first-order models will move the sensor in the proper configuration and then stop the manipulator in the attained position. In fact, once the visual error has been nullified, the minimum norm solution solving (2.27a) is trivially $\dot{\boldsymbol{q}} = 0$.

On the contrary, when the control input is an acceleration, the minimum norm solution returned by (2.27b) does not necessarily correspond to a stable manipulator configuration. In fact, to a null acceleration – the minimum norm solution – corresponds a constant and possibly non-zero velocity. The reality is slightly more complicated due to the nonlinearities in the geometric and kinematic model of the manipulator, which will also cause the kernel of $\mathbf{J}_{\boldsymbol{s}}$ to change continuously. Nonetheless, without modifications of the proposed control law the joint velocity of the manipulator will not be null in general, and will instead continuously change in order to fall within $\ker(\mathbf{J}_{\boldsymbol{s}})$.

In order to solve the issue at hand, one first solution is to attempt to evaluate the control not as a minimum norm acceleration, but rather by using directly (2.21) to compute the efforts from the desired task [Van+16]. One possible choice in this sense would be to compute the actuation command $\boldsymbol{\tau}$ having the least norm. However, this strategy can again lead to large uncontrolled velocities, and it is generally preferable to use a weighted pseudo-inverse [DO18; Man12]:

$$\boldsymbol{\tau} = (\mathbf{J}_{\boldsymbol{s}})^{+}_{\mathbf{D}} \left(\ddot{\boldsymbol{s}}^{\star} - k_d \dot{\boldsymbol{e}}_{\boldsymbol{s}} - k_p \boldsymbol{e}_{\boldsymbol{s}} - \boldsymbol{h}_{\boldsymbol{q}} + \mathbf{J}_{\boldsymbol{s}} \mathbf{A}^{-1} \boldsymbol{b} \right) \tag{2.28}$$

wherein $(\mathbf{J}_{\boldsymbol{s}})^{+}_{\mathbf{D}} \doteq \mathbf{D}\,(\mathbf{J}_{\boldsymbol{s}}\mathbf{D})^{+}$, with $\mathbf{D} > 0$. If repetitive motions are to be performed, other solutions might be preferable, such as the method proposed in [PPT14] which causes the system to stabilize on a periodic orbit, thus eliminating the chaotic behavior that would result otherwise.

Finally, a last, simple, solution would be actively damp the velocity of the manipulator. This would ensure that, when reaching a fixed visual configuration, the joint velocity of the robot is decreased, eventually stopping the robot once the visual task has been completed. This active velocity minimization objective can be considered as a secondary task in the form

$$\ddot{\boldsymbol{q}} = -k_v \dot{\boldsymbol{q}} \tag{2.29}$$

$k_v > 0$ being a positive gain. Using the redundancy framework to ensure that this task does not perturb the visual one, the final acceleration control law becomes:

$$\ddot{\boldsymbol{q}}^{\star\prime} = \ddot{\boldsymbol{q}}^{\star} - \mathbf{Z}\mathbf{Z}^{T} \left(k_v \dot{\boldsymbol{q}} + \ddot{\boldsymbol{q}}^{\star} \right) \tag{2.30}$$

in which $\ddot{\boldsymbol{q}}^{\star}$ is the acceleration evaluated using (2.26). $\mathbf{Z}$ is an orthogonal basis of the kernel of $\mathbf{J}_{\boldsymbol{s}}$, so that the orthogonal projector is given in the efficient form $\mathbf{Z}\mathbf{Z}^{T}$ [EMW14].

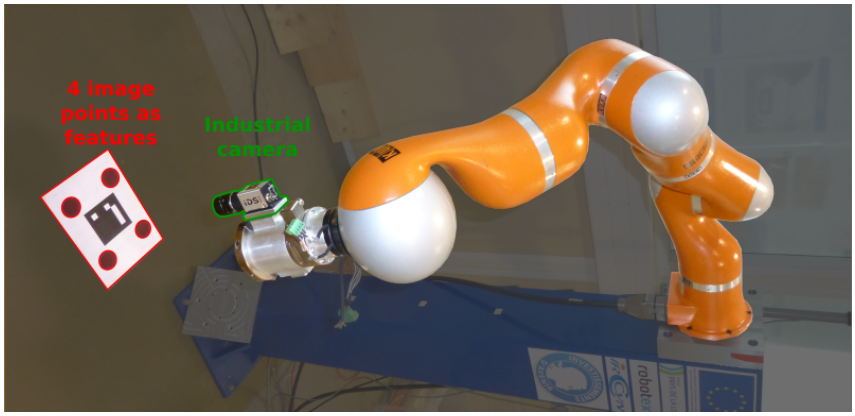


Figure 2.1 – A Kuka LWR4+ manipulator in a eye-in-hand configuration, observing a target with four dots. This same manipulator was simulated in Gazebo to perform the comparison between the different control schemes presented in this section.

Analogously to other schemes, efforts can finally be obtained by injecting into $\ddot{\boldsymbol{q}}^{\star\prime}$ into the IDM.

2.3 Comparison of Velocity and Acceleration-based controllers

In this section, a comparison between two control schemes based only on first-order information and one based on the presented second-order models is proposed. The comparison has been performed in a simulated environment, using Gazebo for physics simulation and ROS for the implementation of controllers and communication. ViSP [MSC05] has been used for Visual Servoing related computations.

The test scenario involves a positioning task from four coplanar points, whose second-order interaction models is reported in Appendix A.2, which are located at the vertices of a square. A perspective camera is mounted on the tip of a 7-dof Kuka LWR4+ robot, as shown in Figure 2.1. A perfectly calibrated sensor with no distortion is assumed for simplicity, and the normalized coordinates of the observed points are used as features. Uncertainties in the Geometric and Dynamic parameters of the robot are not considered in this analysis.

The control period of the simulation is set to 1 ms, and it is assumed that the visual feedback comes from a mixture of image processing and state estimation, and that the unknown parameters $\boldsymbol{z}$ can be estimated as well [DOG07; SGC15]. To take into account

uncertainties due to these operations, normally distributed noise is added to both $\boldsymbol{s}$ and $\boldsymbol{z}$. In addition, the presence of encoders on each motor is simulated by quantization of the readings to a given resolution. More details on noise are presented in the next section.

Three control strategies have been analyzed. The first one, namely First Order (FO), corresponds to a velocity-based scheme as described in the beginning of Section 2.2.3. However, in the computation of the acceleration pseudo-control (2.23) the feed-forward signal $\ddot{\boldsymbol{q}}_{ff}$ is not considered. The second strategy takes this term into account and will be referred to as First Order with Feed-Forward (FOFF).

It must be noted that a problem with the first-order law (2.22) is that it presents a discontinuity in velocity at the very first iteration since the manipulator is initially at rest. In order to guarantee a "smooth start", the computation of $\dot{\boldsymbol{q}}$ in (2.22) was slightly modified, according to the concepts presented in [MC07].

The last controller, denoted as Second Order (SO), evaluates joints accelerations using (2.26) and (2.30). Details about the implementation and tuning of the three controllers are discussed in Section 2.3.2.

Finally, results are provided for two cases: servoing to a fixed feature configuration $\boldsymbol{s}^\star$ in Section 2.3.3 and tracking of a variable reference trajectory in Section 2.3.4.

2.3.1 Sensor Noise

Noise is introduced in the simulation to obtain a more realistic study-case. The first source of noise that is considered affects the current value of the features $\boldsymbol{s}$. In a real setup this might come from, *e.g.*, the image processing routine that extracts the points of interest, estimators that produce a continuous feedback when an image is not available or the sensor itself.

Feature noise is thus included by adding to each coordinate of $\boldsymbol{s}$ 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, a value of $\sigma = 2 \cdot 10^{-3}$ (corresponding to about 1 pixel) was used. A similar strategy has been used to simulate uncertainties in the unknown depths of the observed points, by adding to $\boldsymbol{z}$ a noise with $\sigma = 3 \cdot 10^{-3}$.

Regarding the measurement of joint positions and velocity, an assumption is made: only encoders are available to estimate the current configuration. These devices are simulated by discretizing the values of joints angles with a given resolution Δq. In particular, $\Delta q = 10\,\mu\text{rad}$ was used.

All noisy signals were then filtered using low-pass Butterworth 4^{th}-order filters and finally used in all the calculations performed by controllers.

2.3.2 Controllers Implementation and Parameters Tuning

First-Order (FO) Controller

As briefly mentioned above, the computation of the desired joint velocity (2.22) was modified so as to ensure smoother control signals in the beginning of task execution. In particular, in [MC07] it was shown that whenever 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 the present setup, the velocity reference $\dot{\boldsymbol{q}}^\star$ can be changed into

$$\dot{\boldsymbol{q}}^\star = \mathbf{J}_s^+ \left(\dot{\boldsymbol{s}}^\star - \lambda \boldsymbol{e}_s + e^{-\mu t} \boldsymbol{e}_s(0) \right) \tag{2.31}$$

where $\boldsymbol{e}_s(0)$ corresponds to the features error at $t = 0$, and under the assumption of null desired features velocity at the beginning, *i.e.*, $\dot{\boldsymbol{s}}^\star(0) = \mathbf{0}$. Due to this modification, the command at the very first iteration evaluates to 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

$$\ddot{\boldsymbol{e}}_s + (\mu + \lambda)\, \dot{\boldsymbol{e}}_s + \mu\lambda \boldsymbol{e}_s = \mathbf{0} \tag{2.32}$$

One interesting consequence of this choice is that it is possible to enforce the same desired behavior of the features evolution with both first and second-order controls. 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 (2.31), the effort command is computed as $\boldsymbol{\tau} = \gamma \mathbf{A} \left(\dot{\boldsymbol{q}}^\star - \dot{\boldsymbol{q}} \right) + \boldsymbol{b}$.

Regarding gains tuning, it was decided to impose a critically damped behavior for the transient response, which corresponds to the fastest convergence without oscillations:

$$\lambda = \mu = \omega_n \tag{2.33}$$

The parameter $\omega_n > 0$ defines the roots of the characteristic polynomial of the second-order system, and theoretically regulates the autonomous response of the error according

to $\boldsymbol{e_s}(t) = e^{-\omega_n t}\,(1+\omega_n t)\,\boldsymbol{e_s}(0)$. The selection of ω_n is thus performed so that, after a time T, the error corresponds to a given fraction of its original value, *i.e.*, $e^{-\omega_n T}\,(1+\omega_n T) = \alpha$, $\alpha \in (0,1)$. The equation is solved numerically for $\alpha = 0.05$ and $T = 2\,\mathrm{s}$, leading to the value $\omega_n \simeq 2.37\,\mathrm{s}^{-1}$.

Finally, the proportional velocity gain γ appearing in (2.23) has to be chosen. From a theoretical point of view, higher values will lead to better tracking of the desired velocity profile. However, this would also lead to increased control efforts, and therefore a trade-off has to be found. In the presented simulations, a value of $\gamma = 8\,\mathrm{s}^{-1}$ was used, corresponding to a step response that can reach the 80 % of the target value in 0.2 s.

First-Order with Feed-Forward (FOFF)

The implementation of this controller is very similar to the one of FO. In particular, the smooth velocity transition and the same set of gains are used here as well. The feed-forward term $\ddot{\boldsymbol{q}}_{ff}$ is computed as the numerical derivative of the reference command $\dot{\boldsymbol{q}}^\star$. It is well-known that this process highly amplifies noise, possibly resulting in unfeasible, if not just undesirable, control signals. In order to lessen the impact of noise and thus obtain a better behavior, (2.31) is thus filtered prior to differentiation. This considerably reduces the shakiness of the controller. The final efforts sent to the motors are evaluated using (2.23) and the IDM.

Second-Order (SO) Controller

This controller exploits the acceleration in the feature space. As explained, it computes joints accelerations using (2.30) to complete the visual task while simultaneously damping the velocity of the manipulator, finally exploiting the IDM to compute the actuation signal.

It must be noted that the control law requires an estimation of $\dot{\boldsymbol{e}}_{\boldsymbol{s}}$. For this purpose, it was decided to use directly the first-order interaction model, *i.e.*, $\widehat{\dot{\boldsymbol{e}}_{\boldsymbol{s}}} = \widehat{\mathbf{J}_{\boldsymbol{s}}}\dot{\boldsymbol{q}} - \dot{\boldsymbol{s}}^\star$.

The control gains k_p and k_d are selected to obtain a critically damped evolution as in the FO and First Order with Feed-Forward (FOFF) cases, with $k_p = \omega_n^2$, $k_d = 2\omega_n$, and $\omega_n = 2.37\,\mathrm{s}^{-1}$. The last parameter that need to be tuned is k_v, *i.e.*, the gain appearing in the secondary damping task (2.29). With the selected value of ω_n, the servoing task is expected to be completed in about 2 s. Therefore, k_v is tuned so that the convergence of velocity to zero is achieved after a slightly longer time. $k_v = 1\,\mathrm{s}^{-1}$ was used in practice,

theoretically ensuring that any non-null velocity is reduced to about the 5 % of its initial value in 3 s.

2.3.3 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 m, see Figure 2.2. As the desired value of the features does not change during the tests, in the controllers the quantities $\dot{\boldsymbol{s}}^\star$ and $\ddot{\boldsymbol{s}}^\star$ are set to zero.

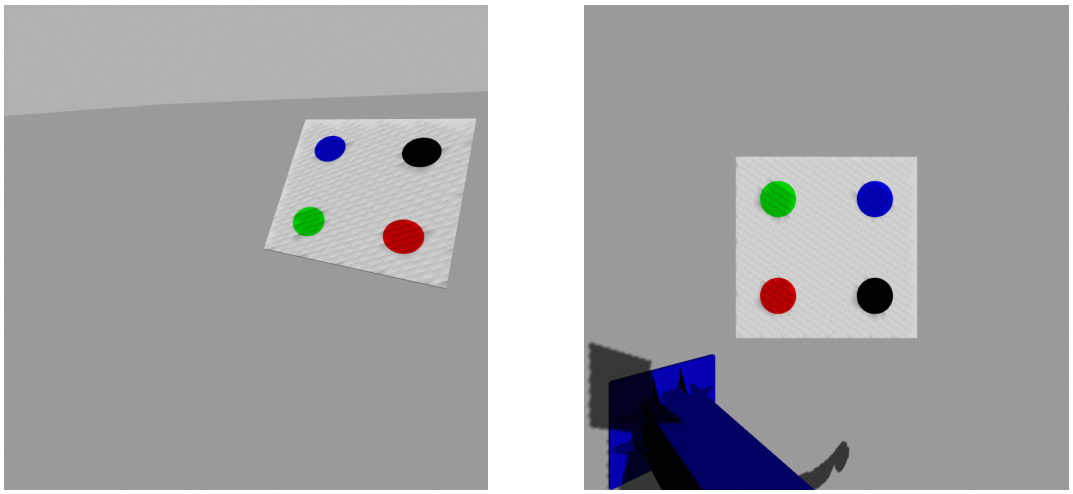


Figure 2.2 – Initial (left) and desired (right) configuration of the observed object. The points used for the task are the centers of the four circles.

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. Nonetheless, this affects all control laws and therefore no attempt to solve the problem was performed.

Results related to two cases are reported: an ideal test run in absence of noise (Figure 2.3) and a more realistic case with disturbances as previously discussed (Figure 2.4). 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 $\dot{\boldsymbol{q}}^\star$. 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.

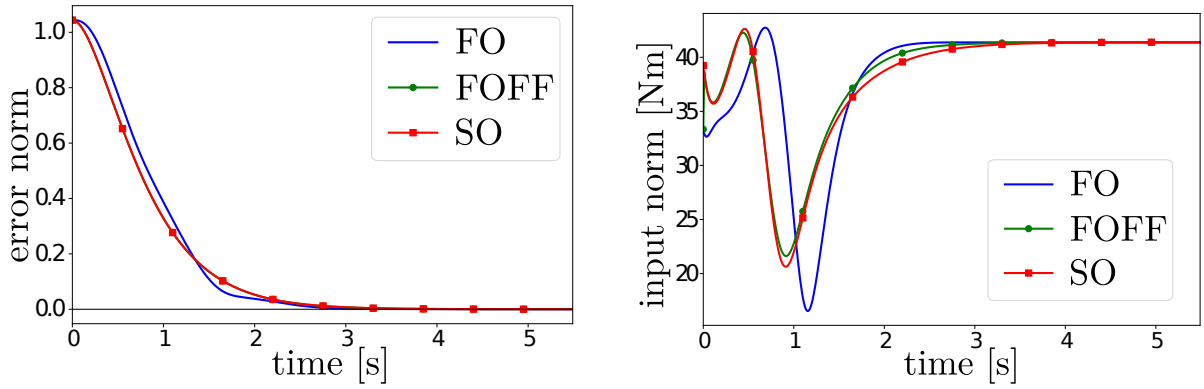


Figure 2.3 – Comparison of the visual error (left) and control input (right) using the different control strategies, in absence of noise. FOFF and SO are almost indistinguishable in terms of error evolution.

When noise is considered, more evident differences appear. FO's convergence worsens, with a little overshoot near to the equilibrium configuration, whereas both 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 N m even at equilibrium. Oscillations of FO and SO controllers are bounded instead to a reasonable amount, between 0.5 N m and 1 N m of amplitude in both cases.

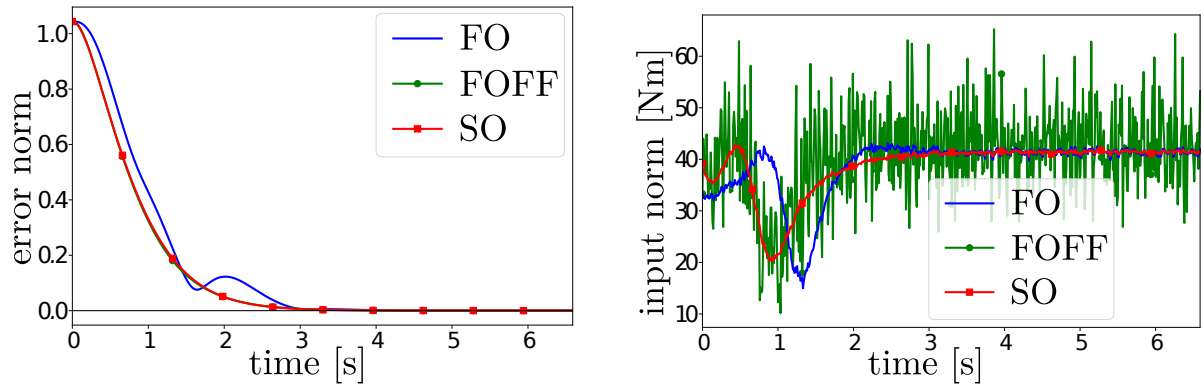


Figure 2.4 – Comparison of visual error (left) and control inputs (right) when noise is injected in simulations. FOFF and SO are still almost indistinguishable in terms of error evolution.

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 amplitudes are increased, 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 smaller compared to those introduced by FOFF.

2.3.4 Tracking of a Visual Trajectory

In this last section, the case of varying features references is considered. For each normalized point $\boldsymbol{s}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$, the motion was defined as a circular trajectory starting at the instant t_0 and ending after a time period T:

$$\begin{cases} x_i^\star(t) = x_i^\star(t_0) + R\left(\cos\left(\rho(t-t_0)\right) - 1\right) \\ y_i^\star(t) = y_i^\star(t_0) + R\sin\left(\rho(t-t_0)\right) \end{cases} \tag{2.34}$$

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 $\ddot{\rho}(0) = \ddot{\rho}(T) = 0$. The desired initial position of each point $\boldsymbol{s}_i^\star(t_0)$ is assumed 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 m since the sensor should be at 0.5 m from the four points, as detailed before. In addition, it is assumed that $t_0 \gg 0$, so that the homogeneous term in (2.31) does not affect the control, since smooth transition are already enforced by the choice of the time-varying reference.

Results are depicted in Figure 2.5, which shows that FO is once again the worst controller in terms of tracking error. FOFF still presents high oscillations in the input commands during the motion, but with good tracking performances. Finally, SO gives the best results, with tracking errors of the same magnitude of FOFF and reduced effort oscillations.

2.4 Conclusions

This chapter investigated the use of second-order models in order to establish a direct link with the dynamics of a manipulator. The resulting control scheme is a Computed Torque Control based on a Proportional and Derivative acceleration pseudo-command. Its performances were compared against two controllers that used instead a two-stages

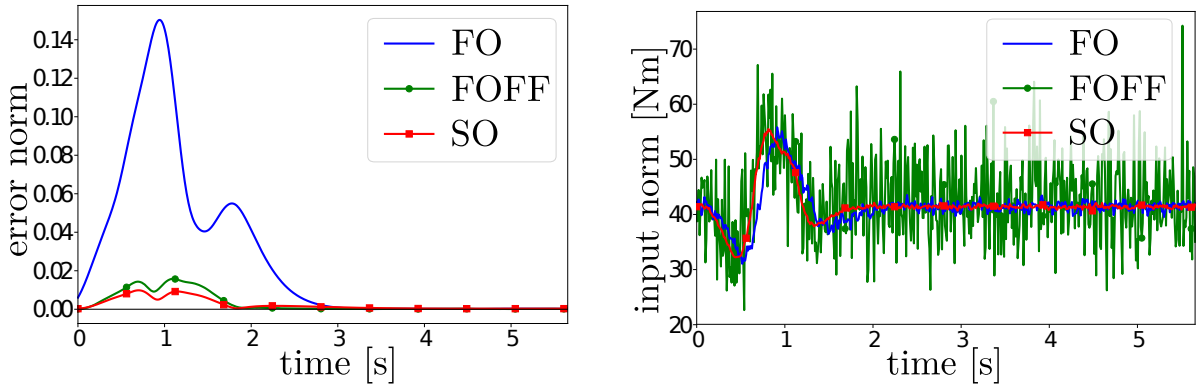


Figure 2.5 – Comparison of the visual error (left) and control input (right) during the execution of a trajectory. Time axes are relative to t_0.

control strategy composed of an outer servoing loop based only on velocity information. The inner loop attempts to track the produced joint velocity reference also by taking into account the dynamics of the robot. However, it was shown by means of simulations that performances are in this case less satisfactory. More specifically, it is necessary to either sacrifice tracking accuracy or to pilot the robot with very noisy control inputs. Instead, the use of second-order controllers provides the best results, allowing good tracking with more regular input efforts. These results were the subject of a conference article [FKM19] that was accepted for publication and presented at IROS 2019.

The interest of this class of models is not only related to feedback linearizing control. Indeed, as it will be shown in the next chapter, they can be used to formulate future predictions for a set of observed features. The quality of these predictions is generally higher than that obtained considering only velocity information. This allows to create new control schemes based on the Model Predictive Control paradigm thanks to which the robot can move faster and with shorter sensor trajectories.

Chapter 3

Visual Predictive Control

Even though feedback linearization techniques are frequently exploited in Visual Servo control, they present some limitations. One important issue is that they might lead to non-optimal behaviors with respect to the motion of the sensor in the Cartesian space. In fact, these strategies rely on the pseudo-inversion of the interaction matrix (or some estimation of it), which, depending on the features configuration, might lead to sub-optimal motions for the sensor. This comes directly from the structure of the interaction matrix of the chosen feature set, and therefore the options allowing to improve the motion are rather limited.

Moreover, it is not always straightforward to include and satisfy general constraints that are imposed on the system. Practical examples of such constraints are the need to keep an observed object inside a camera's field of view, or ensuring that the joint angles of a manipulator always lie within their physical limits.

A last relevant aspect – especially in the context of fast systems – is the one of the time required to converge to a given configuration. When the system is controlled via feedback strategies, the error is usually steered to zero with exponential decay. The rate at which the configuration of the system approaches the desired target is mainly controlled via parameters such as the gain of a proportional controller. In principle, the higher the gain, the faster the response. However, physical limitations of the actuators impose to consider control limits. To deal with them, it would be possible to simply saturate the command on a per-coordinate policy, but this might have undesired consequences such as loss of stability. Another way would be to scale all control input coordinates by a common factor, which ensures the system to follow the same path (thus retaining stability) at lower speed. However, the time needed to converge to the desired configuration would increase.

One of the reasons why feedback linearization techniques are subject to these limitations is that they aim at reducing the mismatch between the current features set and their desired value by following a locally optimal path. To some extent, they behave like a gradient-descent minimization algorithm, looking for the control that locally reduces the

error as quickly as possible. In this sense, one improvement could consist in considering how a control input affects future states reached by the system. This would allow to take better decisions in order to reach the desired state both in shorter time and with nicer motions, while simultaneously ensuring satisfaction of existing constraints. This naturally leads to the study of a powerful technique from the domain of optimal control, Model Predictive Control (MPC), which is the subject of study of this chapter, in particular with a focus on visual servoing applications.

This chapter is divided mainly into three parts: it begins with a quick introduction to the fundamentals of MPC, in particular focusing on existing mathematical formulations and the strategies that can be employed to solve the presented problems. A short review of existing works that used predictive strategies for visual servo control follows. Finally, the original contributions of this work are illustrated, which appeared in a peer-reviewed publication [FKM20]. They constitute a point of junction between Visual Predictive Control (VPC) and the second-order models presented in the previous chapter. Without entering into details yet, the core idea behind the proposed methods is that second-order models can be used to design more accurate predictors which ultimately lead to better robot motions. This claim is supported both by simulations and real experiments. In addition, the generality of these results are confirmed by the use of different types of features in the experiments: both normalized image points and their representation in polar coordinates.

3.1 Model Predictive Control

Model Predictive Control is an optimal control technique that aims at determining the best input signal to be applied to a system by taking into account the future evolution of its state. A continuous-time system is supposed to be described by the generic state-space form:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{3.1}$$

$\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$ being respectively the state vector and the control input of the system. A common formulation of the MPC problem is a finite-horizon, open-loop, optimization that takes into account the model (3.1) and a set of constraints. More precisely, denoting the current time instant as t_0, the mathematical formulation of the control problem can

be described as the following functional optimization:

$$\min_{\boldsymbol{u}(\cdot)} \int_{t_0}^{t_0+T_p} C\left(\boldsymbol{x}(t), \boldsymbol{u}(t), t\right) dt \tag{3.2}$$

subject to the constraints:

$$\boldsymbol{x}(t_0) = \boldsymbol{x}_0 \tag{3.3a}$$

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) \quad t \in [t_0, t_0 + T_p] \tag{3.3b}$$

$$\boldsymbol{x}(t) \in \mathcal{X} \subseteq \mathbb{R}^n \quad t \in [t_0, t_0 + T_p] \tag{3.3c}$$

$$\boldsymbol{u}(t) \in \mathcal{U} \subseteq \mathbb{R}^m \quad t \in [t_0, t_0 + T_p] \tag{3.3d}$$

In this formulation, the constant $T_p > 0$ is named *prediction horizon*, and it corresponds to the time interval up to which future states are evaluated and control inputs are to be designed. The function C – sometimes referred to as *stage cost* – models the desired behavior of the system. As an example, a simple, yet common in practice, choice is the use of a quadratic cost:

$$C\left(\boldsymbol{x}, \boldsymbol{u}, t\right) = \left(\boldsymbol{x} - \boldsymbol{x}^\star(t)\right)^T \mathbf{Q}(t) \left(\boldsymbol{x} - \boldsymbol{x}^\star(t)\right) + \left(\boldsymbol{u} - \boldsymbol{u}^\star(t)\right)^T \mathbf{R}(t) \left(\boldsymbol{u} - \boldsymbol{u}^\star(t)\right) \tag{3.4}$$

in which $\boldsymbol{x}^\star(t)$ and $\boldsymbol{u}^\star(t)$ represent the desired state and input signals over time, while $\mathbf{Q}(t)$ and $\mathbf{R}(t)$ are positive (semi)definite weighting matrices.

Regarding the constraints, the first one is simply used to fix in the optimization the initial state of the system, *i.e.*, $\boldsymbol{x}(t_0)$, while the second constraint reflects the fact that states should be predicted by means of the provided evolution model. Finally, (3.3c) and (3.3d) define the regions of admissible states and control values, and are often as simple as pairs of lower and upper bounds, *e.g.*, $\boldsymbol{x}^- \leq \boldsymbol{x}(t) \leq \boldsymbol{x}^+$.

It is worth remarking that the problem is meant to be solved online at each control step. In the case of $T_p = +\infty$ and with perfect knowledge of the model, it would be theoretically possible to solve it just once and then to apply the computed control signal $\boldsymbol{u}(t)$ to the system in an open-loop fashion. This is however generally infeasible for different reasons.

First and foremost, perfect knowledge of the model is impossible, especially considering the existence of disturbances and identification uncertainties, and the control input will need to be adjusted while the system evolves. A further reason is that the time to solve the optimization problem generally depends on the chosen horizon, and in the case of infinite horizons it might be unsuited for real-time applications. As a consequence, the problem is repeatedly solved online and only the first portion of the obtained control signal is sent to the actuators.

Regarding the way to find solutions to the problem, different strategies can be considered [FA02; Die+06]. Those belonging to the family of *indirect methods* try to solve the original problem "exactly", using, *e.g.*, the calculus of variations, dynamic programming, or so that an optimal feedback function $\boldsymbol{u} = \boldsymbol{\varphi}(\boldsymbol{x})$ is determined in closed-form. One famous example in this sense is the *Linear Quadratic Regulator* (LQR) [LVS12], which provides the optimal solution to the optimization problem in case the system is linear and not subject to state or control constraints. Such solution has the form of a linear state feedback, which can be obtained by solving a continuous-time Riccati differential equation – or an algebraic Riccati equation in case of an infinite-horizon problem. Note, however, that the problem is very challenging in the general nonlinear, constrained, case and usually too computationally expensive for an online implementation.

In order to achieve real-time feasibility, it is instead rather common to solve the problem using a different class of techniques, known as *direct methods*. In this context, the original continuous-time, constrained, functional optimization is approximated with a finite-dimensional minimization problem. First of all, the continuous-time dynamics (3.1) is approximated by a non-linear discrete-time system having the general form:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_d\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \tag{3.5}$$

In this sense, Euler's forward integration is arguably the most commonly used technique, according to which $\boldsymbol{x}_{k+1}$ is approximated by a first-order Taylor expansion:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \Delta t\, \dot{\boldsymbol{x}}_k = \boldsymbol{x}_k + \Delta t\, \boldsymbol{f}\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \tag{3.6}$$

Δt being the time sampling step.

For the purposes of MPC, once the continuous-time dynamics has been approximated with a discrete-time system, the objective becomes to compute a finite number of control samples, rather than the signal $\boldsymbol{u}(t)$ over the whole prediction horizon. More pre-

cisely, at the discrete time-step τ, a control sequence $\boldsymbol{u}_\tau$, $\boldsymbol{u}_{\tau+1}$, $\cdots$, $\boldsymbol{u}_{\tau+n_p-1}$ is to be evaluated, which will steer the system from the current state $\boldsymbol{x}_\tau$ to the new samples $\boldsymbol{x}_{\tau+1}$, $\boldsymbol{x}_{\tau+2}$, $\cdots$, $\boldsymbol{x}_{\tau+n_p}$. In this context, the original prediction horizon T_p is replaced by the positive integer n_p, the two constants being related by the simple formula $T_p = \Delta t\, n_p$. In order to shorten the notation in the following, τ will be omitted in the subscripts. It is implicitly assumed that discrete-time samples are relative to the "current" sample τ: as an example, $\boldsymbol{u}_3$ will stand for $\boldsymbol{u}_{\tau+3}$.

The new MPC optimization problem can finally be stated. The objective (3.2) is replaced by the following:

$$\min_{\underline{\boldsymbol{u}},\,\underline{\boldsymbol{x}}} \sum_{k=0}^{n_p-1} C_d\left(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k, k\right) \tag{3.7}$$

in which the decision variables $\underline{\boldsymbol{u}}$ and $\underline{\boldsymbol{x}}$ correspond to the stacked sequences of control samples and predicted states, *i.e.*,

$$\underline{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_0 \\ \cdots \\ \boldsymbol{u}_{n_p-1} \end{bmatrix} \qquad \underline{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}_1 \\ \cdots \\ \boldsymbol{x}_{n_p} \end{bmatrix} \tag{3.8}$$

It must be noted that the prediction vector $\underline{\boldsymbol{x}}$ does not contain the value $\boldsymbol{x}_0$, as it corresponds to the current state of the real system, measured or estimated, and thus cannot be freely changed during the optimization.

Like in the continuous-time MPC optimization problem, the objective is often in the form of a quadratic stage cost, the obvious differences being that the (continuous) time variable t is now replaced by k:

$$C_d\left(\boldsymbol{x}, \boldsymbol{u}, k\right) = \left(\boldsymbol{x} - \boldsymbol{x}^\star_{k+1}\right)^T \mathbf{Q}_{k+1} \left(\boldsymbol{x} - \boldsymbol{x}^\star_{k+1}\right) + \left(\boldsymbol{u} - \boldsymbol{u}^\star_k\right)^T \mathbf{R}_k \left(\boldsymbol{u} - \boldsymbol{u}^\star_k\right) \tag{3.9}$$

Finally, the constraints (3.3) can be kept almost unchanged:

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_d\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right) \quad k = 0, \cdots, n_p - 1 \tag{3.10a}$$

$$\boldsymbol{x}_k \in \mathcal{X} \quad k = 1, \cdots, n_p \tag{3.10b}$$

$$\boldsymbol{u}_k \in \mathcal{U} \quad k = 0, \cdots, n_p - 1 \tag{3.10c}$$

According to the above formulation, the problem has a total of $(n + m)\, n_p$ decision

variables. Depending on the model to be controlled and on the desired prediction horizon, the necessary time to evaluate a solution might still be significant. Nonetheless, there exists a widely exploited strategy so as to reduce the computational burden associated with the optimization, consisting in limiting the free control samples to be in the number of $n_c < n_p$. The usual way of dealing with this reduction of parameters is generally as simple as reducing the sequence $\underline{\boldsymbol{u}}$ to include only the first n_c samples, while the remaining $n_p - n_c$ inputs are kept constant and equal to $\boldsymbol{u}_{n_c-1}$ (see Figure 3.1). For this reason, the constant n_c is named *control horizon* in the literature.

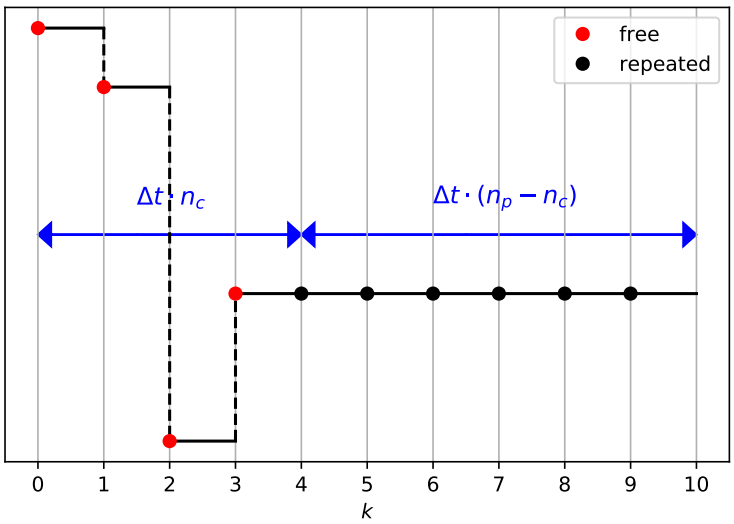


Figure 3.1 – Example of control sequence with $n_c = 4$ and $n_p = 10$. The last $n_p - n_c$ parameters are nothing but a repetition of $\boldsymbol{u}_{n_c-1}$.

Depending on the way the problem is solved, the number of decision variables can be further reduced. In particular, the so-called *single-shooting* methods allow the decision vector to be limited to the control variables $\underline{\boldsymbol{u}}$. On the other hand, *multiple-shooting* algorithms do solve the problem as stated by (3.7) and (3.10), but are able to better exploit parallelization and can thus still provide competitive solutions with respect to single-shooting methods [Die+06]. In both cases, once the problem has been solved the first control sample returned by the optimization, $\boldsymbol{u}_0$, is used as input to actuate the real system for the current control cycle. As in the continuous-time case, the new state reached by the system is then measured (or estimated) and used to formulate the new optimization problem to be solved at the next control iteration.

3.1.1 Single-shooting methods

The core idea behind single-shooting strategies for the solution of an MPC optimization is that the whole prediction vector $\underline{\boldsymbol{x}}$ can be removed from the decision variables thanks to the recursive structure of the discrete-time system (3.5). In fact, one can write:

$$\begin{aligned} \boldsymbol{x}_1 &= \boldsymbol{f}_d(\boldsymbol{x}_0, \boldsymbol{u}_0) \\ \boldsymbol{x}_2 &= \boldsymbol{f}_d(\boldsymbol{x}_1, \boldsymbol{u}_1) = \boldsymbol{f}_d(\boldsymbol{f}_d(\boldsymbol{x}_0, \boldsymbol{u}_0), \boldsymbol{u}_1) \\ \boldsymbol{x}_3 &= \boldsymbol{f}_d(\boldsymbol{x}_2, \boldsymbol{u}_2) = \boldsymbol{f}_d(\boldsymbol{f}_d(\boldsymbol{f}_d(\boldsymbol{x}_0, \boldsymbol{u}_0), \boldsymbol{u}_1), \boldsymbol{u}_2) \\ &\dots \end{aligned} \tag{3.11}$$

which shows that $\underline{\boldsymbol{x}}$ is ultimately a function of $\boldsymbol{x}_0$ and $\underline{\boldsymbol{u}}$ only. Since the former is simply a constant supplied to the optimization problem, only the latter is still to be considered within the decision variables, effectively reducing the dimensionality of the problem. Note that an additional advantage of single-shooting methods is that the nonlinear constraint (3.10a) is enforced by construction, and is not to be checked during the optimization.

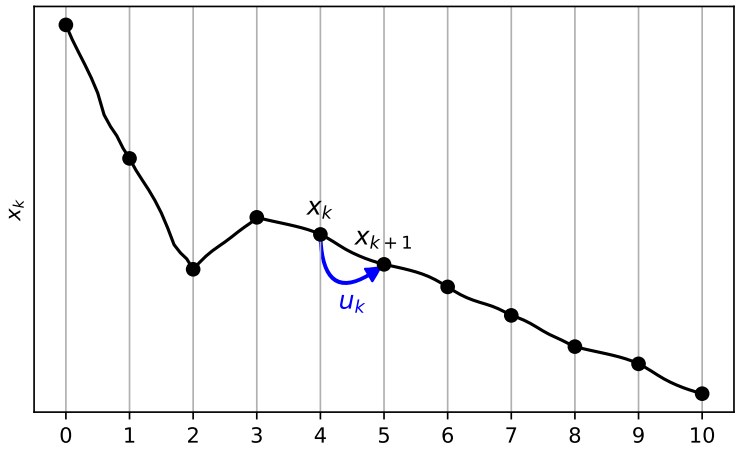


Figure 3.2 – Pictorial example of state prediction in single-shooting methods. Given a control sequence $\boldsymbol{u}_0, \cdots, \boldsymbol{u}_{n_c-1}$ and an initial sample $\boldsymbol{x}_0$, future samples are evaluated in succession using the model function $\boldsymbol{f}_d$. The result is a "continuous signal", whose samples constitute the prediction vector $\underline{\boldsymbol{x}}$.

In order to solve the optimization, various non-linear programming routines can be exploited, such as those detailed in Appendix B. Frequently, these algorithms will require the gradient of the objective and of the constraint functions to be supplied in order to

guide the search for the optimal solution. Obviously, these might be approximated by finite differences methods, but this generally leads to lower performances. In single-shooting algorithms, the main challenge is the computation of the jacobian of the prediction vector $\underline{\boldsymbol{x}}$ with respect to the control sequence $\underline{\boldsymbol{u}}$, since other quantities can be computed rather trivially from it. The first step for the computation of the jacobian $\frac{\partial \underline{\boldsymbol{x}}}{\partial \underline{\boldsymbol{u}}}$ is to focus on the partial derivatives of any predicted state with respect to the $j-th$ control sample, $\boldsymbol{u}_j$. Starting from the prediction model (3.5) and employing the chain rule it is possible to write:

$$\frac{\partial \boldsymbol{x}_{k+1}}{\partial \boldsymbol{u}_j} = \frac{\partial \boldsymbol{f}_d\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right)}{\partial \boldsymbol{x}_k}\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{u}_j} + \frac{\partial \boldsymbol{f}_d\left(\boldsymbol{x}_k, \boldsymbol{u}_k\right)}{\partial \boldsymbol{u}_k}\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_j} \tag{3.12}$$

It should be noted that, for $k, j < n_c$, $\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_j} = \delta_{j,k}\mathbf{I}$ ($\delta_{j,k}$ being Kronecker's Delta), which is a direct consequence of the fact that all elements in $\underline{\boldsymbol{u}}$ are independent variables. Furthermore, since all control samples from n_c and beyond are simply a copy of $\boldsymbol{u}_{n_c-1}$, it is also holds:

$$\frac{\partial \boldsymbol{u}_k}{\partial \boldsymbol{u}_j} = \delta_{j,n_c-1}\mathbf{I} \quad \forall k \geq n_c, \forall j < n_c \tag{3.13}$$

It is also important to notice that $\frac{\partial \boldsymbol{x}_0}{\partial \boldsymbol{u}_j} = \mathbf{O}$ since the initial state of the system is a given constant for the optimization. Finally, by looking at the relations reported in (3.11) it is possible to notice that, more generally, $\frac{\partial \boldsymbol{x}_k}{\partial \boldsymbol{u}_j} = \mathbf{O}$ if $k \geq j$ – which reflects the impossibility for a control input to affect a past state. By putting all these pieces together, the complete jacobian can be computed, and takes the lower block-triangular form shown in (3.14) (zeros have been omitted for easier reading). In the formula, sub-matrices $\mathbf{F}_u^{i,j}$ and $\mathbf{F}_x^{i,j}$ correspond respectively to $\frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{u}}$ and $\frac{\partial \boldsymbol{f}_d}{\partial \boldsymbol{x}}$, evaluated at the points $(\boldsymbol{x}_i, \boldsymbol{u}_j)$.

3.1.2 Multiple-shooting methods

As opposed to single-shooting strategies, multiple-shooting methods solve the optimization problem using both the control sequence and the prediction vector as decision variables. Figure 3.3 provides a graphical example of how the prediction of future samples is achieved during the optimization. First of all, one states is selected for each prediction step and, gathered altogether, these states correspond to a candidate prediction vector. In conjunction with a control sequence $\underline{\boldsymbol{u}}$, each state $\boldsymbol{x}_k$ can be mapped into $\boldsymbol{f}_d(\boldsymbol{x}_k, \boldsymbol{u}_k)$, *i.e.*, the following state that would be visited according to the model of the controlled

system. Obviously, if both $\underline{\boldsymbol{x}}$ and $\underline{\boldsymbol{u}}$ were chosen arbitrarily, the prediction would be discontinuous, with the successor of $\boldsymbol{x}_k$ not corresponding necessarily to $\boldsymbol{x}_{k+1}$. In order to ensure continuity over the prediction horizon, constraint (3.10a) is therefore taken into account explicitly, usually in the form:

$$\boldsymbol{d}_k \doteq \boldsymbol{x}_{k+1} - \boldsymbol{f}_d(\boldsymbol{x}_k, \boldsymbol{u}_k) = 0 \qquad \forall k = 0, \dots, n_p - 1 \tag{3.15}$$

Given that multiple-shooting methods require both a higher number of decision variables and a set of nonlinear equality constraints, it may seem that they necessarily perform worse than single-shooting solutions. However, it was shown that this is not the case and that multiple-shooting methods can offer even superior performances [Die+06]. One of the reasons for this is that they allow for a higher degree of parallelization. As an example, given a decision vector, the entries of constraints (3.15) can be evaluated simultaneously for different values of k, since they do not depend on each other. A similar example is related to the computation of the jacobians of the objective and of the equality constraints. In the first case, the evaluation is trivial since the objective is now a quadratic form in the decision variables directly. In the second case, since prediction samples are now independent variables, the jacobians of the constraint vector $\underline{\boldsymbol{d}}$ (being the stack of all individual equality constraints) take the very simple and sparse forms reported in (3.16). The spar-

$$\frac{\partial \underline{\boldsymbol{x}}}{\partial \underline{\boldsymbol{u}}} = \begin{bmatrix}
\underbrace{\mathbf{F}_u^{0,0}}_{} & & & & & \\
\underbrace{\mathbf{F}_x^{1,1}}_{} \square & \underbrace{\mathbf{F}_u^{1,1}}_{} & & & & \\
\underbrace{\mathbf{F}_x^{2,2}}_{} \square & \underbrace{\mathbf{F}_x^{2,2}}_{} \square & \underbrace{\mathbf{F}_u^{2,2}}_{} & & & \\
\mathbf{F}_x^{3,3} \square & \mathbf{F}_x^{3,3} \square & \mathbf{F}_x^{3,3} \square & \mathbf{F}_u^{3,3} & & \\
\vdots & \vdots & \vdots & \vdots & \ddots & \\
\underbrace{\mathbf{F}_x^{n_c-1,n_c-1}}_{} \square & \underbrace{\mathbf{F}_x^{n_c-1,n_c-1}}_{} \square & \underbrace{\mathbf{F}_x^{n_c-1,n_c-1}}_{} \square & \underbrace{\mathbf{F}_x^{n_c-1,n_c-1}}_{} \square & \underbrace{\mathbf{F}_x^{n_c-1,n_c-1}}_{} \square & \underbrace{\mathbf{F}_u^{n_c-1,n_c-1}}_{} \\
\mathbf{F}_x^{n_c,n_c-1} \square & \mathbf{F}_x^{n_c,n_c-1} \square & \mathbf{F}_x^{n_c,n_c-1} \square & \mathbf{F}_x^{n_c,n_c-1} \square & \mathbf{F}_x^{n_c,n_c-1} \square & \mathbf{F}_x^{n_c,n_c-1} \square \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix} \tag{3.14}$$

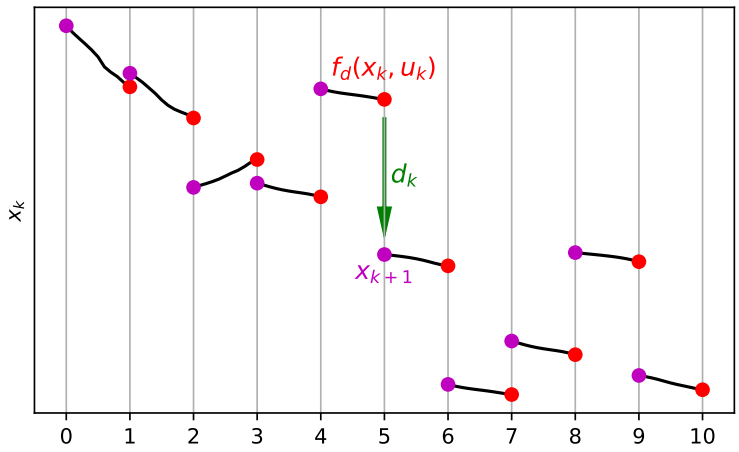


Figure 3.3 – Pictorial example of state prediction in multiple-shooting methods. Given a control sequence (not depicted here) and a set of initial samples (the points in magenta), future states (in red) are evaluated using the model function $\boldsymbol{f}_d$. The result is a "discontinuous signal", whose jumps, each of height $\boldsymbol{d}_k = \boldsymbol{x}_{k+1} - \boldsymbol{f}_d(\boldsymbol{x}_k, \boldsymbol{u}_k)$, should be nullified by the optimization algorithm.

sity is particularly important not only for the parallelization opportunity, but also because it can be exploited in order to efficiently compute solutions to the optimization problem. The result is potentially an algorithm that can converge to a solution in less time than single-shooting methods and possibly returning more stable solutions [Gif+18].

3.2 Predictive Control in Visual Servoing

Visual Predictive Control consists in the application of MPC to Visual Servoing tasks. In this section, a short summary of existing VPC schemes taken from the literature is presented, starting from early works that formulated the optimization in terms of a desired

$$\frac{\partial \underline{\boldsymbol{d}}}{\partial \underline{\boldsymbol{x}}} = \begin{bmatrix} \mathbf{I}_n & & & & & \\ -\mathbf{F}_x^{1,1} & \mathbf{I}_n & & & & \\ & \ddots & \ddots & & & \\ & & -\mathbf{F}_x^{n_c-1,n_c-1} & \mathbf{I}_n & & \\ & & & -\mathbf{F}_x^{n_c,n_c-1} & \mathbf{I}_n & \\ & & & & \ddots & \ddots \end{bmatrix} \quad \frac{\partial \underline{\boldsymbol{d}}}{\partial \underline{\boldsymbol{u}}} = \begin{bmatrix} -\mathbf{F}_u^{0,0} & & \\ & \ddots & \\ & & -\mathbf{F}_u^{n_c-1,n_c-1} \\ & & -\mathbf{F}_u^{n_c,n_c-1} \\ & & \vdots \end{bmatrix} \tag{3.16}$$

sensor pose in the Cartesian space. Afterwards, works that used sensor measurements (image point coordinates) directly in the prediction model are illustrated. The section then concludes with mentions on VPC approaches that introduced new ideas such as the use of reference trajectories or the use of image moments as features.

3.2.1 Generalized Predictive Control

The application of Generalized Predictive Control (GPC) [CMT87] to visual control proposed by Gangloff *et al.* [GDA98; GM00; GD03] is one of the first examples of Visual Predictive Control, in which a velocity-controlled robot is considered. In this approach, the objective is to control the value of a minimal parameterization of camera's pose with respect to an equilibrium configuration. This relative pose is denoted as $\boldsymbol{p}$, and its components are given as the 3D position and the roll-pitch-yaw angles of the camera frame with respect to the desired configuration. Its time variation is related to the kinematic screw of the camera by the relation $\mathbf{v} = \mathbf{J}_p \dot{\boldsymbol{p}}$, where $\mathbf{J}_p$ is defined as:

$$\mathbf{J}_p = \begin{bmatrix} {}^{c^\star}\mathbf{R}_c & \\ & \boldsymbol{\Phi}_{rpy} \end{bmatrix} \tag{3.17}$$

wherein ${}^{c^\star}\mathbf{R}_c$ is the rotation of the sensor frame with respect to the nominal desired pose, and $\boldsymbol{\Phi}_{rpy}$ transforms the rate of change of Euler's angles into angular velocities.

Direct measurements of image points can then be taken into account. Assuming the coordinates of the measured points to be collected in the feature vector $\boldsymbol{s}$, their variation is related to the pose vector by the interaction matrix and the relation above as:

$$\dot{\boldsymbol{s}} = \mathbf{L}_{\boldsymbol{s}} \mathbf{J}_p \dot{\boldsymbol{p}} \tag{3.18}$$

An assumption is now made to simplify the problem, that the current value of $\boldsymbol{p}$ is not considerably different from $\boldsymbol{p}^\star = \mathbf{0}$, *i.e.*, the current sensor pose is close to the desired configuration. Since the value of $\mathbf{J}_p$ tends to the identity matrix when $\boldsymbol{p} \to \mathbf{0}$, the authors propose therefore to approximate $\mathbf{v}$ with $\dot{\boldsymbol{p}}$. After this simplification, a first-order approximation defined around the desired equilibrium values $\boldsymbol{p}^\star = \mathbf{0}$ and $\boldsymbol{s} = \boldsymbol{s}^\star$ is introduced to obtain an initial estimation of $\boldsymbol{p}$ from the sensor measurements:

$$\boldsymbol{p} - \boldsymbol{p}^\star \simeq \mathbf{L}_{\boldsymbol{s}}^{+} \left(\boldsymbol{s} - \boldsymbol{s}^\star \right) \tag{3.19}$$

This estimation of $\boldsymbol{p}$ is used to initialize the state of the predictive model.

The next step in the modeling task is to consider the dynamics of the controlled manipulator. The robot is supposed to be controllable in velocity, but to take into account that a change in velocity is not immediate, an intermediate block is considered. Assuming that each joint velocity is regulated independently, the input-output relation between the reference command $\dot{q}_i^c$ and the actual velocity $\dot{q}_i$ is described by a set of transfer functions $F_i(s)$. These are in turn collected in the diagonal transfer matrix $\mathbf{F}(s)$. The velocity in the joint space is then linked to the operational space via the analytic jacobian $\mathbf{J}$, such that $\dot{\boldsymbol{p}} = \mathbf{J}\dot{\boldsymbol{q}}$. This is used to transform both actual and reference operational velocities, so that the transfer matrix from the command $\dot{\boldsymbol{p}}^c$ to the resulting rate of change $\dot{\boldsymbol{p}}$ is given by $\mathbf{JF}(s)\mathbf{J}^{-1}$.

Few additional blocks are also considered: a Zero-Order Holder (ZOH) models the fact that a reference joint velocity command $\dot{\boldsymbol{q}}^c$ is applied for an entire control period, while an integrator is used to produce the signal $\boldsymbol{p}$ from its derivative $\dot{\boldsymbol{p}}$. The overall sub-system obtained in this way is commonly referred to as Virtual Cartesian Motion Device (VCMD) since its final objective is to track a given operational velocity set-point. The model of the manipulator is completed by introducing two unit-delays, each one representing respectively the control computation and the image-processing step. The overall linearized model is depicted in Figure 3.4.

An equivalent discrete-time model of the VCMD is finally obtained using a step-invariant transformation. Considering also the two unit delays, the overall discrete-time transfer matrix from the sampled input $\dot{\boldsymbol{p}}_k^c$ to the predicted output $\hat{\boldsymbol{p}}_k$ is given by:

$$\mathbf{G}(z) = z^{-2}\left(1 - z^{-1}\right)\mathcal{Z}\left[\frac{\mathbf{JF}(s)\mathbf{J}^{-1}}{s^2}\right] \tag{3.20}$$

where $\mathcal{Z}[\cdot]$ corresponds to the transformation of a continuous-time transfer function to the discrete domain. At each control iteration, $\mathbf{J}$ is assumed to be constant, and, considering the diagonal structure of $\mathbf{F}(s)$, the overall function is therefore rewritten as:

$$\mathbf{G}(z) = z^{-2}\mathbf{JH}(z)\mathbf{J}^{-1} = z^{-2}\mathbf{J}\mathrm{diag}\left\{\frac{a_i(z^{-1})}{b_i(z^{-1})}\right\}\mathbf{J}^{-1} \tag{3.21}$$

with $a_i(z)$ and $b_i(z)$ representing a set of polynomials.

The model is then used for the prediction of n_p samples, given n_c control inputs. The

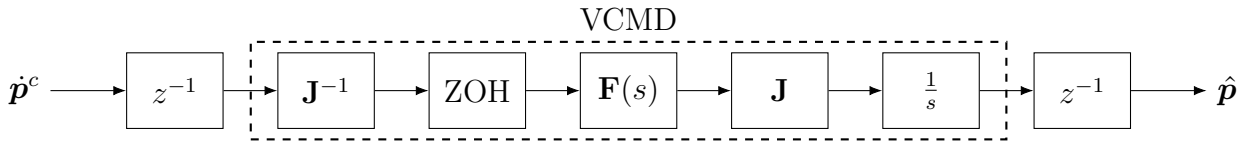


Figure 3.4 – Input-output linearized model of the robot and of the visual system [GD03]. The central block is the Virtual Cartesian Motion Device.

objective is formulated using a quadratic stage cost and takes the form:

$$\sum_{k=1}^{n_p} \|\hat{\boldsymbol{p}}_k - \boldsymbol{p}_k^\star\|^2 + w \sum_{k=0}^{n_c-1} \left\|\dot{\boldsymbol{p}}_k^c - \dot{\boldsymbol{p}}_{k-1}^c\right\|^2 \tag{3.22}$$

with $w > 0$ being a weighting factor. The objective thus aims at minimizing the predicted pose error as well as reducing variations in the control law.

Although the proposed method does not deal with constraints such as visibility or retraction, it is still interesting as it takes into account the effect of low-level controllers and of the joint dynamics. The validity of the prediction model is only local, as the transfer matrix $\mathbf{F}(s)$ should be evaluated around a given configuration in order to get consistent approximations of the joint dynamics, and has therefore to be identified for multiple local configurations.

3.2.2 Features in the optimization objective

Even though the simple linear structure of the previous approach makes the problem easy and efficiently solvable, more complex approaches based on nonlinear predictive controls should be considered for enhanced performances. A work that explored this direction was [Sau+06], in which the nonlinear dynamic model of a manipulator with 6 degrees of freedom was considered to control the pose of its end-effector, assuming a eye-to-hand configuration and four image points as visual features.

In the design of the prediction model, the VCMD is not used. Instead, a low-level proportional regulator is considered directly at the joint level, which produces an auxiliary acceleration signal that is then transformed into motor efforts by considering the IDM of the manipulator. The joint position and velocity of the robot are updated directly from the acceleration signal, assuming perfect identification of the dynamic parameters.

An interesting improvement proposed in this work is also the direct use of the visual features to define the objective. To take into account the evolution of image points in the image, two assumptions are made:

1. The 3D properties of the object are well-known, *i.e.*, the position ${}^{o}\boldsymbol{P}_i$ of each i-th point in the object frame is available;
2. The transformation ${}^{w}\mathbf{T}_c$ from the camera frame to the robot base is available.

From these assumptions, the 3D position of each feature point is built at each prediction step from the geometric model of the robot. Image coordinates are then computed from the classical pinhole camera projection. The overall prediction model, from the joint velocity command to the predicted states, is reported in Figure 3.5.

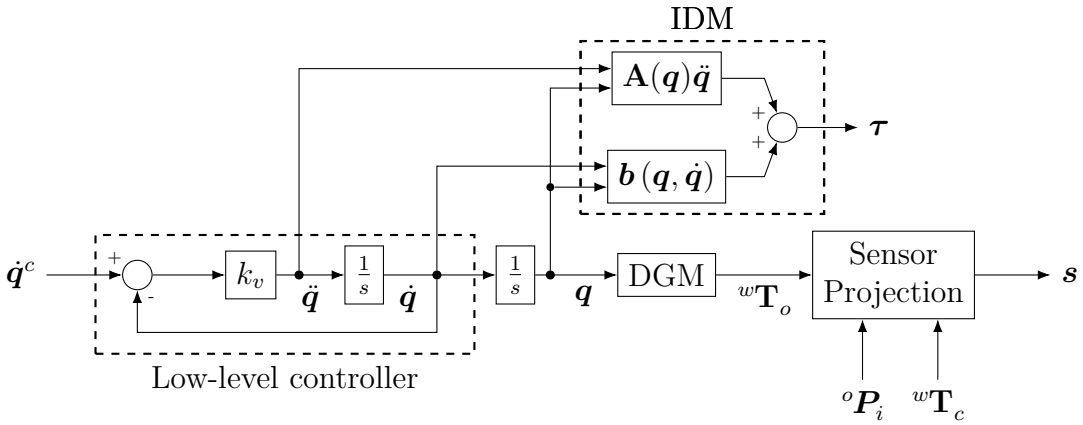


Figure 3.5 – Open-loop prediction model considered in [Sau+06]. The input is a joint velocity command which is internally regulated via a low-level proportional controller. Assuming the dynamic parameters to be perfectly identified, the current velocity and position are obtained simply by integrating the acceleration, and a prediction of the effort is obtained using the IDM. The geometric model of the manipulator in conjunction with the camera perspective projection allows to obtain image features coordinates.

The nonlinear MPC is then built on top of the presented model. In order to accomplish the visual task, a quadratic stage cost is considered directly on the feature error. In addition, minimization of the control inputs is taken into account as well. Constraints are also introduced in order to ensure features visibility and torque limits, as well as to keep joints within their bounds.

This control strategy was compared in simulation against a PI controller, using a sampling period of 5 ms. The control and prediction horizons were set to 1 and 10 respectively, and the optimization was solved using a Sequential Quadratic Programming (SQP) algorithm (refer to Appendix B.3 for additional details). Simulations showed the effectiveness of the proposed approach, in particular concerning constraint handling. The robustness of the controller was also analyzed against relatively large identification errors,

both in dynamic parameters and camera calibration. Nonetheless, the computation time was about 400 ms in average, unacceptable for a real-time implementation.

The work by Allibert *et al.* [ACT08] considered a similar setup. The studied system is composed by a Scara-like robot equipped with an omni-directional visual sensor which is mounted on the end-effector. The dynamics of the robot is not considered in this case, and the joint state is simply obtained from the control input, *i.e.*, a joint velocity command, using Euler's forward integration:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + \Delta t\, \dot{\boldsymbol{q}}_k \tag{3.23}$$

The measurement of features is instead modeled by the following equation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \end{bmatrix} \begin{bmatrix} \beta X \\ \beta Y \\ 1 \end{bmatrix} \qquad \beta = \frac{\xi + m}{Z + \xi\sqrt{X^2 + Y^2 + Z^2}} \tag{3.24}$$

with X, Y and Z being the 3D coordinates of an observed point and the other constants representing the intrinsic parameters of the sensor. As in [Sau+06], the position of the image points are evaluated using the geometric model of the arm and knowledge of the object structure and location.

This work also proposed to use the so-called Internal Model Control Structure to adjust the desired value of the image features. When predicting the future state of observed points, the approximations introduced in the model will induce an error in the new value. Formally, at the current iteration τ the value of the feature vector $\boldsymbol{s}(\tau)$ and the one predicted by the model, $\boldsymbol{s}_m(\tau)$, will differ by a given quantity $\boldsymbol{\varepsilon}(\tau)$:

$$\boldsymbol{\varepsilon}(\tau) = \boldsymbol{s}(\tau) - \boldsymbol{s}_m(\tau) \tag{3.25}$$

If we assume that this error is constant for the given model, then the task of tracking the desired sensor configuration $\boldsymbol{s}^\star$ with the real output can be assumed to be equivalent as tracking the modified reference $\boldsymbol{s}_m^\star$ with $\boldsymbol{s}_m$. Therefore, it is suggested to use, as desired state in the objective, the values:

$$\boldsymbol{s}_m^\star = \boldsymbol{s}^\star - \boldsymbol{\varepsilon}(\tau) \tag{3.26}$$

The simulations provided in this work clearly show the advantage of using nonlinear predictive controllers to regulate the sensor output while taking into account constraints.

In particular, when considering large displacements classical feedback control might cause the features to leave the field of view, leading to failure. MPC strategies can instead optimize the control input in a way that ensures visibility.

3.2.3 Models based on the interaction matrix

All prediction models presented so far require full knowledge of the observed object, since the calculation of visual features is based on this information and on the use of the geometric model of the controlled robot. However, to update the state of image features one could directly consider the differential relation between their derivatives and the sensor twist by using the interaction matrix. This idea seems to appear for the first time in two different works [LB09; AC09].

In the former, the update model considers the robot as a VCMD whose continuous-time transfer matrix corresponds to a proportional regulator $\mathbf{G}(s) = \text{diag}\left\{\frac{k_v}{s+k_v}\right\}$ – analogous to the low level controller of Figure 3.5, but directly applied to sensor's velocity. After transforming it in the discrete domain, the update equation for the feature vector is given as:

$$\boldsymbol{s}_{k+1} = \boldsymbol{s}_k + \Delta t\, \mathbf{L}_{\boldsymbol{s}} \mathbf{G}(z^{-1}) \mathbf{v}_k^c \tag{3.27}$$

in which $\mathbf{v}_k^c$ corresponds to the desired velocity of the camera at step k, which is assumed to be the control input of the overall system. The interaction matrix is computed at each iteration, using the current feature prediction $\boldsymbol{s}_k$. To deal with depth changes, the authors propose to use a scheme that gives the depth as a function of the desired point coordinates and the current image projection [LB08].

After validating the proposed predictor, simulations were presented. They confirm as well that MPC can outperform feedback strategies by finding better motions for the system. In particular, a small pure rotation about the optical axis of the camera is considered. Compared to a PI control, which tries to move the features along straight lines towards their desired location, the MPC is able instead to optimize the motion so that a simpler circular one is performed.

The approach presented in [AC09] is similar to that of [LB09], in the sense that an update scheme analogous to (3.27) is exploited. The main differences in this case are that the VCMD is not considered and that the depth at the desired configuration is used to approximate the interaction matrix. Extensive simulations are included to analyze

different test-cases, all considering a unitary control horizon.

A first set of simulations focuses on pure rotations around the optical axis of the camera, with different parameters tuning. First of all, a unitary prediction horizon and a 90° rotation are considered. In this case, the path followed by the features is close to the one given by classical control strategies. However, the decay of the feature error is not exponential, due to the fact that some camera velocities will be kept saturated to achieve lower values of the objective. This result is rather interesting in the context of this thesis, as it shows that the convergence rate of the servoing task can be improved even with a unitary prediction horizon. The value of n_p is successively increased to either 10 or 20, with a stage cost matrix $\mathbf{Q}_k = \mathbf{I}$. Thanks to the possibility of foreseeing future samples, the features follow a circular path in the image space, with less retraction in the z-direction. Finally, an increasing weighting policy is tested in the objective, by setting $\mathbf{Q}_k = 2^k\mathbf{I}$. The 2D behavior of the features is similar to the previous case, but in addition the 3D motion of the sensor appears nicer, with even less retraction than in the previous case. This is justified by the fact that the optimization has to favor the minimization of the error in future samples, thus discouraging "greedy behaviors" and leading to motions that, despite being sub-optimal in the short-term, are globally more satisfactory.

A second set of simulations considered larger displacements, requiring both rotational and translational motions of the sensor. Similarly to the case of a pure rotation, it is shown that the use of a larger prediction horizon and of an increasing weighting policy can lead to superior performances, with smooth motions and better decoupling in the sensor's degrees of freedom. The same scenario was also simulated in presence of calibration errors and image processing disturbances to show the robustness of the proposed scheme.

The authors later extended the approach by considering new models in [ACC10], classified in two categories: *Local Models* (LM) and *Global Models* (GM). First of all, the previously proposed predictor, based on $\mathbf{L}|_{z=z^\star}$, was renamed LM_p, and a new model was introduced by assuming that depths are estimated at each iteration and used in place of $\boldsymbol{z}^\star$. This second local approach has been named LM_c.

Global methods assume instead to know the 3D coordinates (X^i, Y^i, Z^i) of the observed points in the camera frame. In this case, point locations are assumed to evolve

according to:

$$\begin{bmatrix} X^i \\ Y^i \\ Z^i \end{bmatrix}_{k+1} = \mathbf{R}\left(\mathbf{v}_k\right) \begin{bmatrix} X^i \\ Y^i \\ Z^i \end{bmatrix}_k + \boldsymbol{t}\left(\mathbf{v}_k\right) \tag{3.28}$$

where $\mathbf{R} \in \mathrm{SO}(3)$ and $\boldsymbol{t} \in \mathbb{R}^3$ define a homogeneous transformation matrix. This matrix corresponds to the local change of pose of the sensor between the steps k and $k+1$ due to the action of the control input $\mathbf{v}_k$. From the 3D information, the output is obtained by normalizing the coordinates of the points:

$$\boldsymbol{s}_k^i = \begin{bmatrix} u^i \\ v^i \end{bmatrix}_k = \begin{bmatrix} X^i/Z^i \\ Y^i/Z^i \end{bmatrix}_k \tag{3.29}$$

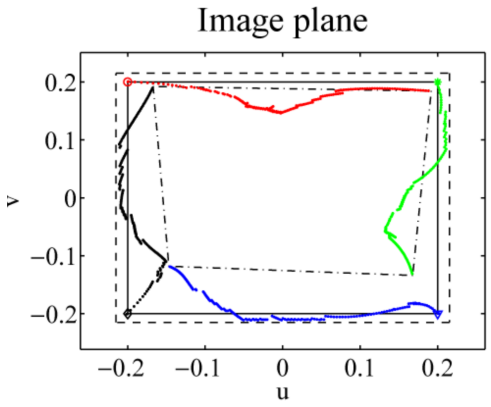
Similar to the case of Local Models, two schemes were considered depending on the availability of points depth at the current control iteration. GM_p assumes that the 3D state is initialized from the current image measurements $\boldsymbol{s}$ and the depth at equilibrium $Z^{\star i}$:

$$\begin{bmatrix} X^i \\ Y^i \\ Z^i \end{bmatrix}_0 = \begin{bmatrix} x^i Z^{\star i} \\ y^i Z^{\star i} \\ Z^{\star i} \end{bmatrix}_\tau \tag{3.30}$$

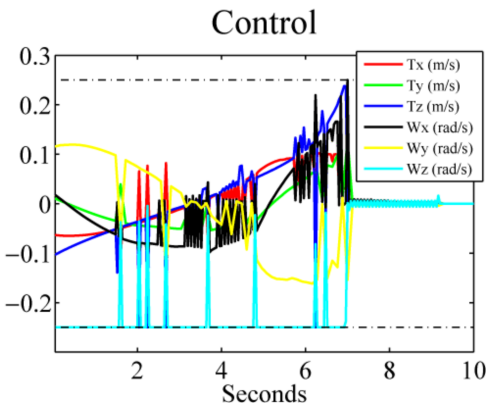
whereas GM_c substitutes $Z^{\star i}$ with the current depth estimation $\hat{Z}_\tau^i$.

Simulation results reported in [ACC10] show the benefits of predictive controllers, especially when the depth is updated during the estimation. A remarkable observation included in the analysis concerns the so-called *Optimal IBVS* strategy. This scheme corresponds to a VPC with unitary control and prediction step. As already reported before, this choice leads to a behavior that is generally similar to what a feedback control would do, but with enhanced convergence rate. Nonetheless, a major advantage consists in the ability of Optimal IBVS to deal with constraints such as visibility. As shown in Figure 3.6(a), the controller is able to find a solution that accomplishes the visual task without letting features leave the field of view. There is however a drawback: since the prediction horizon is limited to a single step in the future, the control law is not smooth, with a sort of "bang-bang" action (see Figure 3.6(b)).

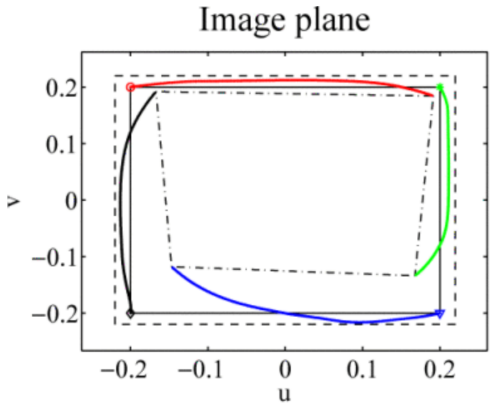
The problem with Optimal IBVS is easily solved by extending the prediction horizon, giving smoother and more effective results. In addition, the use of variable depth in the prediction process allows to explicitly enforce better 3D behavior of camera's mo-



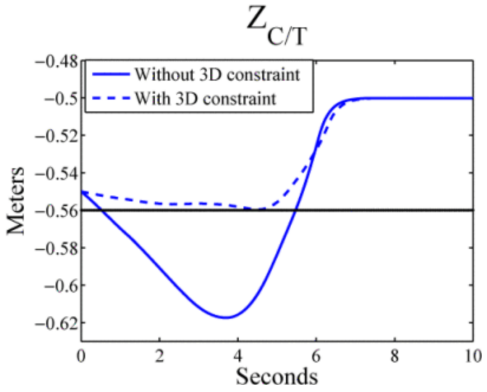
(a) Features evolution using Otimal IBVS.



(b) Control inputs resulting from Optimal IBVS.



(c) Features evolution using GM_c with $n_p = 10$ and $n_c = 1$.



(d) Distance of the camera from the object in two VPC simulations.

Figure 3.6 – Results from Visual Servoing simulations using either Optimal IBVS or GM_c VPC [ACC10]. Visibility constraints are enforced during the motion using both strategies, which is generally not the case in classical feedback controllers. However, when using Optimal IBVS, input control signals are highly discontinuous (b), resulting in an irregular features evolution (a). On the other hand, with a larger prediction horizon ($n_p = 10$) the resulting motion is much smoother (c). Finally, in (d), a comparison of the distance of the camera from the object along the vertical axis with and without workspace bounds to explicitly limit retraction.

tion. Considering the case of large rotational displacements, previously existing VPCs were exploiting the simple idea that when a short control horizon is selected, the optimal input sequence generally corresponds to an almost pure rotation with reduced translation components, at least in comparison with classical approaches. However, once depth is considered the retraction problem can be handled explicitly by imposing workspace bounds. As an example Figure 3.6(d) includes results from the study in which both 2D and 3D constraints are included. By constraining the vertical workspace of the sensor as $Z_c \in [-0.6\,\mathrm{m}, -0.5\,\mathrm{m}]$, retraction is directly prevented.

3.2.4 Visual Predictive Control with reference trajectory

As stated in the beginning of this section, the stage cost for the optimization does not need to consider a fixed setpoint. Instead, the predicted future states $\boldsymbol{x}_k$ might be required to track a given reference trajectory $\boldsymbol{x}_k^\star$ (*cfr.* (3.9)). If a way to define such a (possibly feasible) trajectory is available, this can be advantageous in order to help the system evolve in a controlled way.

An example in this regard can be found in the work from Lazar *et al.* [LBC11] which adds to the VPC presented in [LB09] a trajectory generator based on homographies as described in [AF05]. It is well known that when a set of coplanar (and non-collinear) points is observed from two different perspectives a and b, their coordinates obey the homography relation [HZ03]:

$$ {}^b\boldsymbol{P} = \mathbf{H}\,{}^a\boldsymbol{P} = \left({}^b\mathbf{R}_a + \frac{{}^b\boldsymbol{t}_a \boldsymbol{n}_a^T}{d_a} \right) {}^a\boldsymbol{P} \tag{3.31} $$

where ${}^b\mathbf{R}_a$ and ${}^b\boldsymbol{t}_a$ represent the homogeneous transformation from a to b, while the couple $\boldsymbol{n}_a, d_a$ defines the plane equation in a, *i.e.*, such that $\boldsymbol{n}_a^T {}^a\boldsymbol{P} = d_a$. By taking into account camera intrinsic parameters by means of a matrix $\mathbf{K}$, the relation between points given in image coordinates becomes:

$$ \begin{bmatrix} {}^b\boldsymbol{s} \\ 1 \end{bmatrix} = \alpha \mathbf{K}\mathbf{H}\mathbf{K}^{-1} \begin{bmatrix} {}^a\boldsymbol{s} \\ 1 \end{bmatrix} = \mathbf{G} \begin{bmatrix} {}^a\boldsymbol{s} \\ 1 \end{bmatrix} \tag{3.32} $$

with α a scaling parameter (theoretically equal to the depth ratio of the points in the two perspectives) and $\mathbf{G}$ named *collineation matrix*. By considering the feature set at the current iteration and at equilibrium, a link is established by (3.32) involving the current pose of the camera with respect to its desired one. In order to produce a suitable

reference trajectory in the image space, it is possible to consider a continuous sequence of homogeneous transformations in the form:

$$\mathbf{H}(\rho) = \mathbf{R}(\rho) + \frac{\boldsymbol{t}(\rho)\,\boldsymbol{n}_i^T}{d_i} \tag{3.33}$$

with d_i and $\boldsymbol{n}_i$ defining the equation of the observed plane at the initial view, and ρ a monotonic function of time ranging from 0 to 1. If we impose $\mathbf{R}(0) = \mathbf{I}$, $\boldsymbol{t}(0) = \mathbf{0}$, $\mathbf{R}(1) = {}^{\star}\mathbf{R}_i$ and $\boldsymbol{t}(1) = {}^{\star}\boldsymbol{t}_i$, the resulting collineation matrix is such that $\mathbf{G}(\rho)\boldsymbol{s}_i$ will change from the current values of $\boldsymbol{s}$ to $\boldsymbol{s}^{\star}$.

In order to obtain a smooth motion from the current time τ and the prediction horizon, the interpolation variable ρ can be chosen, *e.g.*, as a quintic polynomial in the normalized time $\frac{t-\tau}{T_p}$, with initial conditions $\rho(\tau) = 0$, $\rho(\tau + T_p) = 1$ and $\dot{\rho}(0) = \dot{\rho}(\tau + T_p) = \ddot{\rho}(0) = \ddot{\rho}(\tau + T_p) = 0$. An alternative design of the interpolation variable might impose a non-zero initial condition on $\dot{\rho}$ [CLB12]. This results in reference trajectories that approach the equilibrium configuration in shorter time – obviously, the initial condition should be selected not to cause overshooting.

Regarding the interpolation of $\mathbf{R}$ and $\boldsymbol{t}$, three cases can be considered:

1. Pure translation: the rotation matrix $\mathbf{R}$ is kept as the identity, and linear interpolation is performed as $\boldsymbol{t}(\rho) = \rho\,{}^{\star}\boldsymbol{t}_i$.

2. Pure rotation: the translation is kept constant to zero, while spherical linear interpolation (SLERP) can be used for the rotation matrix (either in terms of Rodrigues' formula or using quaternions).

3. Rotation and translation: the desired orientation is still obtained via SLERP, while the translational path is computed using a helicoidal shape. The linear translation axis of the helix is chosen aligned to the rotation vector $\theta\boldsymbol{u}$ corresponding to ${}^{\star}\mathbf{R}_i$.

The overall VPC strategy is then almost identical to previously presented approaches, with $\boldsymbol{s}^{\star}$ being replaced by the generated time-dependent reference samples $\boldsymbol{s}_k^{\star}$.

As a concluding remark on the reported procedure, it must be noted that in practice it is possible to recover the relative translation ${}^{\star}\boldsymbol{t}_i$ only up to a scaling factor. Therefore, in the proposed approach the translational motion should actually be replaced by a scaled version. Nonetheless, this does not alter the validity of (3.32), in the sense that the obtained feature projection will be identical no matter the value of the unknown scale factor.

3.2.5 Visual Predictive Control with image moments

Image moments have been considered as well for the design of Visual Predictive Controllers [CBL11; CLB12]. As already discussed, (planar) image moments can be evaluated from a set of image points using either (1.17) or (1.18) depending on whether continuous or discrete moments are considered, with the related interaction matrices given by (A.38). Following the same notation adopted in related works, the transformation giving image moments from a set of points will be denoted by the symbol $\mathbb{M}$.

Two main approaches can be considered in order to establish a prediction model [CLB12]. In the first case, a set of point features $\boldsymbol{s}$ is considered and their future value is estimated using the interaction matrix as in [LB09] (see (3.27)). In this matrix, however, the inverse depth is replaced by the corresponding value computed from the plane parameters A, B and C, *i.e.*, for the i-th image point $\frac{1}{Z_i} = Ax_i + By_i + C$. This strategy gives a sequence of predicted states $\boldsymbol{s}_k$ which can then be transformed into corresponding image moments $\boldsymbol{m}$, *i.e.*, $\boldsymbol{m}_k = \mathbb{M}(\boldsymbol{s}_k)$.

The second strategy operates directly on moments sets. Starting from the current value of point features, $\boldsymbol{s}_0$, corresponding moments $\boldsymbol{m}_0$ are obtained using $\mathbb{M}$. For future predictions, the interaction matrix of image moments is assumed constant over the prediction horizon, and is used to obtain predictions for the image moments as:

$$\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \Delta t \, \widehat{\mathbf{L}}_{\boldsymbol{m}} \mathbf{v}_k \tag{3.34}$$

In practice, the update model considers $\widehat{\mathbf{L}}_{\boldsymbol{m}}$ to be the half-sum of the interaction matrix computed at the current iteration and at equilibrium, *i.e.*,

$$\widehat{\mathbf{L}}_{\boldsymbol{m}} = \frac{1}{2} \left(\mathbf{L}_{\boldsymbol{m}}(\boldsymbol{m}_0) + \mathbf{L}_{\boldsymbol{m}}(\boldsymbol{m}^\star) \right) \tag{3.35}$$

Two VPC schemes were designed from the given prediction models. The objective can be designed to either reach a fixed set-point as in the classical setup or to track a reference trajectory using the generator illustrated in the previous section. In this case, to define the target motion in terms of image moments, it is possible to exploit the homography-based approach to firstly define a reference trajectory for the image points and then to transform them into reference moments via $\mathbb{M}$. Regarding the weighting matrices used in the stage cost, it is proposed to use $\mathbf{Q}_k = e^k \mathbf{I}$ when a fixed set-point is considered as target, thus increasing the weight to achieve a faster response towards the goal. Conversely, when considering a trajectory over the prediction horizon, $\mathbf{Q}_k = e^{-k} \mathbf{I}$ is used, in order to enforce

better tracking of the time-varying reference.

3.3 Proposed Contributions to Visual Predictive Control

In previous sections, an introduction to Model Predictive Control has been presented, with a focus on Visual Servoing applications. The reason to consider such kind of control strategy for the purposes of this thesis is that, as discussed, when the initial configuration is characterized by large displacements with respect to the desired one, the optimization generally leads to shorter and more effective motions of the controlled system. In addition, the feature error usually decreases with a faster response than the one given by the exponential decay even when the current configuration approaches the target one.

There are different directions that can be considered to extend existing approaches, one of them consisting in considering new prediction models. In particular, the use of second-order approximations of the discrete-time feature prediction seemed a viable new method to be investigated. Another factor to be taken into account is related to redundancy. While the outlined studies focused on manipulators featuring 6 degrees of freedom, our experimental platform is instead redundant with respect to the visual task. As such, a proper modified objective should be considered to properly exploit the additional degree of freedom. Finally, it must be noted that the computational cost of the nonlinear optimization can be relatively high. Given the objective of designing a fast control scheme, some attention should be dedicated to an efficient implementation allowing to obtain control inputs in real-time with sufficiently fast motions.

3.3.1 Predictors Based on Second-Order Models

The first proposed contribution to VPC in this thesis consists in taking into account higher-order models in the predictive update equation. In fact, by taking into account also the acceleration of the visual features, the resulting prediction should be more accurate and therefore lead to better control sequences.

Similarly to what was proposed in [LB09; AC09], we propose to use a local model in which the feature vector of the system is updated only using local information on the features derivatives. However, contrarily to existing models, we propose to exploit not only the interaction matrix of the feature set, but also second-order information in the

update equations. More precisely, it is proposed to predict the value of the feature vector at the next iteration using the following model, corresponding to a second-order Taylor expansion of the features evolution:

$$\boldsymbol{s}_{k+1} = \boldsymbol{s}_k + \Delta t\, \mathbf{L}_{\boldsymbol{s}} \mathbf{v}_k + \frac{1}{2}\Delta t^2 \left(\mathbf{L}_{\boldsymbol{s}} \mathbf{a}_k + \boldsymbol{h}_{\boldsymbol{s}}\right) \tag{3.36}$$

In the formula, both $\mathbf{L}_{\boldsymbol{s}}$ and $\boldsymbol{h}_{\boldsymbol{s}}$ are evaluated using the values contained in $\boldsymbol{s}_k$ and $\boldsymbol{z}_k$, *i.e.*, the interaction models are updated at each prediction step.

Regarding the estimation of the features parameters $\boldsymbol{z}_k$ along the prediction horizon, there are mainly two possibilities. The first one is to keep their value constant, either to the equilibrium value $\boldsymbol{z}^\star$ or to the last available estimation before starting the optimization. Another strategy would be to predict their value as well, which we propose to do in a similar fashion to (3.36):

$$\boldsymbol{z}_{k+1} = \boldsymbol{z}_k + \Delta t\, \mathbf{L}_{\boldsymbol{z}} \mathbf{v}_k + \frac{1}{2}\Delta t^2 \left(\mathbf{L}_{\boldsymbol{z}} \mathbf{a}_k + \boldsymbol{h}_{\boldsymbol{z}}\right) \tag{3.37}$$

In principle, the two prediction equations presented above require the spatial velocity $\mathbf{v}$ to be a state of the model as well, and the spatial acceleration $\mathbf{a}$ to be the control input. One could design a control scheme in this way and, at each iteration, recover the joint commands to be sent to the manipulator by means of its geometric jacobian. However, directly taking into account the evolution of the joint state of the robot during the predictive optimization allows to consider a broader set of constraints, such as joint limits. For this reason, we assume that the system is controlled via joint acceleration commands, *i.e.*, $\boldsymbol{u} = \ddot{\boldsymbol{q}}$. In this case, the full state of the manipulator can be selected as:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} \\ \dot{\boldsymbol{q}} \\ \boldsymbol{s} \\ \boldsymbol{z} \end{bmatrix} \tag{3.38}$$

that is, by considering the joint positions/velocities and the observed features (with the related parameters). Considering a piece-wise constant input, the joint state of the manipulator can be updated exactly from:

$$\begin{bmatrix} \boldsymbol{q}_{k+1} \\ \dot{\boldsymbol{q}}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_k + \Delta t\, \dot{\boldsymbol{q}}_k + \frac{1}{2}\Delta t^2\, \ddot{\boldsymbol{q}}_k \\ \dot{\boldsymbol{q}}_k + \Delta t\, \ddot{\boldsymbol{q}}_k \end{bmatrix} \tag{3.39}$$

whereas the second-order approximations (3.36) and (3.37) are used to evaluate the features and the related parameters. Vectors $\mathbf{v}_k$ and $\mathbf{a}_k$ are evaluated in function of joints positions, velocities and accelerations via the Forward Kinematic Model of the arm. Note that keeping track of the time derivatives of $\boldsymbol{s}$ and $\boldsymbol{z}$ inside the state is not necessary, since they can be obtained at each prediction step directly from the interaction matrix and $\mathbf{v}_k$.

As it will be shown later, the new predictor proposed above has been implemented and tested both in simulation and with the real robot, and, with the objective of evaluating its performances against classical approaches, a predictor based solely on first-order information was also considered. Its state consists in the joint coordinates $\boldsymbol{q}$ of the manipulator, altogether with the features $\boldsymbol{s}$ and the visual parameters $\boldsymbol{z}$. The control input is assumed to be the joint velocity of the manipulator, *i.e.*, $\boldsymbol{u}_k = \dot{\boldsymbol{q}}_k$, and the prediction model thus writes as:

$$\begin{bmatrix} \boldsymbol{q}_{k+1} \\ \boldsymbol{s}_{k+1} \\ \boldsymbol{z}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_k \\ \boldsymbol{s}_k \\ \boldsymbol{z}_k \end{bmatrix} + \Delta t \begin{bmatrix} \mathbf{I} \\ \mathbf{J}_{\boldsymbol{s}} \\ \mathbf{J}_{\boldsymbol{z}} \end{bmatrix} \boldsymbol{u}_k \tag{3.40}$$

wherein $\mathbf{J}_{\boldsymbol{s}}$ is the feature Jacobian of $\boldsymbol{s}$, as defined in (1.6a). Similarly, $\mathbf{J}_{\boldsymbol{z}}$ corresponds to $\mathbf{L}_{\boldsymbol{z}}{}^{c}\mathbb{T}_{w}\mathbf{J}$. In order to distinguish between this prediction model and the one that integrates also features acceleration information, we will refer to the former as $\mathcal{M}_1$ and to the latter as $\mathcal{M}_2$.

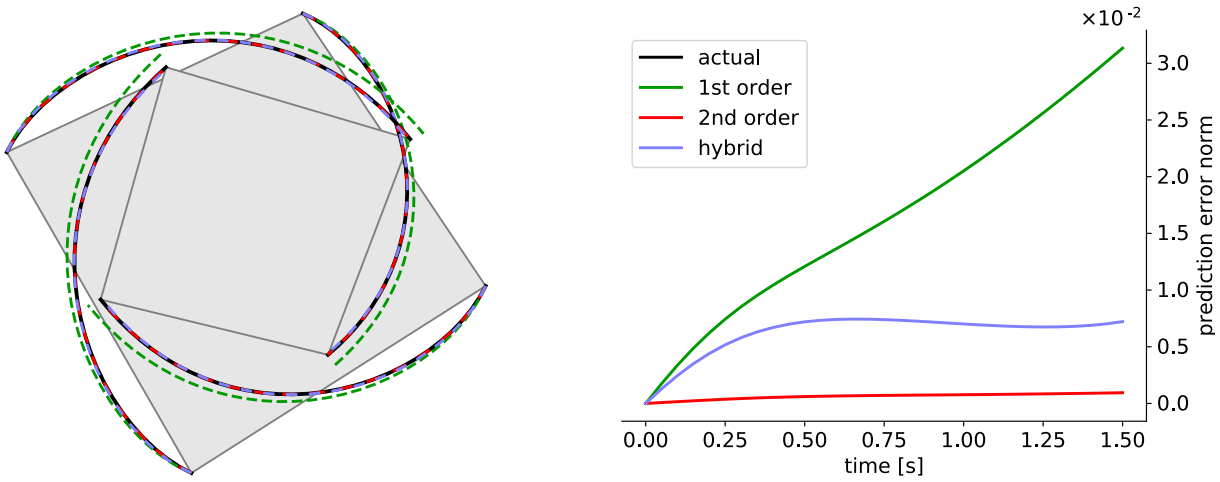
One disadvantage related to $\mathcal{M}_2$ compared to $\mathcal{M}_1$ is its larger state vector, which leads to a higher computational burden in the optimization. To overcome this issue, a new model is thus proposed, lying between classical approaches based solely on velocity information and the acceleration-controlled model presented above. Its objective is to keep the state vector of the same size as in $\mathcal{M}_1$, while integrating higher-order information similarly to $\mathcal{M}_2$. In particular, the new model assumes the control input $\boldsymbol{u}$ to be the velocity of the manipulator like in classical approaches. However, it also exploits the fact that the second term in (2.13) depends only on the current features/parameters and the velocity of the sensor. As it is done in classical local models, we assume that the twist of the camera is piece-wise constant, and at the time-sample k it is thus possible to evaluate $\mathbf{v}_k$ as ${}^{c}\mathbb{T}_{w}\mathbf{J}\boldsymbol{u}_k$. Altogether with $\boldsymbol{s}_k$ and $\boldsymbol{z}_k$, this allows to approximate features acceleration as $\ddot{\boldsymbol{s}}_k \simeq \boldsymbol{h}_{\boldsymbol{s}}$ (the acceleration of the sensor is therefore not taken into account). Using a

similar procedure to evaluate $\ddot{\boldsymbol{z}}_k$, this provides the following update scheme:

$$\begin{bmatrix} \boldsymbol{q}_{k+1} \\ \boldsymbol{s}_{k+1} \\ \boldsymbol{z}_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{q}_k + \Delta t\, \dot{\boldsymbol{q}}_k \\ \boldsymbol{s}_k + \Delta t\, \mathbf{L}_{\boldsymbol{s}} \mathbf{v}_k + \frac{1}{2}\Delta t^2\, \boldsymbol{h}_{\boldsymbol{s}} \\ \boldsymbol{z}_k + \Delta t\, \mathbf{L}_{\boldsymbol{z}} \mathbf{v}_k + \frac{1}{2}\Delta t^2\, \boldsymbol{h}_{\boldsymbol{z}} \end{bmatrix} \tag{3.41}$$

Since this model is a mix between those based on first and second order models, we like to refer to it as "hybrid" predictor, $\mathcal{M}_{\mathcal{H}}$.

To illustrate why the proposed models can lead to better results, Figure 3.7 shows a simple example of predictions made using the three schemes detailed above. This case considers a free-flying camera that observes four image points while retracting and rotating about its optical axis. The velocity and acceleration of the sensor, which are both continuous and non-constant, are sampled with a period of 50 ms, and fed to each model to perform predictions for $\boldsymbol{s}$ and $\boldsymbol{z}$ in open-loop. $\mathcal{M}_1$ slightly drifts from the actual state during time, whereas $\mathcal{M}_2$ provides the most reliable predictions. Finally, the hybrid predictor $\mathcal{M}_{\mathcal{H}}$, despite being less accurate than $\mathcal{M}_2$ due to the truncated approximation of $\ddot{\boldsymbol{s}}$, provides a rather good prediction of the real evolution.



(a) (b)

Figure 3.7 – On the left, the predictions of the evolution of a normalized point's coordinates. The solid black line is the ground-truth, while dashed lines are obtained using different models fed with the applied control inputs sampled at $\Delta t = 0.05$ s. On the right, the prediction error of each model with respect to the ground-truth.

3.3.2 Visual Predictive Control Scheme

The proposed overall VPC scheme solves the optimal control problem using a single-shooting direct method, as presented in Section 3.1 and 3.1.1 in particular, with some adaptations specific to our studied case.

First of all, the main objective of the controller is to steer the features from the initial configuration to a fixed value $\boldsymbol{s}^\star$. There is instead no need to ensure the convergence of neither $\boldsymbol{q}$ nor $\boldsymbol{z}$ to any specific value. For this reason, the matrices $\boldsymbol{Q}_{k+1}$ in the stage cost (3.9) are selected as diagonal, with the elements corresponding to the joint configuration and to the features parameters equal to zero.

Regarding the weight associated to the features, the diagonal weighting matrix $\mathbf{Q}_{k+1}^{(s)} = \alpha^k \mathbf{I}$ is adopted, α being a tunable scalar parameter, similarly to what was done in some of the related works presented in previous sections. More specifically, it appears that decreasing weighting policies are beneficial when used altogether with an online trajectory generator [CBL11; BCL14], while it was shown that increasing weights can bear to higher decoupling in the camera motion, generally shorter paths and faster convergence [AC09; ACC10]. As the scenarios presented in the following sections fall into this second category, $\alpha > 1$ was chosen.

Given the redundancy of the controlled manipulator, additional objectives can be considered in the optimization. In particular, a contribution should be added to minimize the velocity of the manipulator, since the visual task constrains only six degrees of freedom and is thus not sufficient to fully stabilize the joint position. By minimizing the velocity of the manipulator, once the robot has completed the visual task the optimal solution becomes to stop all motors, thus solving the problem. To do that, it is possible to use the constant diagonal matrix $\mathbf{Q}^{(\dot{q})} = \beta \mathbf{I}$, $\beta > 0$ controlling the relative importance with respect to the visual task. It must be noted that the objective is to minimize the joint velocity independently from the model used for predictions. Since $\mathcal{M}_1$ and $\mathcal{M}_\mathcal{H}$ both assume a velocity control, the matrices of the quadratic stage cost write in these cases as:

$$\mathbf{Q}_{k+1} = \begin{bmatrix} \mathbf{O} & & \\ & \alpha^k \mathbf{I} & \\ & & \mathbf{O} \end{bmatrix} \qquad \mathbf{R}_k = \beta \mathbf{I} \tag{3.42}$$

On the other hand, when dealing with $\mathcal{M}_2$ the joint velocity is part of the state vector.

As such, the weighting matrix is formally to be inserted as a part of $\mathbf{Q}_{k+1}$, *i.e.*,

$$\mathbf{Q}_{k+1} = \begin{bmatrix} \mathbf{O} & & & \\ & \beta\mathbf{I} & & \\ & & \alpha^k\mathbf{I} & \\ & & & \mathbf{O} \end{bmatrix} \tag{3.43}$$

with the desired state $\dot{\boldsymbol{q}}_k^\star$ simply being a zero-vector. Note that in this case, a term that tries to minimize the input is not considered, *i.e.*, $\mathbf{R}_k = \mathbf{O}$. Indeed, minimizing the acceleration would lead to a conservative velocity behavior, resulting in potential instability or longer time to convergence.

In principle, one could try to exploit the redundancy in other ways than to damp the joint velocity of the controlled system. As an example, three other criteria have been taken into consideration: the minimization of the joint efforts, of the gravitational potential and of the kinetic energy. Despite these options being attractive in theory, it turned out that they are not necessarily effective in practice. In fact, the first criterion encourages the system to move according to its free dynamic evolution, which is almost always in contrast with the Servoing task and does not guarantee the stabilization of the joint angles to a given value. Similar issues are encountered when attempting to minimize the gravitational efforts, since the optimal solution would "push" the manipulator downward, which is generally in contrast with the visual task. The optimization thus ends up in a local minimum that constitutes a compromise between the two components, without however satisfying neither of them completely. Finally, the minimization of the kinetic energy can be viewed as a special form of the quadratic stage cost, in which the weighting matrix $\mathbf{Q}^{(\dot{q})}$ corresponds to the generalized inertia matrix $\mathbf{A}(\boldsymbol{q}_k)$. In this sense, the criterion is compatible with the formulation presented above, as it simply provides a different weighting of the velocity components. However, it was found that in practice it does not bear major differences with respect to the simpler, computationally lighter, constant weighting matrix.

Two kinds of state constraints are considered during the optimization, in addition to control input bounds. The first one is introduced to keep the joint configuration of the robot within the allowed limits, *i.e.*, $\boldsymbol{q}_{min} \leq \boldsymbol{q}_k \leq \boldsymbol{q}_{max}$, $\forall k = 1, \cdots, n_p$. The second set of state constraints is introduced instead to prevent the object to leave the field of view of the camera. In particular, when servoing from image points this constraint simply writes as $\boldsymbol{s}_{min} \leq \boldsymbol{s}_k \leq \boldsymbol{s}_{max}$, with $\boldsymbol{s}$ containing the x and y coordinates of the observed points.

When polar coordinates are used, the constraints above are expressed for each point i in terms of the features ρ_i and θ_i as:

$$\begin{aligned} x_{min} &\leq \rho_{i,k} \cos\theta_{i,k} \leq x_{max} \\ y_{min} &\leq \rho_{i,k} \sin\theta_{i,k} \leq y_{max} \end{aligned} \tag{3.44}$$

3.3.3 Implementation Details

The optimization problem of the VPC scheme detailed in the previous section was solved using a single-shooting direct method. More in details, the problem was formulated as in Section 3.1.1, exploiting also the Internal Model Control Structure already adopted in [ACC10] in order to reduce the effect of modeling uncertainties.

It must be noted that, in order to achieve relatively high performances, the optimization needs to be solved rather quickly. In fact, assuming a desired control frequency of 50 Hz, the maximum allowed optimization time amounts to 20 ms. To solve the optimization it was thus decided to use Sequential Quadratic Programming, which is a fast local optimization strategy. The algorithm is well suited to the problem at hand for different reasons, the main one being related to the iterative nature of the VPC scheme. In fact, if on one hand a new optimization has to be solved at each control iteration, on the other it is also reasonable to expect that successive problem definitions – and consequently their solutions – will not differ too drastically, given the relatively high control frequency. This allows to re-use the solution to one problem instance as the initial guess for the following iteration, and to perform a local refinement to evaluate the new optimum. SQP algorithms can provide quadratic convergence to a local optimum, while simultaneously enforcing equality and inequality constraints, and they are thus an ideal candidate for VPC optimizations.

Another interesting aspect of using SQP in a single-shooting method is that, during a single optimization process, the objective is monotonically reduced during successive iterations. The consequence is that, even if the optimization is halted prematurely for some external reason, the algorithm can return a control sequence that cannot be worse than the initial guess, and hopefully is "near enough" to the optimum. In addition, the returned solution is consistent with the prediction model, since constraint (3.10a) is implicitly enforced in single-shooting methods. This aspect is important, since, as it will be detailed later in this chapter, under some circumstances the VPC optimization would have required more than 20 ms to complete. Since this is incompatible with a real-time implementation

on the real hardware, the optimization needs to be stopped even if the optimum has not been reached. As it will be shown, this is not necessarily a hard limit and the proposed schemes can still run in practice.

The only thing that is still required for the scheme to be effective, is a good initialization of the control inputs. Here, it is proposed to let a further optimization algorithm evaluate this initial guess. The computation is performed as an offline pre-optimization, which is allowed to consume a large maximum time, while the system is held static in its initial state.

To conclude this section, few details are now presented regarding the software implementation of the outlined control scheme, which ultimately resulted in a C++ library. To solve the constrained nonlinear optimization problems, the open-source library NLopt [Joh] was chosen. Among the large list of available algorithms, SLSQP [Kra88; Kra94] and ISRES [RY05] were employed. In particular, the former is a SQP implementation that was used mainly for the local optimization during each VPC iteration. It was also used to obtain a candidate guess in the pre-optimization phase, starting from a null control sequence. The second algorithm, which is instead a global optimization routine, was also exploited in the pre-optimization step to obtain a further initial guess of the control inputs. In practice, the sequences returned by the two strategies were compared at the end of the offline initialization and the best one was kept and used as seed for the VPC iterative scheme.

SLSQP requires gradients, both for the objective and the constraints, to be supplied in order to build and solve the optimization. These can be evaluated on top of (3.14) since a single shooting method is used. Given the possibly large dimensionality of the problem, the evaluation of the different functions and their gradients can become computationally expensive. Therefore, a set of libraries was selected in order to speed up as much as possible the process. In particular, the Eigen library [GJO10] was used for the matrix arithmetic, allowing to exploit processor parallelization and resulting in overall faster computations. Pinocchio [CM18; Car+19] was instead exploited to represent the manipulator and to compute its geometric and kinematic models. Finally, the code generation utilities provided by CppADCodeGen [Lea19] were used to compile the prediction models to further reduce the execution times.

3.3.4 Simulations

To investigate the performances of the proposed models, several simulations were conducted. The main goal was to gather preliminary data related to the behavior of the VPC schemes resulting from the choices detailed in the previous sections, and also to study the effect of the different parameters that appear in the problem formulation. This was also convenient in order to obtain a rough initial estimation of the values that could then be used in real experiments.

All simulations were run in a simple environment that updates the state of the robot by integrating velocity or acceleration commands, therefore not taking into account the dynamic model of the manipulator. Four image points were used as visual features, corresponding to the vertices of a squared shape of side 0.2 m. In all tests, the arm was required to reposition the camera in front of the squared object at a distance of 0.5 m, starting from different poses depending on the simulation. The object was kept static for the entire duration of each test, and thus the feature vector changed only due to the motion of the manipulator. To simulate the visual feedback gathered by a camera, the coordinates of the four points were evaluated using the geometric model of the Kuka arm, since all transformations were perfectly known in simulation. The evaluated features were finally sent at a rate of 50 Hz, to match the acquisition frequency used in real experiments, and the control period Δt_c was thus set to 20 ms. As discussed in the previous section, the optimization might require more time than the control period to be solved to the desired precision. To achieve real-time feasibility a time limit of $0.9 \cdot \Delta t_c = 18$ ms was imposed in NLopt as termination condition in addition to other stopping criteria.

Regarding the remaining parameters of the VPC scheme, the prediction horizon was set to $n_p = 10$. This implies that the control horizon n_c has to be chosen between 1 and 10, with smaller values corresponding to faster optimizations, but with less flexibility. In the next sections, simulations are reported for the two cases $n_c = 1$ and $n_c = 10$.

The sampling time Δt used in the three predictors could be chosen equal to the control period Δt_c. However, it would lead to a prediction vector spanning a rather short time period, *i.e.*, $T_p = n_p \cdot \Delta t = 200$ ms. Larger values would lead to better performances in theory, but it would require either to increment the prediction horizon or to use a larger value for Δt. While the first option implies an increase in the computational load, the second one does not change the complexity of the optimization. However, it leads to larger modeling errors due to a less accurate discrete-time approximation of the controlled system. Nonetheless, it was shown by experimental comparison that this is not a critical

issue for the proposed predictors, which are more accurate thanks to the inclusion of acceleration information. To support this claim, the included simulations consider different values for the prediction sampling time. In particular, both $\Delta t = \Delta t_c$ and $\Delta t = 5\Delta t_c = 0.1\,\mathrm{s}$ have been used.

Finally, the coefficients α and β appearing in the objective were tuned by trial and error. In particular, the process was split in two different steps, the first one consisting in fixing α to 1 while tuning β. Its value was chosen to be small compared to the one of α in order to allow motions to be rather fast in the beginning, thus achieving quicker convergence to the desired feature configuration. It was noticed that depending on the kind of control input, the value needed to be adjusted differently: $\beta = 10^{-3}$ was used for velocity-controlled models ($\mathcal{M}_1$ and $\mathcal{M}_{\mathcal{H}}$). Regarding $\mathcal{M}_2$, a smaller factor proved to work better in general, and thus β was set to 10^{-4} in that case. The second step in the tuning procedure consisted in finding a suitable value for α. As discussed in Section 3.3.2, it was decided to use a value greater than 1 in order to disfavor greedy motions that would move the features straight towards the goal configuration. It was noticed that this is a suitable choice especially when the camera has to perform large rotations, like in most of the reported simulations. The parameter was tuned by considering the weight associated to the last prediction sample, $\alpha' \doteq \alpha^{n_p} = \alpha^{10}$. By considering few different values for it, we could coarsely tune α. Our final choice was $\alpha' = 2.5$, leading to $\alpha \simeq 1.1$.

First set of simulation results

This first set of simulations was used to form a base case for later comparison. The prediction sampling time was set equal to the control period, *i.e.*, $\Delta t = \Delta t_c$, and a unit control horizon was used. This corresponds to a reduced dimension of the decision vector in the optimization, and consequently solutions can be evaluated in a relatively short amount of time.

To test and compare the performances of the different predictors, three scenarios were considered, each characterized by a different initial pose of the sensor. In the first two, the camera was located in the correct position with respect to the object, but rotated around its optical axis of either 90° or 180°. The last scenario also required the sensor to perform a half-turn around the optical axis, as well as a small translation in all directions, for a total displacement of 7 cm.

Figure 3.8 shows the results obtained in the first scenario. In the case of classical feedback laws, the paths followed by image point features in the image plane would be

almost straight lines, resulting in sensor retraction. Instead, the optimal motion would be a pure rotation about camera's optical axis, and as shown in Figures 3.8(a) and 3.8(g), the predictive laws based on $\mathcal{M}_1$ and $\mathcal{M}_{\mathcal{H}}$ are almost able to perform it. Distortions are likely introduced by the fact that the minimization of the velocity is done in the joint space, and therefore the camera velocities are not perfectly decoupled. Nonetheless, their performances are sufficiently satisfactory, especially in terms of convergence, which is achieved in about 0.5 s.

On the other hand, with the current settings the controller based on $\mathcal{M}_2$ provides unsatisfactory results. In particular, despite presenting a very smooth motion in the beginning, the sensor retracts quite significantly. Moreover, when reaching a neighborhood of the desired configuration the system shows an oscillatory behavior. The oscillations are damped along the simulation, and are eventually suppressed after a significant amount of time – more than the 3 s shown in Figure 3.8. This behavior can be explained by the fact that, when $n_c < n_p$, the last evaluated input is applied again for $n_p - n_c$ times. Since this model assumes an acceleration command, a constant input over a relatively large time interval leads to a growth in velocity and steers the system away from the desired features configuration. In addition, the prediction time interval is not long enough to anticipate and compensate for this unwanted velocity growth. It would be possible to reduce the oscillations by increasing the weight of the velocity objective coefficient, but the time required to complete the task would also grow significantly.

Before advancing, it is important to remark that the evolution of the two velocity-controlled VPCs is almost indistinguishable. This is justified by the fact that the only difference in the two predictors is given by the terms $\frac{1}{2}\Delta t^2 \boldsymbol{h_s}$ and $\frac{1}{2}\Delta t^2 \boldsymbol{h_z}$ introduced in (3.41): given that the sampling period Δt is rather small, these terms are not significant enough to make a difference in the control scheme. This holds for the other two scenarios presented in this section. However, as it will be shown later, with a larger value of Δt differences can finally be appreciated.

It is now possible to consider the second scenario, involving a π rad rotation about the optical axis, whose results are depicted in Figure 3.9. As anticipated in the remark, $\mathcal{M}_1$ and $\mathcal{M}_{\mathcal{H}}$ perform similarly, and, notably, worse than in the previous test. The reason is likely to be found in the prediction period T_p not being long enough to properly optimize the motion. Furthermore, the acceleration-controlled VPC is now unstable, with persistent oscillations, for the same reasons detailed before. Again, one way to reduce the oscillations could consist in increasing the weight associated to the velocity minimization task, but

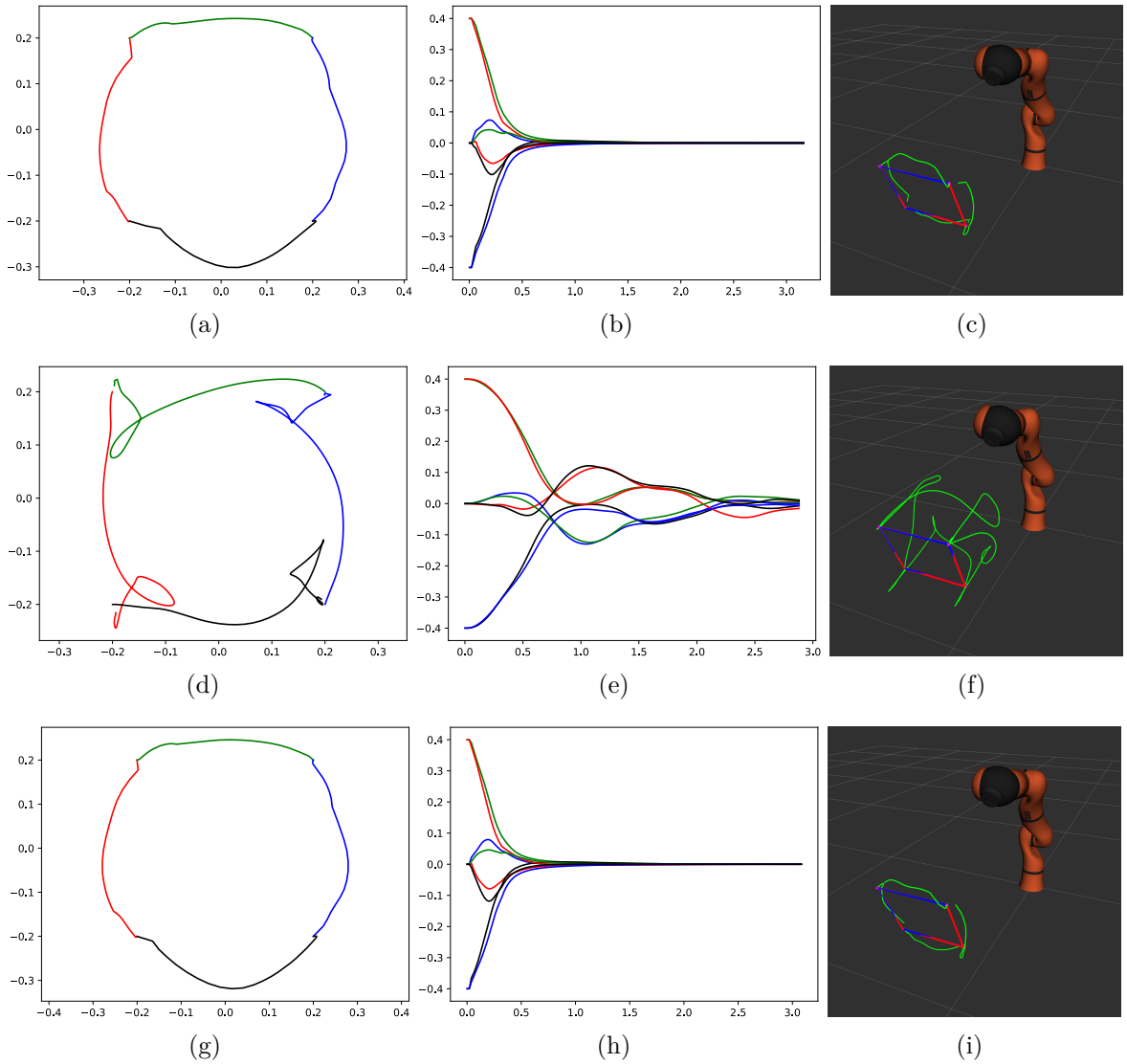


Figure 3.8 – First simulation set ($\Delta t = \Delta t_c$, $n_c = 1$), first scenario (90° rotation around the optical axis). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_{\mathcal{H}}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

the total time needed to complete the task in this case would be significantly larger than that needed by $\mathcal{M}_1$ and $\mathcal{M}_\mathcal{H}$.

To conclude this first set of simulations, Figure 3.10 shows the results obtained in the last scenario. Performances are not dissimilar to the previous case, with velocity-controlled models converging to the desired configuration with a sub-optimal motion and $\mathcal{M}_2$ still featuring large, uncontrolled, oscillations.

Second set of simulation results

Results reported so far seem to suggest that there is little interest in using the models proposed earlier, since $\mathcal{M}_\mathcal{H}$ performs very similarly to $\mathcal{M}_1$ and $\mathcal{M}_2$ seems to be impractical when medium-large motions have to be performed. However, the results reported in this section will show that this is not always true, and that the performances of the proposed models can fortunately be increased.

One issue that appears clearly in all approaches is that, having a high control frequency, the prediction horizon is not long enough to foresee what the optimal motion should be in the case of large displacements. In the presented setup, the prediction horizon allows in fact to predict over a time interval of just 0.2 s. It is argued here that this is one of the key factors limiting the performances of the models in the previous simulations. As already mentioned, one possible way to improve performances would therefore be to increase n_p. However, this would also imply a larger computational cost. One alternative is instead to use a larger time discretization step Δt in the prediction models, as this allows to increase the time horizon without having to change n_p, thus keeping the same complexity for the optimization. This second option has its own different drawback: it introduces larger modeling errors, since the predictors are obtained from a truncated expansion of the real features evolution. Nonetheless, it is claimed here that models including the acceleration information are less affected by this issue, since they are intrinsically more accurate. In conclusion, in the attempt to obtain better performances the tests shown in this section all used a larger sampling time, $\Delta t = 5\Delta t_c = 100\,\text{ms}$.

Note that the increased sampling time in the prediction does not mean that the control frequency is lowered as well. On the contrary, the optimization is still repeated every $\Delta t_c = 20\,\text{ms}$. Considering this fact, a slight change is also introduced in the definition of the seed fed to the optimization algorithm at the next iteration. As remarked in Section 3.3.3, the SQP algorithm used for the optimization successively refines an initial guess supplied by the user, and providing a good starting point is important in order to achieve faster

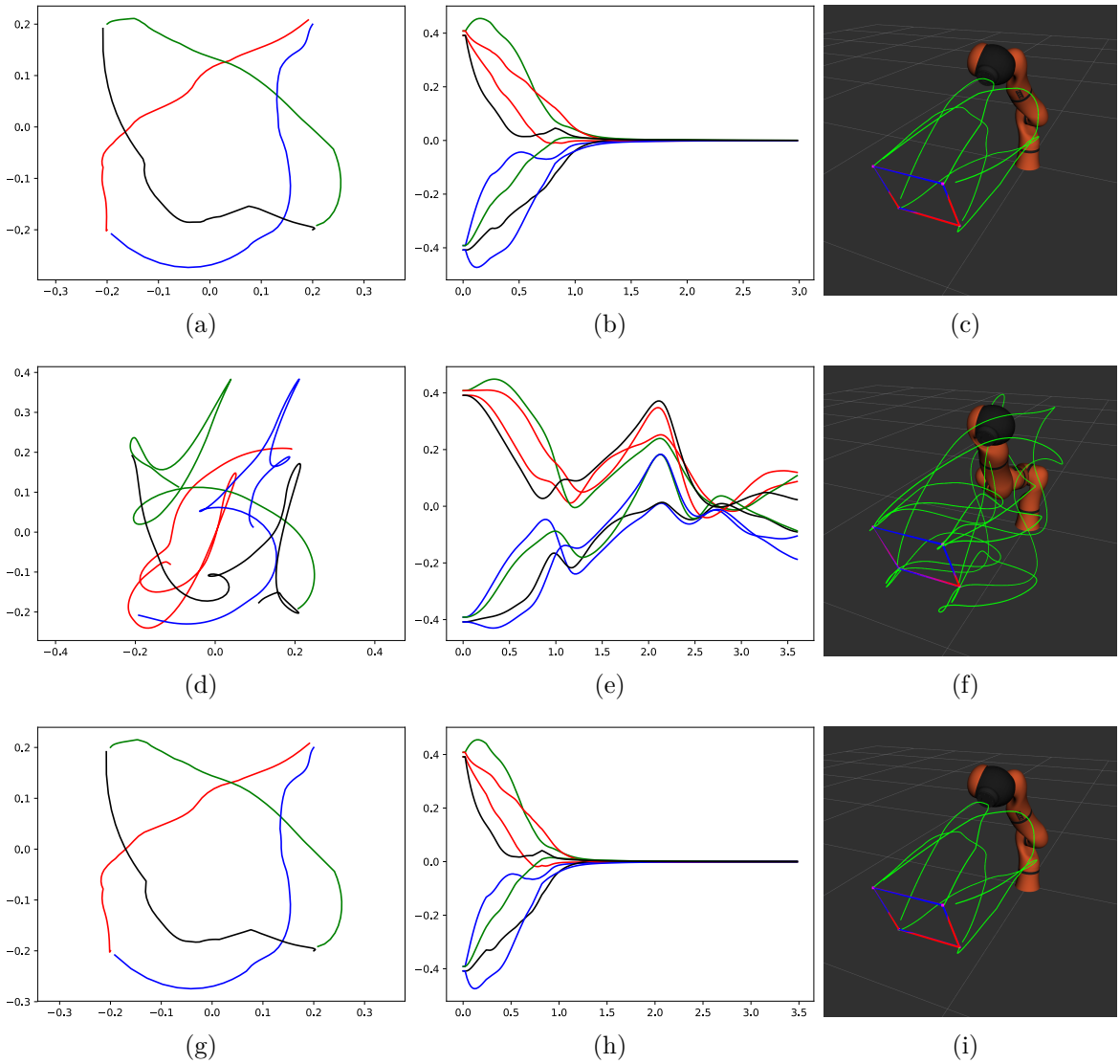


Figure 3.9 – First simulation set ($\Delta t = \Delta t_c$, $n_c = 1$), second scenario (180° rotation around the optical axis). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_{\mathcal{H}}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

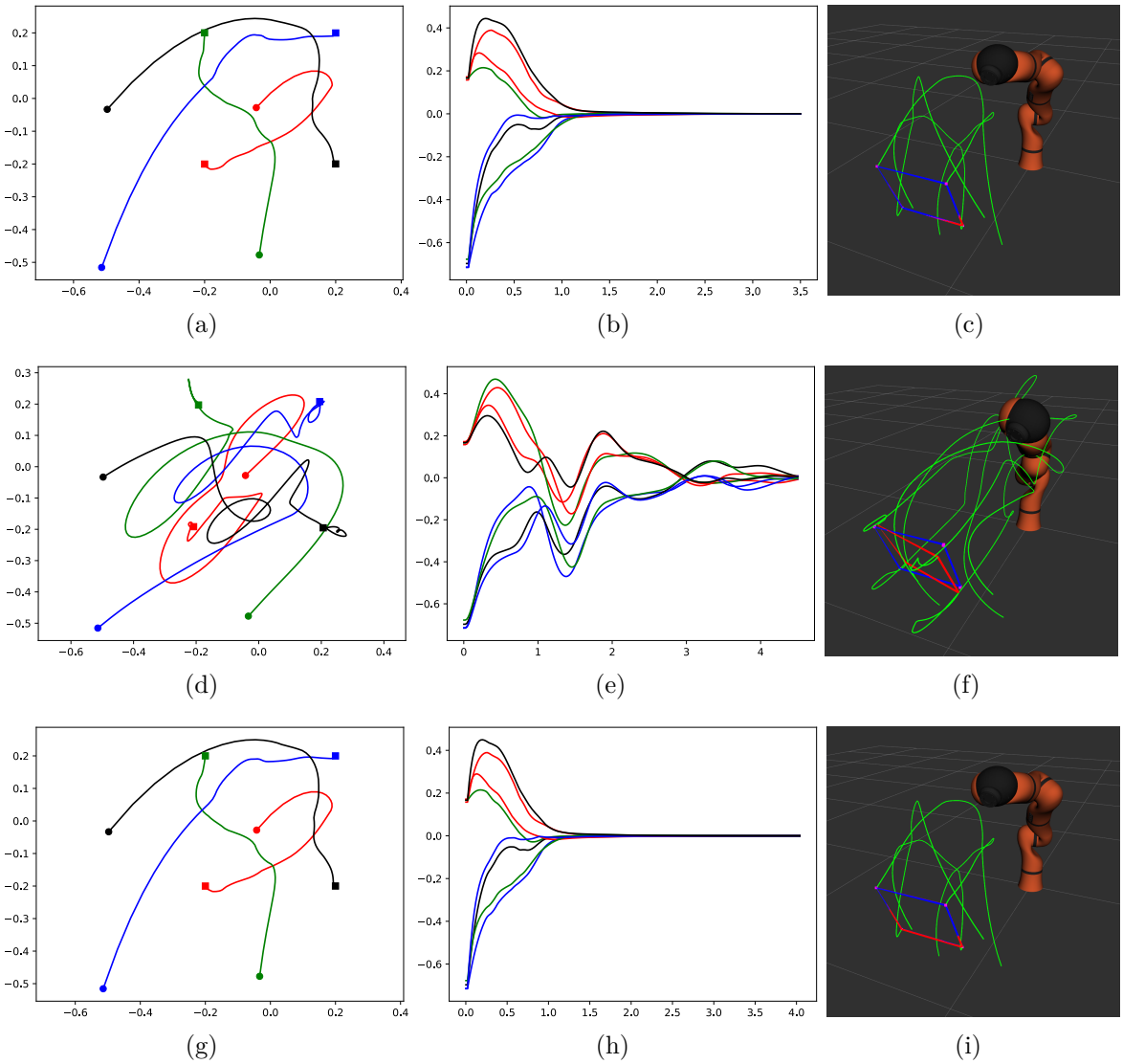


Figure 3.10 – First simulation set ($\Delta t = \Delta t_c$, $n_c = 1$), third scenario (180° around the optical axis and short translation). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_\mathcal{H}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

performances. Thus, to help the optimization start from a better point, we propose a very simple heuristic in which the controls $\boldsymbol{u}_k^{(init)}$ used for initialization of the next step are evaluated from the last available solution samples $\boldsymbol{u}_k$ by interpolation:

$$\boldsymbol{u}_k^{(init)} = \begin{cases} \boldsymbol{u}_k + \frac{\Delta t_c}{\Delta t}\left(\boldsymbol{u}_{k+1} - \boldsymbol{u}_k\right) & (k < n_c - 1) \\ \boldsymbol{u}_k & (k = n_c - 1) \end{cases} \tag{3.45}$$

The same tests presented in the previous paragraph were then repeated with the increased time-step. Results from the first scenario, reported in Figure 3.11, show that good motions are still found by both $\mathcal{M}_1$ and $\mathcal{M}_{\mathcal{H}}$, apparently even a little smoother than before. Interestingly, the motion obtained in the space using $\mathcal{M}_2$ is much shorter than before, and the evolution in the image space is also closer to that of the other two predictors. However, oscillations near to the target configuration are still present.

It must be noted that the convergence time is higher in comparison with the previous simulations. A possible explanation for this behavior is the simultaneous use of a longer sampling time Δt and of a unitary control horizon. To explain why this might be the cause for an overall slower evolution, imagine the system to be like a car that has to drive at constant velocity v for a fixed amount of time T_p. If the goal is to reach a location at distance d, then the "optimal" velocity at which the car should proceed in order to be at the target after the time T_p would trivially be $v = d/T_p$. Even if this does not exactly describe what happens in the simulations with the manipulator, it provides an close enough example: if T_p is increased – which is exactly what has been done in this section, by increasing Δt – we can expect velocities to be lower, justifying the increment in time to complete the task.

The results from the other two scenarios are particularly interesting. Specifically, when the camera has to perform only the 180° rotation (see Figure 3.12) two facts should be noted. First of all, the performances obtained with $\mathcal{M}_2$ are similar to those of the classical predictor. Secondly, $\mathcal{M}_{\mathcal{H}}$ now performed a much shorter trajectory compared to the one obtained previously and shown in Figure 3.9. Unfortunately, like in the first scenario, the execution of the task is slower compared to the timing performances obtained in the previous section.

The third scenario offers similar results. As depicted in Figure 3.13, $\mathcal{M}_2$ almost provides a nice motion, was not for the familiar oscillations that appear as soon as the camera has reached a neighborhood of the desired configuration. Moreover, $\mathcal{M}_{\mathcal{H}}$ ranks first also in this scenario, with the shortest motion among the three predictors.

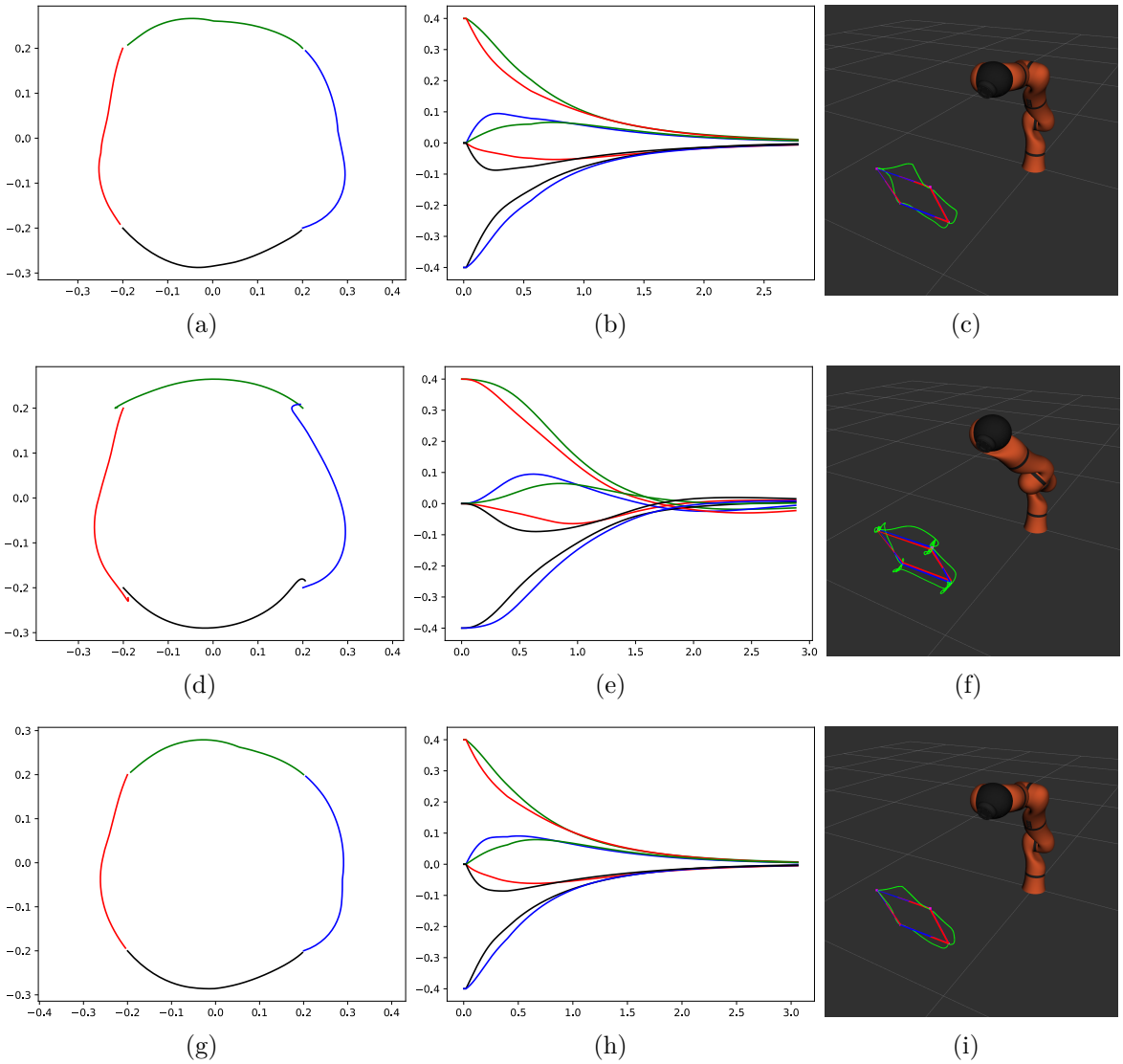


Figure 3.11 – Second simulation set ($\Delta t = 5\Delta t_c$, $n_c = 1$), first scenario (90° rotation around the optical axis). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_{\mathcal{H}}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

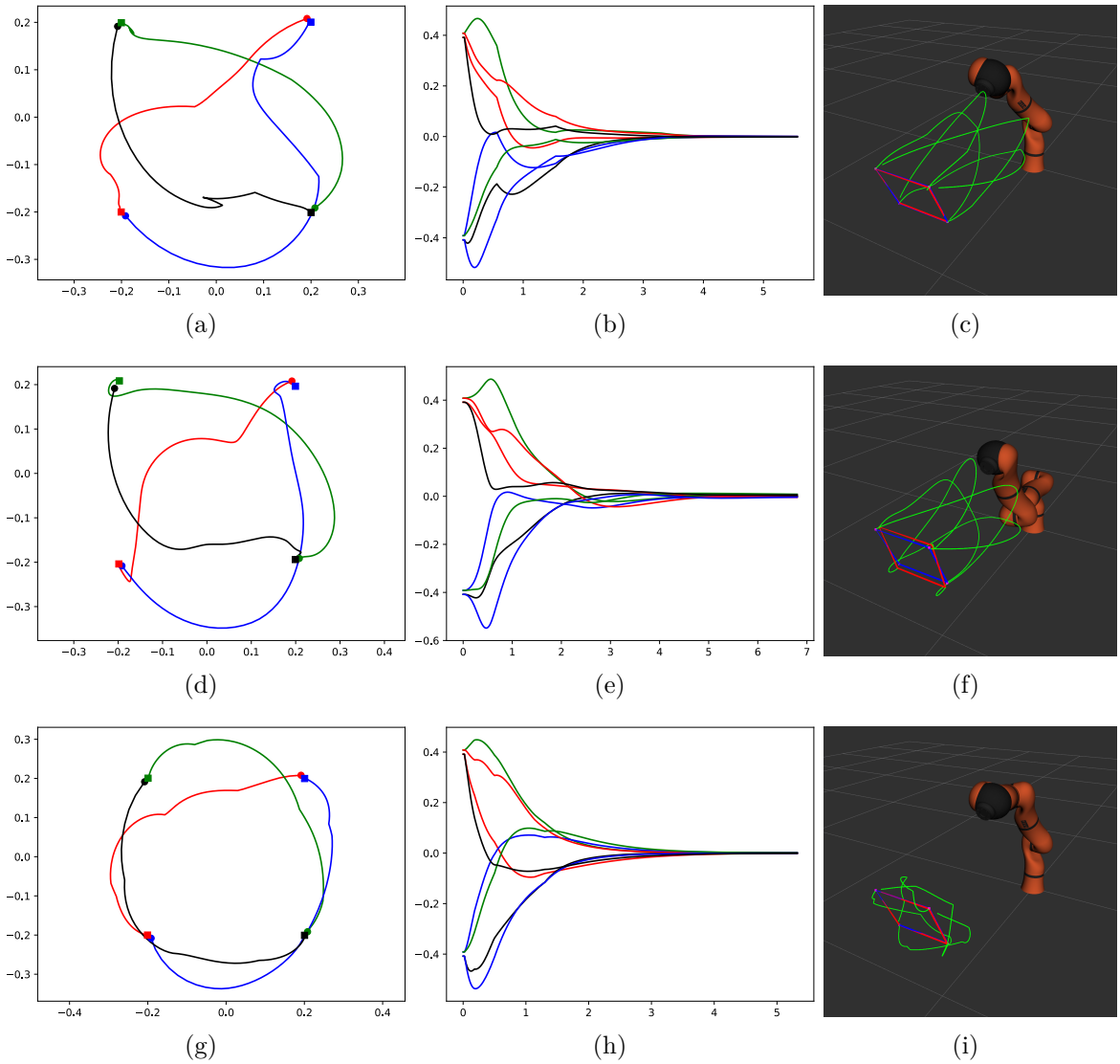


Figure 3.12 – Second simulation set ($\Delta t = 5\Delta t_c$, $n_c = 1$), second scenario (180° rotation around the optical axis). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_{\mathcal{H}}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

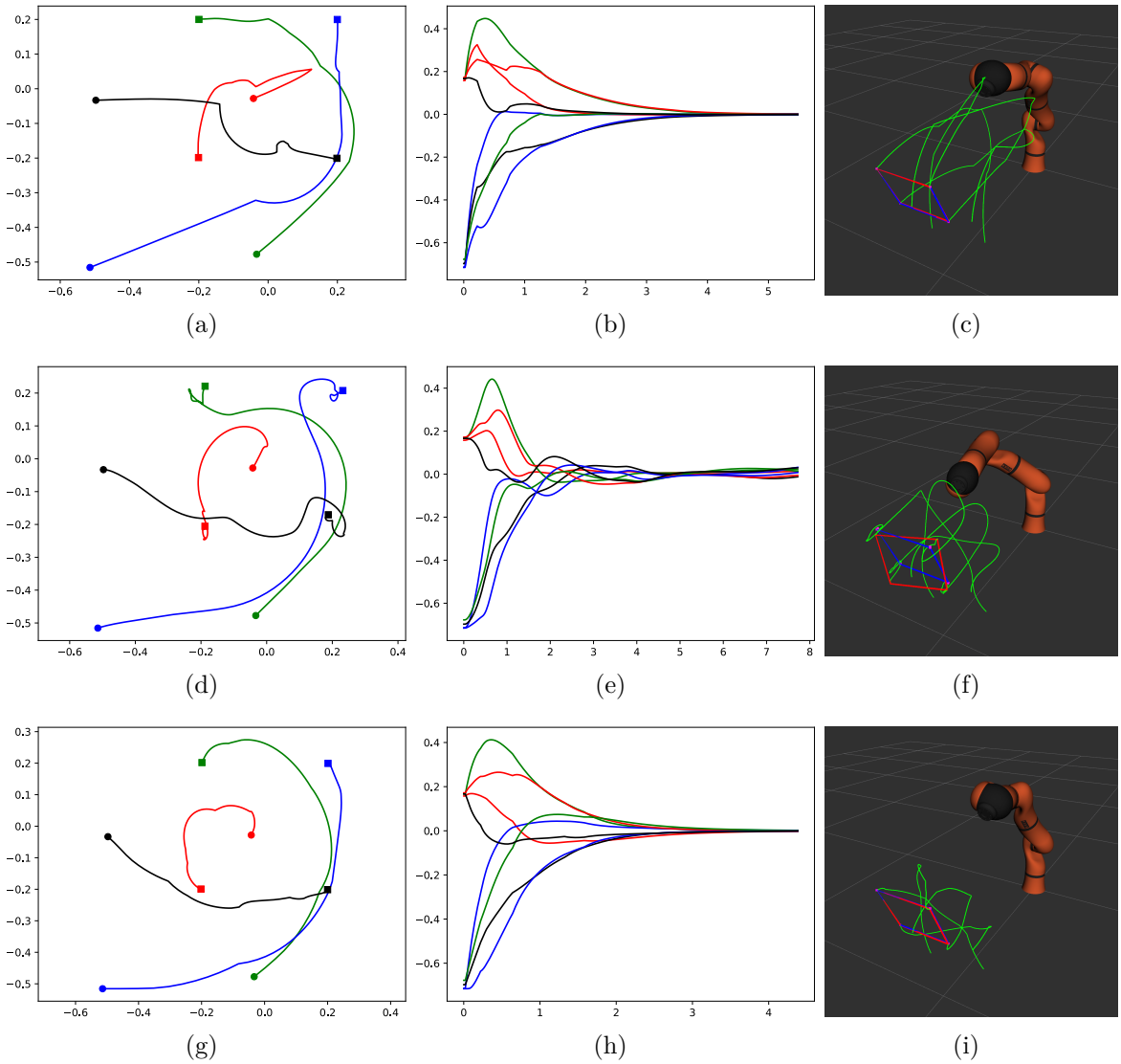


Figure 3.13 – Second simulation set ($\Delta t = 5\Delta t_c$, $n_c = 1$), third scenario (180° around the optical axis and short translation). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_{\mathcal{H}}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

Third set of simulation results

It has just been shown that using a larger discretization step allows to obtain better results with the proposed predictors, as they can take into account states that are farther in the future and consequently optimize their motion. A side-effect was however found, ultimately resulting in poorer timing performances. As discussed, this is likely due to the fact that, as a single control sample has to be re-used over the whole prediction, it has to ensure that no overshoots appear.

A better trade-off could be found by increasing n_c. The optimization algorithm would have in this way the ability to exploit larger inputs at least in the beginning of the prediction horizon, slowing down the robot with the last control samples once the sensor has already approached the desired configuration.

Another reason for considering the use of larger control horizons is related to $\mathcal{M}_2$. Up to now, no stable behavior could be fully ensured using this model, ending up almost always with oscillations. As already suggested, this might be due to the fact that a single acceleration signal prevents the system to stably reach the desired image configuration and stop the robot at the same time. Not forcing the optimizer to re-use the same acceleration sample along the whole prediction horizon would hopefully allow to find better motions, in which the system initially gains velocity and then decelerates as needed.

For the reasons above, a further set of simulations was run, featuring a larger control horizon. In particular, $n_c = n_p = 10$ was used, providing the highest number of degrees of freedom to the optimization. This came at the price of a heavier computational load, causing the optimizations in each simulation to halt due to the time-limit criterion. Nonetheless, this was not a major limitation, as discussed later.

Figure 3.14 depicts the results obtained with the increased control horizon in the first scenario. The superiority of the proposed predictors is here arguably undeniable. First of all, the features are moved by $\mathcal{M}_1$ along the sides of an almost rectangular shape, which somehow reminds of the trajectories obtained with proportional controllers, also resulting in retraction. On the other hand, both $\mathcal{M}_2$ and $\mathcal{M}_\mathcal{H}$ lead to better motions: the motion of the sensor is much closer to a pure rotation, with smooth movements and nearly no retraction. Remarkably, $\mathcal{M}_2$ is almost no longer affected by oscillations, with just a slight overshoot in the features error (see Figure 3.14(e)), yet far from what it used to be in the previous tests. Finally, the timing performances are also satisfactory, suggesting the correctness of the explanation for the longer execution times provided in the previous section.

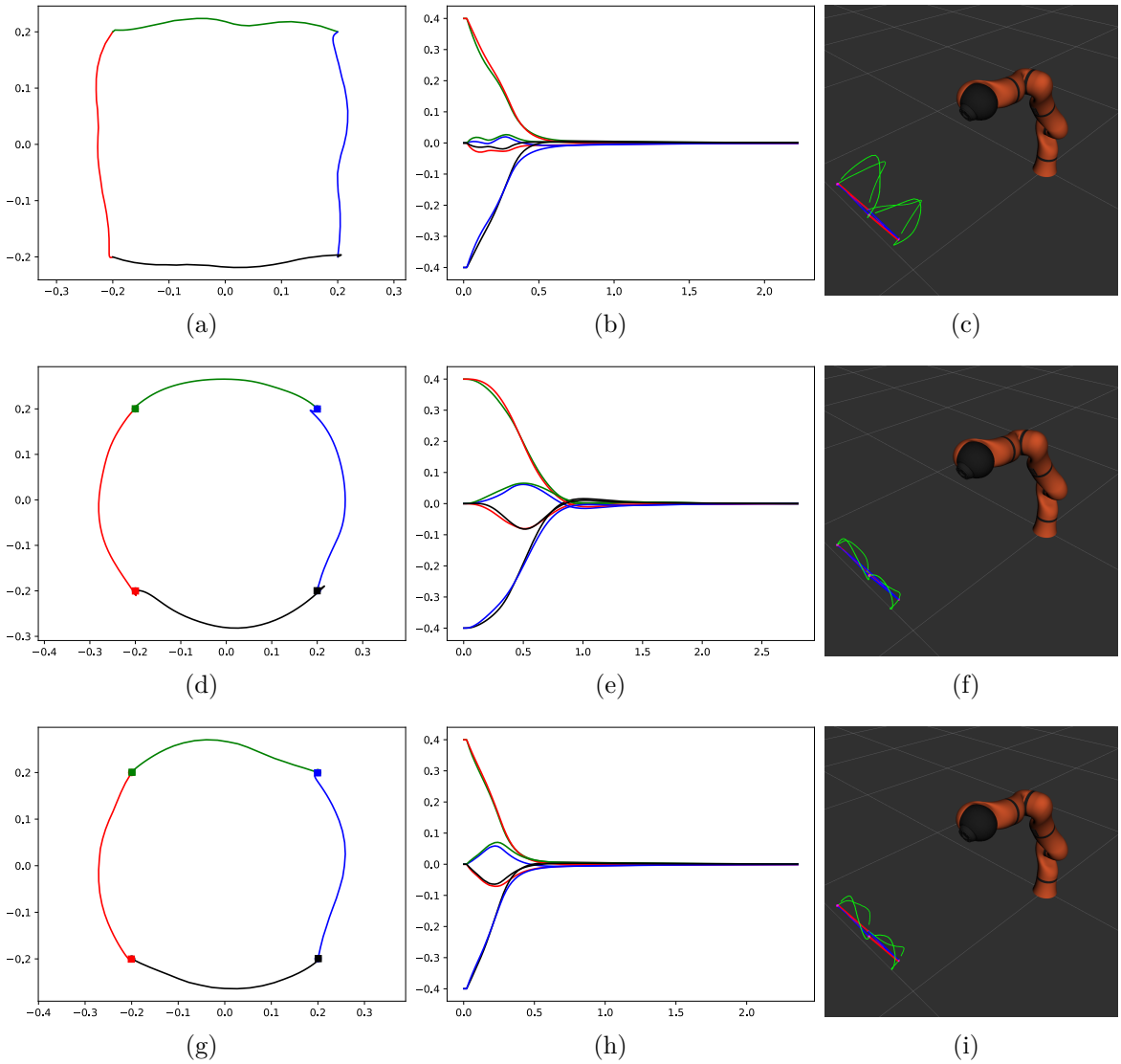


Figure 3.14 – Third simulation set ($\Delta t = 5\Delta t_c$, $n_c = 10$), first scenario (90° rotation around the optical axis). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_{\mathcal{H}}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

Similar results are found when the sensor has to perform a doubled rotation, as shown in Figure 3.15, as well as in presence of translation, *cfr.* Figure 3.16. All VPC schemes are able to quickly complete the servoing task in both scenarios. However, $\mathcal{M}_1$ appears to be the worst strategy, in the sense that the observed image features present discontinuities and the resulting 3D behavior is the one with the largest retraction. The proposed predictors lead instead to smooth trajectories with a more decoupled sensor motion in Cartesian space.

Discussion

Before presenting the experimental results obtained with the real robot in the next section, few remarks are included to clarify the influence of some factors that were not discussed in the previous sections.

Let us start with a note on the time required to solve the optimizations. Table 3.1 reports the average computation times needed by each optimization strategy depending on the test set and the scenario. When considering a simple optimization with just a single control sample (test sets 1 and 2) the optimization can be completed much earlier than the maximum allowed time. However, when the control horizon is increased all strategies do not actually converge to the minimum within the allotted time. Instead, they simply return the best control sequence found so far. One obvious question is whether the performances of $\mathcal{M}_1$ could be improved to match those of the other models simply by allowing longer computations, and in such case, how much time would be needed. The last set of simulations was therefore repeated by changing the time limit, while keeping the control frequency unchanged (this investigation was thus possible only in simulation). It turns out that the performances of the three predictors are not significantly enhanced even if the controller is allowed to take as much as 200 ms to solve the minimization problem. Conversely, the time limit was also reduced to better understand when the premature stopping can induce instability in the control. In this sense, the performances were not heavily affected until the limit was reduced to less than 13 ms.

Another aspect worth discussing is the sensitivity of the three VPCs to changes in the objective coefficients. Since these values are tuned by trial and error, it is interesting to better understand how precisely (or coarsely) they need to be estimated. In this regard, it was noted that the parameter β does not need to be finely tuned and that it is instead sufficient to find a suitable order of magnitude. The situation is different when tuning the value of α. Since it is the base of an exponential term, small changes can lead to

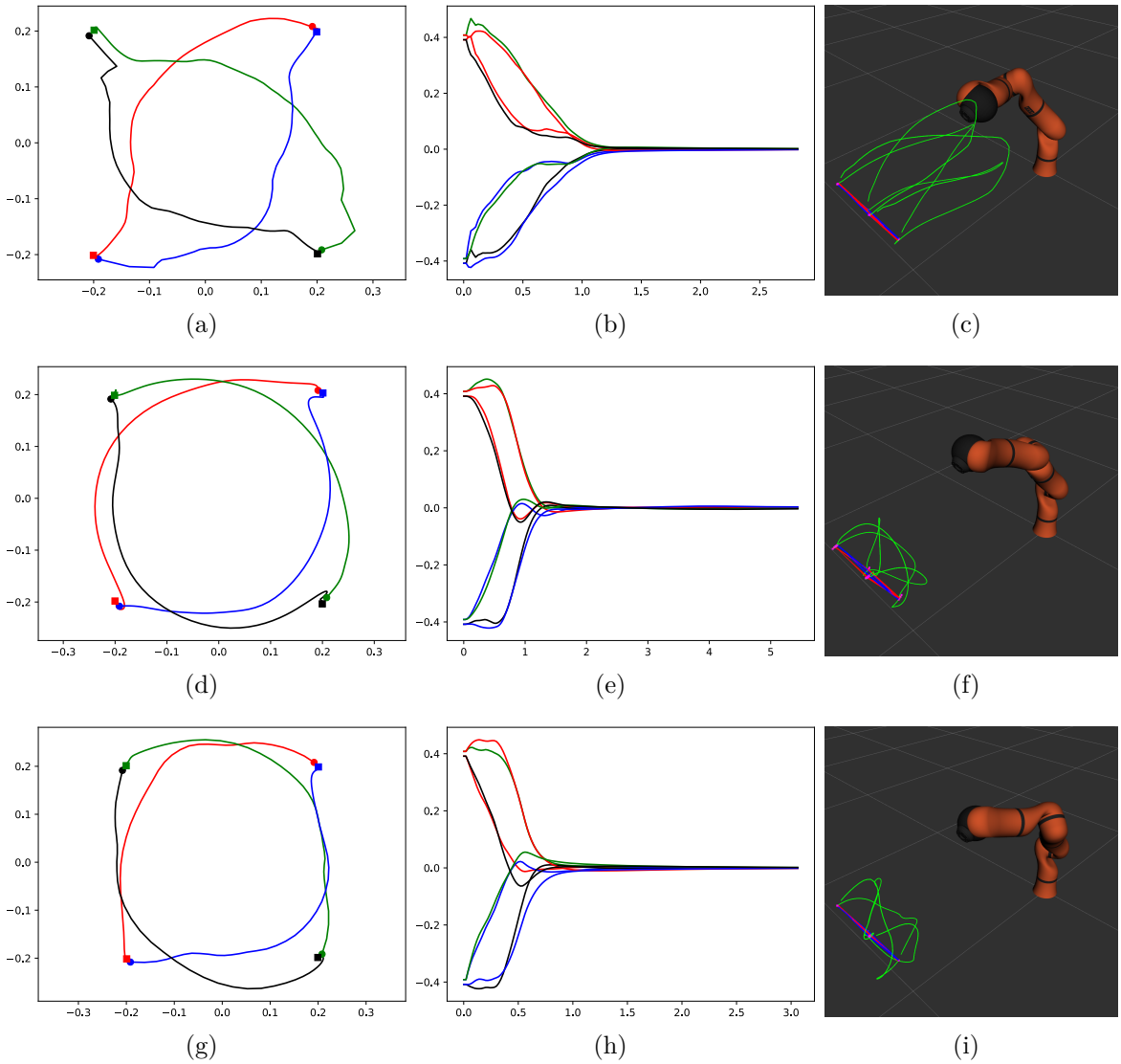


Figure 3.15 – Third simulation set ($\Delta t = 5\Delta t_c$, $n_c = 10$), second scenario (180° rotation around the optical axis). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_\mathcal{H}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

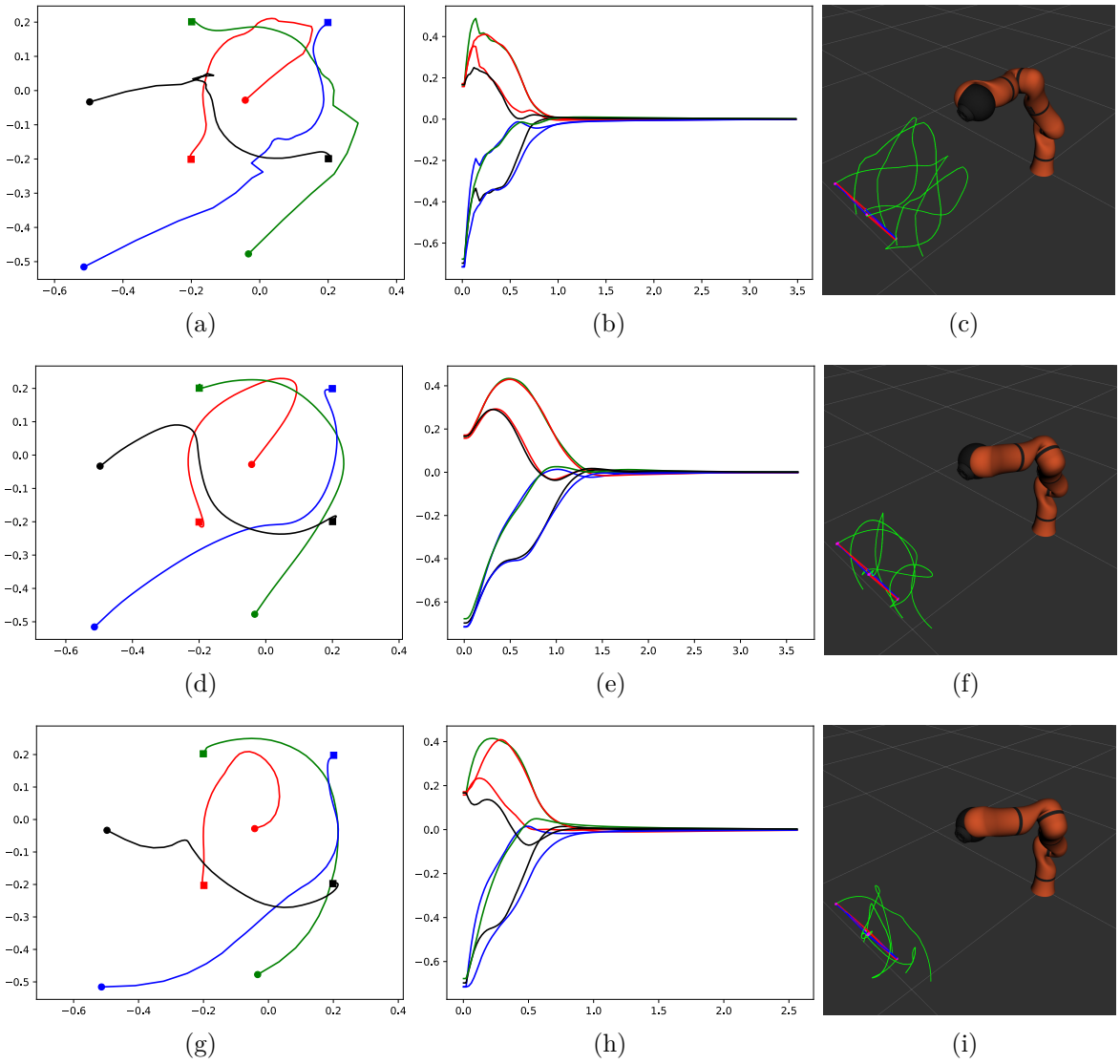


Figure 3.16 – Third simulation set ($\Delta t = 5\Delta t_c$, $n_c = 10$), third scenario (180° around the optical axis and short translation). The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_\mathcal{H}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

Table 3.1 – Computation time (wall clock) for the different VPC strategies.

Test Set	Scenario	Model	Average time
1	1	$\mathcal{M}_1$	2.06 ms
		$\mathcal{M}_2$	6.30 ms
		$\mathcal{M}_\mathcal{H}$	3.54 ms
	2	$\mathcal{M}_1$	1.43 ms
		$\mathcal{M}_2$	3.72 ms
		$\mathcal{M}_\mathcal{H}$	3.38 ms
	3	$\mathcal{M}_1$	1.80 ms
		$\mathcal{M}_2$	3.79 ms
		$\mathcal{M}_\mathcal{H}$	2.83 ms
2	1	$\mathcal{M}_1$	1.20 ms
		$\mathcal{M}_2$	4.20 ms
		$\mathcal{M}_\mathcal{H}$	3.09 ms
	2	$\mathcal{M}_1$	1.18 ms
		$\mathcal{M}_2$	4.30 ms
		$\mathcal{M}_\mathcal{H}$	3.01 ms
	3	$\mathcal{M}_1$	1.32 ms
		$\mathcal{M}_2$	3.53 ms
		$\mathcal{M}_\mathcal{H}$	2.69 ms
3	1	$\mathcal{M}_1$	18 ms
		$\mathcal{M}_2$	18 ms
		$\mathcal{M}_\mathcal{H}$	18 ms
	2	$\mathcal{M}_1$	18 ms
		$\mathcal{M}_2$	18 ms
		$\mathcal{M}_\mathcal{H}$	18 ms
	3	$\mathcal{M}_1$	18 ms
		$\mathcal{M}_2$	18 ms
		$\mathcal{M}_\mathcal{H}$	18 ms

completely different values of the individual state weights in the objective. This is why, while discussing the tuning selected for the presented simulations, it was proposed to select a value for α^{n_p} rather than a direct selection of α. Indeed, it was found that the predictive schemes are much less sensitive to changes in the value of α^{n_p}, which also provides a more intuitive way of tuning the feature weights.

Finally, in all simulations presented so far, no rotations around the X and Y axes of the sensor were necessary, and the translational motion required in the last scenario was rather small. The choice of focusing on rotational motions was justified by the ease of interpretation of the results. However, in order to show that the performances obtained with the proposed predictors are consistent with other types of scenarios, we conclude this section by including a further test. In this case, the motion to be performed is more generic. A longer translation is required (20 cm in total), as well as a generic rotation of 15°, 30° and 165° around the X, Y and Z axes respectively. The motions are depicted in Figure 3.17, and were obtained with the exact same parameters used in the third set of simulations presented in Section 3.3.4. The proposed predictors provide smoother motions, less retraction and take slightly less to complete the servoing task, confirming the results found so far.

3.3.5 Experiments

This section details the experimental results obtained while comparing the different models. Similarly to what was done in simulation, the task consists in positioning the sensor in front of a planar object with four black circles, detected using the ViSP library [MSC05]. In addition, in order to support the effectiveness and generality of the proposed approaches, the controllers were tested with different features: standard image point coordinates were firstly considered, as in the simulations included above. In a second moment, also the polar coordinates of the centers of the four circles printed on the object were used (their second-order models can be found in Appendix A.3). In both cases, our predictors outperformed classical local models, confirming the validity of our work.

Let us mention for sake of completeness that since our hardware does not yet allow to directly control the robot in acceleration, a simple workaround was exploited. In order to test $\mathcal{M}_2$, we numerically integrated the acceleration commands produced by the VPC and sent the resulting signal to the low-level velocity controller running on our robot.

Reported experiments all used the same settings as in the last simulations set, *i.e.*, the prediction and control horizon were both set to 10 samples, and the prediction sampling

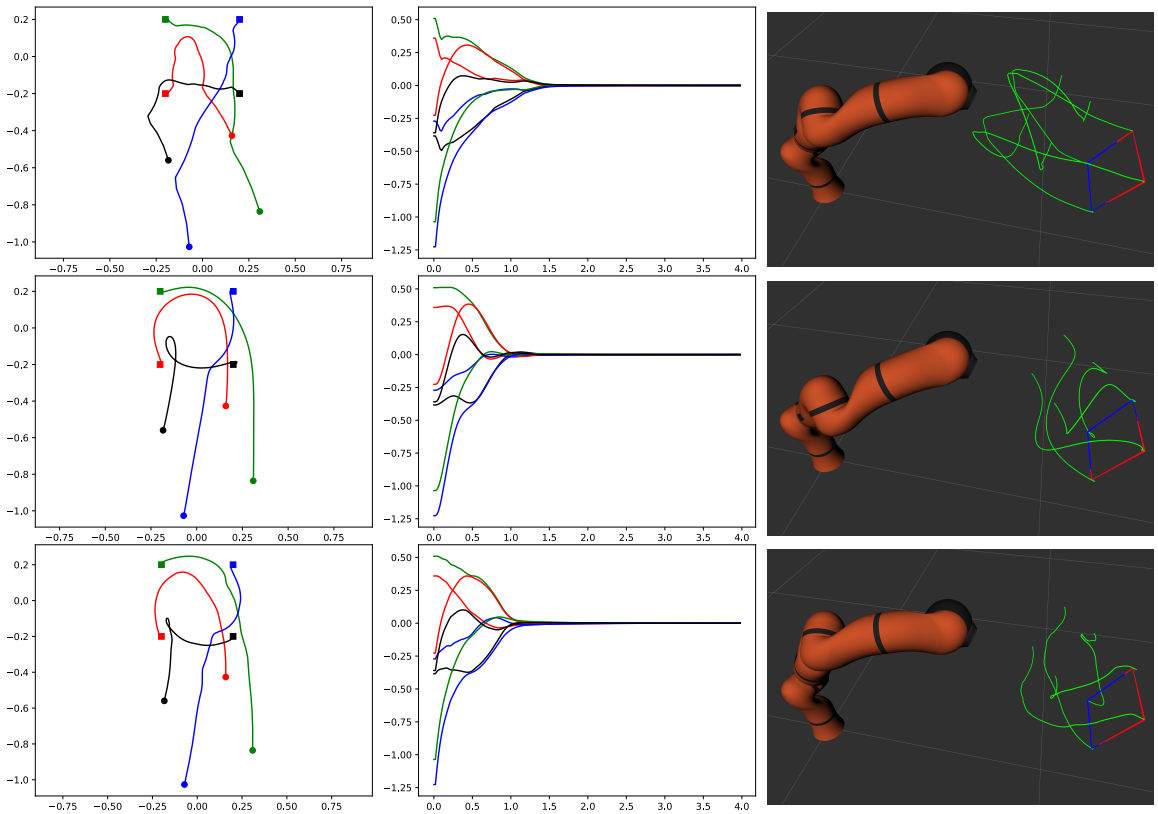


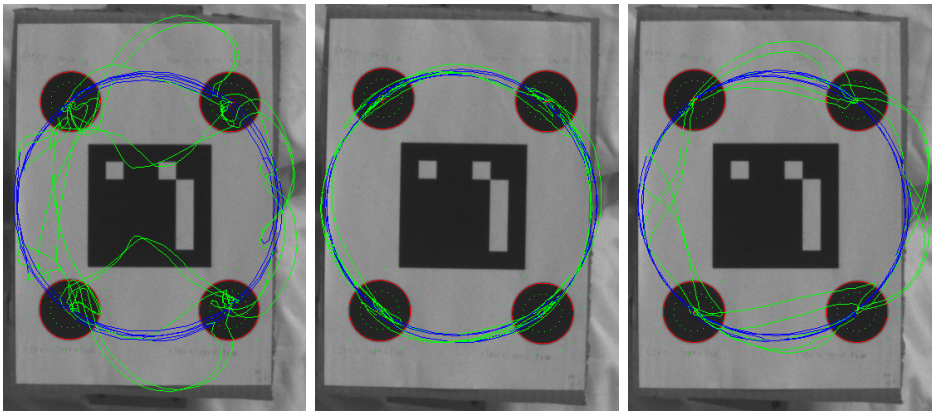
Figure 3.17 – Last simulation test, which uses $\Delta t = 5\Delta t_c$ and $n_c = 10$. In addition to a 20 cm translation, the sensor needs to perform rotations of 15°, 30° and 165° around the X, Y and Z axes. The rows correspond to $\mathcal{M}_1$ (top), $\mathcal{M}_2$ (middle), $\mathcal{M}_\mathcal{H}$ (bottom), while the columns to the features trajectories in the image (left), the feature error (center) and a 3D view of the manipulator (right).

time Δt was set to 0.1 s. The weights in the objective were the only parameters that needed to be adjusted, as it will be discussed in each experiment.

Servoing from Image Points

The tests reported in the following were performed by placing the planar object in front of the robot and rotating it multiple times about its normal axis, of an angle of almost 90°. Since the configuration $\boldsymbol{s}^\star$ was still kept to a constant value, the robot is ideally supposed to follow the motion of the object.

As anticipated, experiments were run with almost the same settings used in the last sets of simulations. In particular, in this this test only the value of β needed to be modified. It was in fact noticed that a larger value better suits real experiments in which noise is also present. Increasing the value of this parameter mainly reduces the overall speed of the system, but also smooths the resulting trajectories. In practice, the parameter was modified from the original 10^{-3} to 10^{-2} for the two velocity-controlled schemes, $\mathcal{M}_1$ and $\mathcal{M}_\mathcal{H}$. In the case of $\mathcal{M}_2$, it was necessary to increase it slightly more in order to reduce oscillations and prevent the velocity to grow excessively. The final choice was $\beta = 5 \cdot 10^{-3}$, which, although not sufficient to completely remove all oscillations, was found to be a good trade-off in order to maintain acceptable time to convergence.



(a) $\mathcal{M}_1$ (b) $\mathcal{M}_2$ (c) $\mathcal{M}_\mathcal{H}$

Figure 3.18 – Features trajectory in the image using respectively $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_\mathcal{H}$. Blue segments correspond to features displacements induced by the object rotation, while green ones are due to the control moving the camera.

An example of features trajectories obtained with the three predictors is visible in

Figure 3.18 [1]. Note that the points in the image can change not only due to the motion of the robotic arm, but also due to the aforementioned rotation of the object itself. While analyzing the recorded image streams produced by the camera, it was noticed that controllers necessitate a short amount of time before reacting to the object motion. To help interpreting the results, the paths crossed by the features in these phases are colored in blue, and correspond to nearly perfect circles. Features motions induced by the action of the VPCs are instead colored in green.

As the results show, the features behavior is quite different depending on the involved predictor. $\mathcal{M}_1$ is not able to perform a nice rotation, with the features going either too near or too far from the ideal circumference. This also results in an undesirable motion in Cartesian space. On the contrary, the trajectories obtained using $\mathcal{M}_2$ are much closer to a pure rotation about the optical center of the camera, and the motion in 3D is satisfactory as well. Finally, $\mathcal{M}_{\mathcal{H}}$ gives intermediate results, with features paths that, despite not being as nice as in the case of the acceleration-controlled predictor, are not deviating too much from a pure rotation. Furthermore, the motion of the sensor features almost no retraction.

The evolution of the feature error during time for the first rotation of the object is also reported in Figure 3.19. It can be seen that the predictor based solely on the interaction matrix is outperformed by $\mathcal{M}_{\mathcal{H}}$, taking almost half a second longer to reach the same error magnitude. It should also be noted that $\mathcal{M}_2$, besides being the slowest controller, presents a rather large overshoot, which is hard to observe from Figure 3.18(b) alone. Such oscillatory behavior, as mentioned in the beginning of this section, could be reduced by increasing β, but at the cost of further sacrificing time performances. We believe that one explanation for these oscillations is to be found in the use of the numerical integrator to obtain the velocity command from the acceleration. As neither this block nor the low-level velocity controller are taken into account in the prediction, model uncertainties become larger and predictions are thus less reliable, finally leading to lower performances in practice.

Servoing from Polar Coordinates

The goal of this last set of experiments is to check if the performances of the proposed predictors remain consistent also when different features are selected. In particular, the polar coordinates of the point centers, whose second-order models are reported in Appendix A.3, are used in the sequel.

1. A video is also available online at `https://youtu.be/xJhzVkGdXPw`

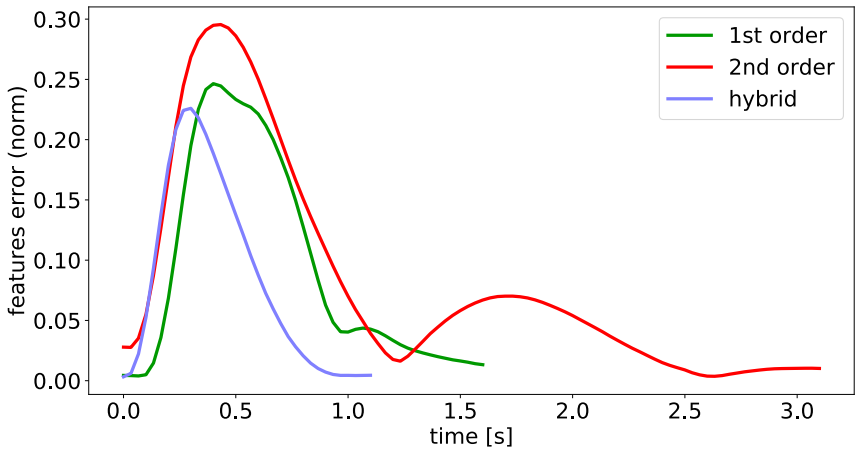


Figure 3.19 – Evolution of the features error norm, while servoing from four image point coordinates. The plotted data correspond to the signals obtained during the first rotation of the object.

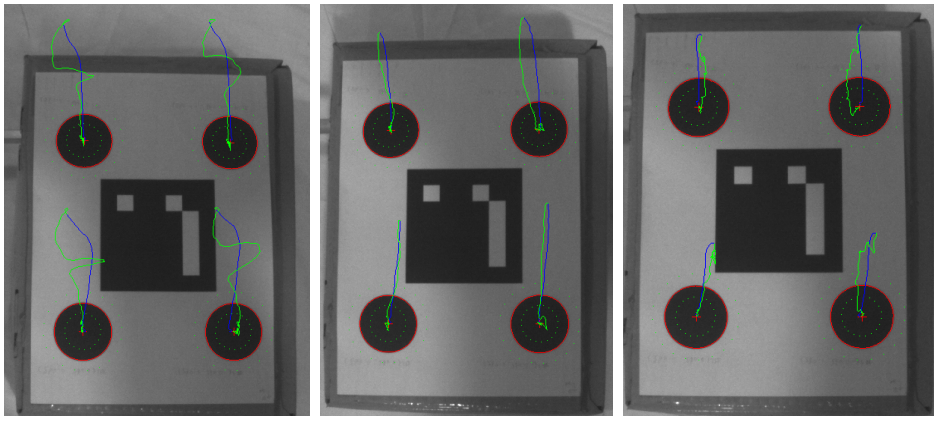
In these experiments almost all parameters were kept unchanged from the previous section, with only a small modification in the objective being necessary. In fact, when considering polar coordinates as features, the angles e_{θ_i} (expressed in radians) tend to vary more than the radii ρ_i for the same displacement. For this reason, their weight in the objective was reduced, so that angular and radial errors bore more homogeneous contributions to the objective. Specifically, the matrix $\mathbf{Q}_{k+1}^{(s)}$ was updated from being simply $\alpha^k \mathbf{I}$ to:

$$\mathbf{Q}_{k+1}^{(s)} = \alpha_k \, \mathrm{diag}(1, 1/5, 1, 1/5, 1, 1/5, 1, 1/5) \tag{3.46}$$

It is well-known that, in classical servoing via feedback linearization, polar coordinates perform well when the object is subject to a pure rotation around the optical axis of the camera. Hence, the experiment of this section focus on the complementary case, in which the object performs a relatively large translational motion. This is known to lead to less satisfactory trajectories for the system due to the non-linearities in the interaction matrix.

The results obtained in this experiment, visible in Figures 3.20 and 3.21, are coherent with previous findings. In terms of points trajectories, $\mathcal{M}_1$ is the predictor that features the worst results, while motions obtained with $\mathcal{M}_2$ and $\mathcal{M}_\mathcal{H}$ are more regular and closer to straight lines. The motion in Cartesian space is also better when using the proposed predictors, with the controller inducing firstly a rotational motion that points the the sensor in the direction of the target, and subsequently a translational motion that completes

the positioning task.



(a) $\mathcal{M}_1$ (b) $\mathcal{M}_2$ (c) $\mathcal{M}_{\mathcal{H}}$

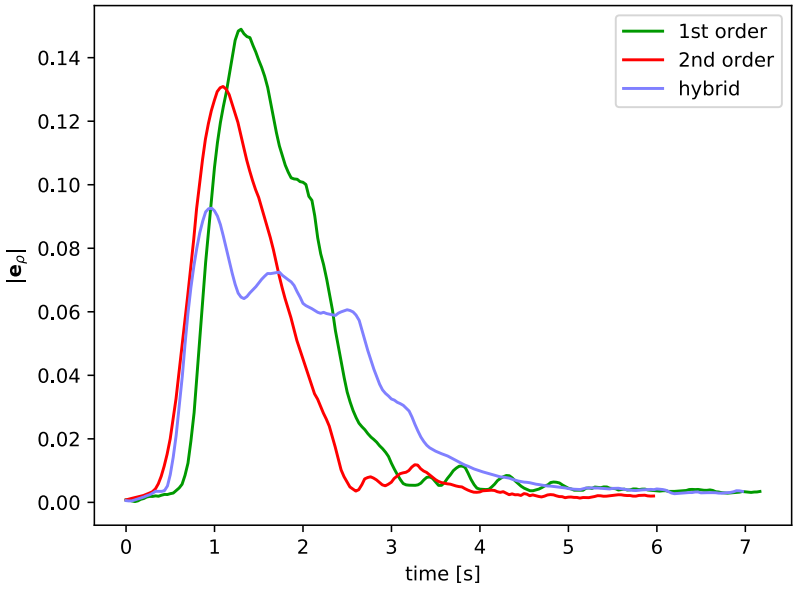
Figure 3.20 – Trajectory of the point centers in the image using respectively $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_{\mathcal{H}}$ when servoing from polar coordinates. Blue segments correspond to displacements induced by the object translation, while green ones are due to the control moving the camera.

In terms of features errors (*cfr.* Figure 3.21) it is worth noticing that all models present a less smooth evolution compared to the case of servoing from image coordinates. A possible explanation for this fact can be found in the optimization being harder to accomplish due to the rather high non-linearities corresponding to polar coordinates. Nonetheless, the benefits coming from considering the acceleration in the predictors are still evident, with $\mathcal{M}_2$ and $\mathcal{M}_{\mathcal{H}}$ being characterized by lower maximal errors with respect to $\mathcal{M}_1$.

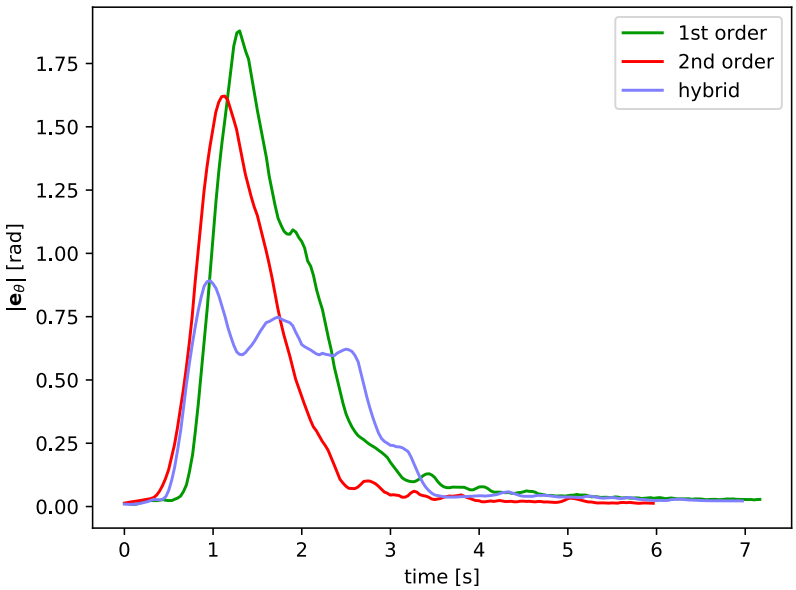
3.4 Conclusions

This chapter, and more specifically the last section, proposed two new models for Visual Predictive Control that lead to better robot motions thanks to the inclusion of features acceleration in the feature space. To support this claim, the performances of these models have been compared against another local predictor based solely on the first-order interaction matrix. By means of simulations, it was shown that convergence can be achieved in a short time while avoiding undesirable behaviors even in presence of large displacements, which is in line with the objectives of this thesis.

Moreover, the results were validated thanks to real experiments with an industrial



(a)



(b)

Figure 3.21 – Evolution of the feature error norms, while servoing from polar coordinates. In (a), the error associated to ρ, while in (b) the one associated to θ.

redundant robot, not only confirming the benefits of integrating acceleration information into the predictors, but also demonstrating the practical feasibility of the control schemes. In addition, by considering different features to describe the observed object, it was practically demonstrated that the approach is general and that its effectiveness is not bound to a specific parameterization of the observed object. The work was proposed to the scientific community in the form of an article which was peer-reviewed and accepted for publication [FKM20]. The article was also accepted for presentation in IROS 2020.

There is still room for future improvements in different forms. As an example, it would be interesting to test the proposed approach using image moments as features, a case that is more complex due to different reasons. First of all, the number of intermediate calculations required to compute the second-order models is rather large, *cfr.* Appendix A.5, since each image moment depends on a large parameter vector. A second reason is to be found in the structure of the interaction matrix: it depends on higher-order moments, theoretically leading to a parameter vector $\boldsymbol{z}$ of infinite dimension.

In addition, as mentioned in the last section, the controllers are not able to react immediately to object motions. This is most likely because such event is not foreseeable by the VPC schemes, as it is not modeled in the predictors. In order to achieve high speed performances in dynamic environments, it is believed that a key improvement would be to actively estimate the kinematics of the observed object [AA+06; Dah+08], and to take it into account explicitly inside the prediction models.

A further development might consider time-varying features references, rather than fixed configurations as it was done in simulations and experiments. This will likely require to adapt the formulation of the optimization problem, since the objective function currently contains a contribution that tries to minimize the joint velocity of the manipulator, which is in contrast with a task that requires the manipulator to continuously move. Remember that the second term is needed due to the redundancy, and it would not be sufficient to simply remove it. A possible solution would be to formulate the problem as a hierarchical, multi-objective, optimization in which the visual objective has the highest priority.

Chapter 4

Parameterized Model Predictive Control

The previous chapter focused on the use of Model Predictive Control in Visual Servoing, since it provides means to complete the relative positioning task faster, while also ensuring the satisfaction of constraints. As explained, the optimization required by predictive schemes can be computationally demanding, and as a result the complexity of the problem should be kept as small as possible so as to obtain solutions quickly enough to keep up with the target control frequency of the robot. However, the optimization should be able to consider a long enough time interval for the prediction process. In this sense, the predictors proposed in the previous chapter allowed to use a larger sampling time in the prediction models, allowing to extend T_p without necessarily increase the complexity of the optimization in terms of decision variables.

Nonetheless, in order to further improve the results obtained in the previous chapter, it could be necessary to extend the prediction horizon even more, or to slightly reduce the time sampling. As previously discussed, the optimization currently needs to be halted before the real minimum is attained, thus allowing to reach only a neighborhood of the actual optimum. As the control and prediction horizons are gradually increased, the optimization requires even more time to find solutions, meaning that, given the same amount of optimization time, the approximated minima will lie farther away from the real ones, ultimately leading in instabilities. Increasing only the prediction horizon clearly is a first way to limit a growth in complexity, but there are limitations as well. Indeed, as shown in the simulations of Section 3.3.4, when n_c is small compared to n_p the performances can deteriorate.

It seems that a stale has been reached: once the prediction horizon has been increased, we cannot increase n_c since it would prevent real-time feasibility, but at the same time the limited control horizon leads to sub-optimal performances. The stale is due to the fact that, according to the MPC formulation presented so far, incrementing the control horizon

directly implies a growth in the number of decision variables used in the optimization. However, what if a new, different, decision vector of equal dimension was used? Depending on what new variables are selected, it could be theoretically possible to change the result of the optimization – hopefully improving it – without significantly alter the computational load. This is precisely the subject of the present chapter, in which the concept of *control parameterizations* is studied. Rather than using the values of the first n_c input samples as optimization variables, the control sequence is synthesized from a set of parameters. Some of these can be considered as design constants and are selected in advance, while others are to be tuned online by the optimization algorithm. Since this concept is rather new in the robotics field, this chapter will focus on presenting the topic and to test its performances on simple, academic examples. In particular, the next section introduces the general concept of control parameterization and then focuses on some examples of linear parameterizations. Section 4.2 addresses the case of MPC with linear parameterizations for the control of linear systems, in order to show the benefits of this new approach in a simpler setup. Finally, the more challenging case of parameterized nonlinear MPC is presented in Section 4.3, which includes simulation results obtained with both simple nonlinear model – an inverted crane-pendulum – but also with a free-flying camera performing a visual tracking task.

4.1 Control Parameterizations

The idea behind Parameterized Model Predictive Control [Ala06; Ala13] is to use, as decision variables for the problem, not the control inputs directly, but rather a smaller set of parameters which can define the entire control sequence. In other words, control samples are obtained as functions of the set of parameters $\boldsymbol{p} \in \mathbb{R}^{n_\pi}$, *i.e.*,

$$\boldsymbol{u}_k = \boldsymbol{\pi}_k(\boldsymbol{p}) \qquad \boldsymbol{\pi}_k : \mathbb{R}^{n_\pi} \to \mathbb{R}^m \tag{4.1}$$

The goal is thus to find a suitable parameterization that allows the controls to vary along the whole prediction horizon without having to increase the complexity of the problem in terms of number of free variables.

One sub-family of parameterizations that can be considered is the one of linear parameterizations, according to which control input samples are expressed as a linear combination

of the parameters $\boldsymbol{p}$:

$$\boldsymbol{u}_k = \boldsymbol{\Pi}_k \boldsymbol{p} \tag{4.2}$$

for a given set of matrices $\boldsymbol{\Pi}_k \in \mathbb{R}^{m \times n_\pi}$. An equivalent relation can be written to map the parameter vector directly into the whole control sequence, simply by stacking the individual combination matrices $\boldsymbol{\Pi}_k$:

$$\underline{\boldsymbol{u}} = \boldsymbol{\Pi}\, \boldsymbol{p} \qquad \boldsymbol{\Pi} \doteq \begin{bmatrix} \boldsymbol{\Pi}_0 \\ \cdots \\ \boldsymbol{\Pi}_{n_p-1} \end{bmatrix} \tag{4.3}$$

It is interesting to note that the classical way of selecting the input samples in Model Predictive Control can be viewed as an example of linear parameterization. In particular, with a control and prediction horizons of n_c and n_p respectively, the combination matrix takes the following block form:

$$\boldsymbol{\Pi} = \begin{bmatrix} \mathbf{I} & & \\ & \ddots & \\ & & \mathbf{I} \\ & & \mathbf{I} \\ & & \vdots \\ & & \mathbf{I} \end{bmatrix} \begin{matrix} \Big\} n_c \\ \\ \\ \Big\} n_p - n_c \\ \\ \end{matrix} \tag{4.4}$$

wherein each block is of size m-by-m. In practice, the entries of $\boldsymbol{p}$ correspond to the free input samples, with the last m values being repeated as many times as needed to complete the control sequence.

One issue with this "simple parameterization" [1] is that almost all degrees of freedom are located at the beginning of the control sequence. The consequence is that, if the prediction horizon is extended, the last control sample will be used for a significant amount of time. This is not desirable, especially when considering unstable systems. In this sense, an example is given by the first two sets of simulations presented in the previous chapter: when considering $\mathcal{M}_2$, a single acceleration sample was not sufficient to stabilize the system.

Parameterizations can thus be used to avoid the aforementioned issue, in particular

1. The name *simple* will be used throughout the chapter to label the classical approach, since it is arguably the simplest conceivable parameterization.

by distributing the degrees of freedom along the prediction horizon. In particular, in the following sections three types of linear parameterizations are introduced, which can be viewed as successive generalizations of the classical approach.

4.1.1 Zero-Order-Holder

This first parameterization is the most similar to the simple one. However, rather than keeping all free control samples at the beginning of the sequence, the degrees of freedom are distributed along the prediction horizon. Since the concept is better illustrated with an example, Figure 4.1 depicts a possible control sequence produced by this parameterization.

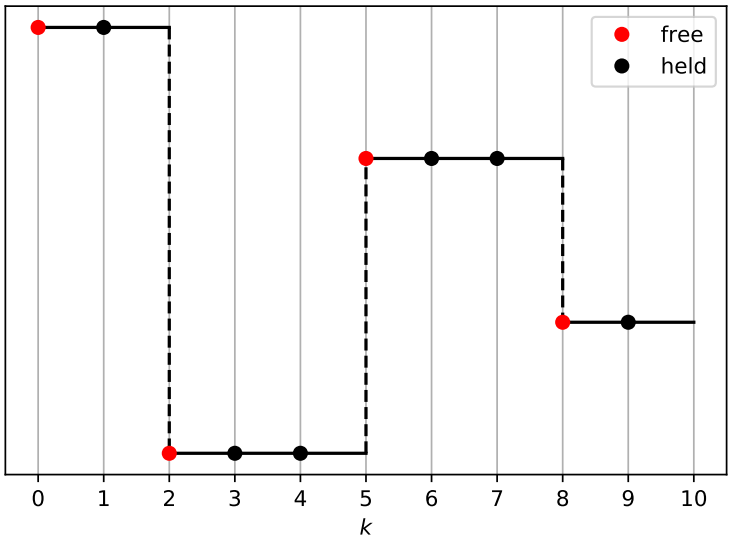


Figure 4.1 – Example of sequence that could be produced by a ZOH parameterization. Here, free samples are located at $k = 0, 2, 5, 8$, thus leading to a parameter vector of dimension $n_\pi = 4$.

In practice, some "free indices" are selected within the available samples, with the value of the corresponding control inputs being added to the set of parameters $\boldsymbol{p}$. The remaining, non-free, control values are evaluated simply by "holding" the same value of their nearest preceding free sample. For this reason, the parameterization is named in the sequel Zero-Order-Holder (ZOH).

The combination matrix $\boldsymbol{\Pi}$ writes in this case in a block-diagonal form that is similar

to the following:

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{I} & & & \\ \mathbf{I} & & & \\ & \mathbf{I} & & \\ & \mathbf{I} & & \\ & \mathbf{I} & & \\ & & \ddots & \\ & & & \mathbf{I} \\ & & & \mathbf{I} \end{bmatrix} \tag{4.5}$$

in which each sub-block consists in the vertical concatenation of as many identity matrices as the times the corresponding parameters are held to produce control samples.

4.1.2 Linear Interpolation

One drawback of the ZOH parameterization presented just now is that, by construction, it alternates between sudden changes of the control input and flat regions in which the value remains constant. It could be instead interesting to gradually change the values from one free sample to another. This would still allow to distribute the degrees of freedom along the control sequence, but also to obtain a smoother signal. One simple, intuitive, way to achieve this goal is to generate intermediate values by linear interpolation from a free sample to the following one. Once again, this new parameterization can be written as a linear combination with a block matrix that is similar to the following:

$$\mathbf{\Pi} = \begin{bmatrix} \mathbf{I} & & & & \\ \frac{1}{2}\mathbf{I} & \frac{1}{2}\mathbf{I} & & & \\ & \mathbf{I} & & & \\ & \frac{2}{3}\mathbf{I} & \frac{1}{3}\mathbf{I} & & \\ & \frac{1}{3}\mathbf{I} & \frac{2}{3}\mathbf{I} & & \\ & & \mathbf{I} & & \\ & & & \ddots & \\ & & & & \mathbf{I} \\ & & & & \mathbf{I} \end{bmatrix} \tag{4.6}$$

The effect of this matrix is better visualized in Figure 4.2, which shows that the control

sequence is obtained by firstly selecting the free samples (in red) and then by performing interpolation in between to obtain the missing intermediate samples – with the exception of the trailing input values, which are eventually taken as a repetition of the last free sample.

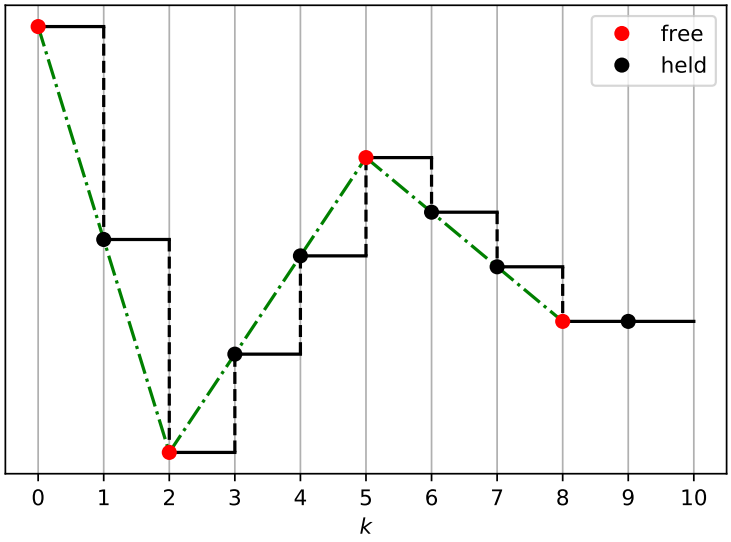


Figure 4.2 – Example of sequence that could be produced by a LERP parameterization. Here, free samples are located at $k = 0, 2, 5, 8$, thus leading to a parameter vector of dimension $n_\pi = 4$.

This parameterization, whose name will be shortened to LERP, has an additional benefit with respect to the simple and ZOH ones. Consider adding constraints on the control inputs in the following form:

$$-\Delta \boldsymbol{u}_{lim} \leq \boldsymbol{u}_k - \boldsymbol{u}_{k-1} \leq \Delta \boldsymbol{u}_{lim} \tag{4.7}$$

which simply sets a limit on how much successive input samples can differ in a control sequence. Given a certain value of $\boldsymbol{u}_{-1}$, *i.e.*, the last control sample used to actuate the system, the constraint reduces the range of achievable control inputs, since free values are no more independent from each other. Given that the simple and ZOH parameterizations allw control changes only in a limited number of locations, the range of achievable inputs is thus reduced substantially, as shown in Figure 4.3. Conversely, the LERP parameterization allows to reach a broader range of inputs, since the control sequence is allowed to change along the whole prediction horizon.

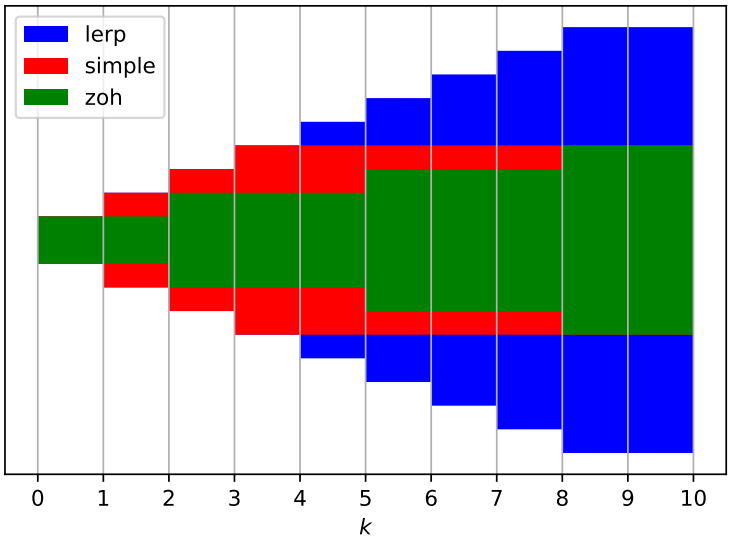


Figure 4.3 – Sets of achievable controls using the simple (red), ZOH (green) and LERP (blue) parameterizations, under the variations constraints (4.7).

4.1.3 Basis Functions

In this section, rather than a single, specific, parameterization, a whole family of parameterizations is introduced, which relies on the concept of function spaces and function bases. It is well known that certain classes of functions can be generated by the sum of some (possibly infinitely many) functions. As an example, polynomials can be summed together to generate any analytical function according to its Taylor expansion. Similarly, sinusoidal signals with harmonic frequencies can be combined to form any periodic function as stated by Fourier's series expansion.

Taking inspiration from this idea, it is possible to consider parameterizations that combine a certain number of functions to generate at once the whole control sequence, which can be viewed as an extension of the proposal by Alamir to parameterize the control sequence with a decaying exponential profile [Ala13]. In practice, it is possible to select a certain number n_b of basis functions $\mathcal{B}_i : \mathbb{R} \to \mathbb{R}$ $(i = 1, \cdots, n_b)$ and, considering for simplicity the case of a scalar-valued control input, combine them together to produce a continuous-time signal:

$$u(t) = \sum_{i=1}^{n_b} h_i \, \mathcal{B}_i(t) \tag{4.8}$$

wherein the scalars h_i represent combination coefficients. Intuitively, the use of a finite

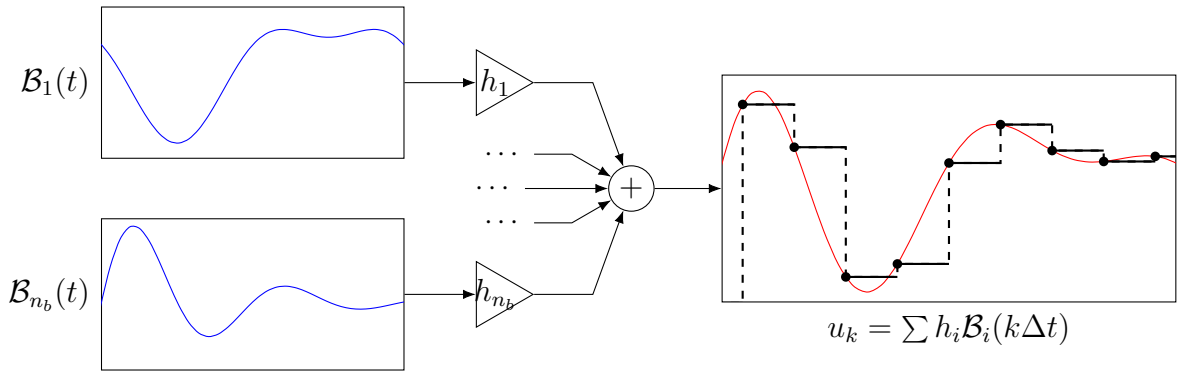


Figure 4.4 – Pictorial example showing how a control sequence is produced from a set of basis functions with given combination coefficients.

number of functions will generally limit the shapes that can be produced as output. However, it turns out that in practice even a reduced number of functions is sufficient to define a sufficiently rich variety of signals that can be effectively used in predictive control schemes.

Starting from (4.8), in order to produce the corresponding discrete-time control sequence it suffices to sample the generated function with proper time spacing Δt:

$$u_k = \sum_{i=1}^{n_b} h_i \, \mathcal{B}_i(k\Delta t) \tag{4.9}$$

Note that by adjusting the coefficients h_i, it is possible to change the shape of the resulting signal, which naturally leads to the idea of using $h_1, \cdots, h_{n_b}$ as parameters. In other words, rather than acting on individual samples as all previous parameterizations did, the parameterization now acts on the components (the basis function) that constitute the control signal. In addition, similarly to the case of the LERP parameterization, control sequences obtained with these parameterizations tend to change more smoothly from sample to sample if the basis functions are smooth themselves.

Interestingly, if the shape of each basis function $\mathcal{B}_i$ is fixed in advance, then the parameterization is linear in the coefficients h_i. In fact, the control sequence vector can be written as:

$$\underline{\mathbf{u}} = \begin{bmatrix} u_0 \\ \vdots \\ u_{n_p-1} \end{bmatrix} = \begin{bmatrix} \mathcal{B}_1(0) & \cdots & \mathcal{B}_{n_b}(0) \\ \mathcal{B}_1(\Delta t) & \cdots & \mathcal{B}_{n_b}(\Delta t) \\ \vdots & \ddots & \vdots \\ \mathcal{B}_1\left((n_p-1)\Delta t\right) & \cdots & \mathcal{B}_{n_b}\left((n_p-1)\Delta t\right) \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_{n_b} \end{bmatrix} \tag{4.10}$$

showing that this indeed constitutes a linear parameterization, with the free parameters $\mathbf{p}$ corresponding to the collection of all coefficients h_i. Finally, even if only the case of scalar-valued controls was discussed here, the extension to a multidimensional input is trivial.

Coming now to the problem of choosing a proper set of basis functions, several options can be considered. One possibility would be to spread along the prediction horizon Gaussians in the form

$$\mathcal{B}_i(t) = e^{-(t-t_i)^2/\sigma_i^2} \tag{4.11}$$

or any other kind of radial function. These are often used to interpolate unknown functions from a given set of points, and are known for providing quite good approximations for a large number of cases. In other words, when combined altogether they can produce a large number of shapes, which is exactly what is required for the proposed parameterization. Another example could be that of sinusoidal functions which, according to Fourier's expansion, can be viewed as the building blocks of any function within a given time window. A third option considered in the following is the one of exponentially damped Laguerre polynomials [MD16], which take the following form:

$$\mathcal{B}_i(t) = e^{-\varepsilon t} \sum_{j=0}^{i} \binom{i}{j} \frac{(-2\varepsilon t)^j}{j!} \tag{4.12}$$

Before concluding this section, let us remark that all "internal" parameters appearing in the selected basis functions must be set prior to the optimization for the resulting parameterizations to be linear. As an example, the scaling factor ε in (4.12) or the variances and centers t_i and σ_i in (4.11) need to be fixed in advance using either some knowledge on the problem or by trial and error. This is perhaps the biggest drawback of these parameterizations, as they introduce an additional user-based tuning which might affect the final performances. One possibility to remove this additional step would consist in considering these values not as constants to be set by the user, but within the set of parameters that the optimization algorithm can use. This would lead to loss of linearity and likely higher computation times, but almost surely enhancing the performances since the tuning of the internal basis parameters becomes automatic and not delegated to the user.

4.2 Linearly Parameterized Linear Model Predictive Control

In this section, the simpler case of MPC for linear systems with linear parameterizations is considered, which is still of high interest for robotics applications despite its simplicity. In fact, even when dealing with generic systems, a simple tangent linearization can be exploited to obtain an approximate linear systems, if the nonlinearities are mild. Furthermore, some nonlinear systems, such as differentially flat ones, can be transformed in equivalent linear models and a MPC scheme can be designed to control the new system without having to deal with the nonlinear dynamics in the optimization problem [ACT07; De +16]. As it will be shown, the specific structure of linear models allows to formulate the optimization problem as a linearly constrained least squares minimization problem, which can be solved using any quadratic programming approach.

Given that both formulating and solving this class of problems does not pose major challenges, linearly parameterized linear MPC schemes are perfect candidates to benchmark the proposed parameterizations and gather insightful preliminary results. With this objective in mind, the following sections thus focus firstly on stating the general optimization problem and secondly on analyzing the performances with and without parameterizations in the case of two specific linear systems: a triple integrator and an under-damped oscillator.

4.2.1 Problem Formulation

We consider in the following the class of discrete-time linear-affine systems, written in state-space form as:

$$\boldsymbol{x}_{k+1} = \mathbf{A}\boldsymbol{x}_k + \mathbf{B}\boldsymbol{u}_k + \boldsymbol{w} \tag{4.13}$$

where $\mathbf{A}$ and $\mathbf{B}$ are matrices of appropriate dimension and $\boldsymbol{w} \in \mathbb{R}^n$. These quantities are assumed to be invariant with respect to time, but a generalization of the following results is trivial.

The goal is to show that the optimization problem that arises from a MPC scheme applied to this class of systems is a quadratic program. For this purpose, the first step is to show that the prediction vector $\underline{\boldsymbol{x}}$ can be expressed concisely as a linear combination of the initial state $\boldsymbol{x}_0$ and of the inputs. The first terms of the prediction sequence write

as:

$$\boldsymbol{x}_1 = \mathbf{A}\boldsymbol{x}_0 + \mathbf{B}\boldsymbol{u}_0 + \boldsymbol{w} = \mathbf{A}\boldsymbol{x}_0 \qquad +\boldsymbol{w} \quad +\mathbf{B}\boldsymbol{u}_0 \tag{4.14a}$$

$$\boldsymbol{x}_2 = \mathbf{A}\boldsymbol{x}_0 + \mathbf{B}\boldsymbol{u}_1 + \boldsymbol{w} = \mathbf{A}^2\boldsymbol{x}_0 \qquad +(\mathbf{A}+\mathbf{I})\boldsymbol{w} \; +\mathbf{A}\mathbf{B}\boldsymbol{u}_0 \quad +\mathbf{B}\boldsymbol{u}_1 \tag{4.14b}$$

$$\boldsymbol{x}_3 = \mathbf{A}\boldsymbol{x}_2 + \mathbf{B}\boldsymbol{u}_2 + \boldsymbol{w} = \mathbf{A}^3\boldsymbol{x}_0 + (\mathbf{A}^2+\mathbf{A}+\mathbf{I})\boldsymbol{w} + \mathbf{A}^2\mathbf{B}\boldsymbol{u}_0 + \mathbf{A}\mathbf{B}\boldsymbol{u}_1 + \mathbf{B}\boldsymbol{u}_2 \tag{4.14c}$$

from which a pattern clearly emerges, with the general formula being:

$$\boldsymbol{x}_{k+1} = \mathbf{A}^{k+1}\boldsymbol{x}_0 + \left(\sum_{i=0}^{k}\mathbf{A}^i\right)\boldsymbol{w} + \sum_{i=0}^{k}\mathbf{A}^{k-i}\mathbf{B}\boldsymbol{u}_i \tag{4.15}$$

which precisely means that predicted samples are a linear combination of the initial state $\boldsymbol{x}_0$ and of the control inputs (plus a constant term that depends on the affine term $\boldsymbol{w}$). Moreover, the whole prediction vector can be written in the following linear-affine form:

$$\underline{\boldsymbol{x}} = \boldsymbol{\eta} + \tilde{\mathbf{B}}\underline{\boldsymbol{u}} \tag{4.16}$$

wherein $\boldsymbol{\eta}$ and $\tilde{\mathbf{B}}$ are defined as

$$\boldsymbol{\eta} \doteq \begin{bmatrix}\mathbf{A}\\ \mathbf{A}^2\\ \mathbf{A}^3\\ \cdots\end{bmatrix}\boldsymbol{x}_0 + \begin{bmatrix}\mathbf{I}\\ \mathbf{A}+\mathbf{I}\\ \mathbf{A}^2+\mathbf{A}+\mathbf{I}\\ \cdots\end{bmatrix}\boldsymbol{w} \tag{4.17a}$$

$$\tilde{\mathbf{B}} \doteq \begin{bmatrix}\mathbf{B} & & & \\ \mathbf{A}\mathbf{B} & \mathbf{B} & & \\ \mathbf{A}^2\mathbf{B} & \mathbf{A}\mathbf{B} & \mathbf{B} & \\ \vdots & \vdots & \vdots & \ddots\end{bmatrix} \tag{4.17b}$$

We can finally introduce a linear input parameterization in the form of (4.3), leading to:

$$\underline{\boldsymbol{x}} = \boldsymbol{\eta} + \tilde{\mathbf{B}}\boldsymbol{\Pi}\boldsymbol{p} \tag{4.18}$$

The objective is defined by the quadratic cost (3.9), while linear constraints in the

control inputs and predicted states are introduced in the form:

$$\mathbf{C}_u \underline{\boldsymbol{u}} \leq \boldsymbol{d}_u \tag{4.19a}$$

$$\mathbf{C}_x \underline{\boldsymbol{x}} \leq \boldsymbol{d}_x \tag{4.19b}$$

which, among others, allow to set limits both on the magnitude and on the variation of the controls and states. After some trivial calculations, the overall problem can be finally stated as:

$$\min_{\boldsymbol{p}} \frac{1}{2} \boldsymbol{p}^T \mathbf{H} \boldsymbol{p} + \boldsymbol{f}^T \boldsymbol{p} \tag{4.20a}$$

subject to:

$$\mathbf{C}_p \boldsymbol{p} \leq \boldsymbol{d}_p \tag{4.20b}$$

wherein the different matrices and vectors are defined as follows:

$$\mathbf{D}_x \doteq \operatorname{diag}(\mathbf{Q}_k) \tag{4.21a}$$

$$\mathbf{D}_u \doteq \operatorname{diag}(\mathbf{R}_k) \tag{4.21b}$$

$$\mathbf{H} \doteq 2\boldsymbol{\Pi}^T \left(\tilde{\mathbf{B}}^T \mathbf{D}_x \tilde{\mathbf{B}} + \mathbf{D}_u \right) \boldsymbol{\Pi} \tag{4.21c}$$

$$\boldsymbol{f} \doteq 2\boldsymbol{\Pi}^T \left(\tilde{\mathbf{B}}^T \mathbf{D}_x \left(\boldsymbol{\eta} - \underline{\boldsymbol{x}}^\star \right) - \mathbf{D}_u \underline{\boldsymbol{u}}^\star \right) \tag{4.21d}$$

$$\mathbf{C}_p \doteq \begin{bmatrix} \mathbf{C}_u \\ \mathbf{C}_x \tilde{\mathbf{B}} \end{bmatrix} \boldsymbol{\Pi} \tag{4.21e}$$

$$\boldsymbol{d}_p \doteq \begin{bmatrix} \boldsymbol{d}_u \\ \boldsymbol{d}_x - \mathbf{C}_x \boldsymbol{\eta} \end{bmatrix} \tag{4.21f}$$

The problem thus corresponds to a linearly constrained quadratic minimization, as stated in the beginning of this section, which can be solved using, *e.g.*, the active set method described in Appendix B.2. Note that if the constraints are feasible, the problem is convex and always admits a solution, since $\mathbf{H}$ is at least positive semidefinite.

4.2.2 Control of a Triple Integrator

In this section, a first example of linearly parameterized MPC is presented, in which the controlled system is a simple triple integrator. The continuous-time dynamics is discretized assuming piecewise constant inputs due to a zero-order-holder device with sampling period $\Delta t = 0.1\,\mathrm{s}$. The model in state-space representation is given by the following:

$$\boldsymbol{x}_{k+1} = \begin{bmatrix} 1 & 0.1 & 0.005 \\ 0 & 1 & 0.1 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{x}_k + \begin{bmatrix} 0.000167 \\ 0.005 \\ 0.1 \end{bmatrix} \boldsymbol{u}_k \tag{4.22}$$

The objective of the control scheme is to track a variable reference using the first coordinate of the state vector, or, with a little abuse of terminology, "position". This reference is initially constant and equal to 0 for the first 2.5 s of the simulation. Afterwards, it is changed according to a sinusoidal profile with unitary amplitude and $1.25\,\mathrm{s}^{-1}$ pulsation ($\simeq 0.2\,\mathrm{Hz}$ frequency).

Symmetric bounds on the scalar input are set so that it lies in the interval $[-5, 5]$, and in addition constraints in the form of (4.7) are added, with maximum variation set to $\Delta \boldsymbol{u}_{lim} = 1$.

Finally, the prediction horizon is set to $n_p = 20$, allowing to predict over a period of time of $T_p = 2\,\mathrm{s}$, and four types of parameterization are considered: the simple one from classical MPC theory, Zero-Order-Holder, Linear Interpolation and one based on the damped Laguerre polynomials (4.12) with $\varepsilon = 0.75$. These parameterizations are compared in the sequel using a different number of parameters. More precisely, three sets of simulations are proposed, with n_π equal to 6, 3 and 2 respectively.

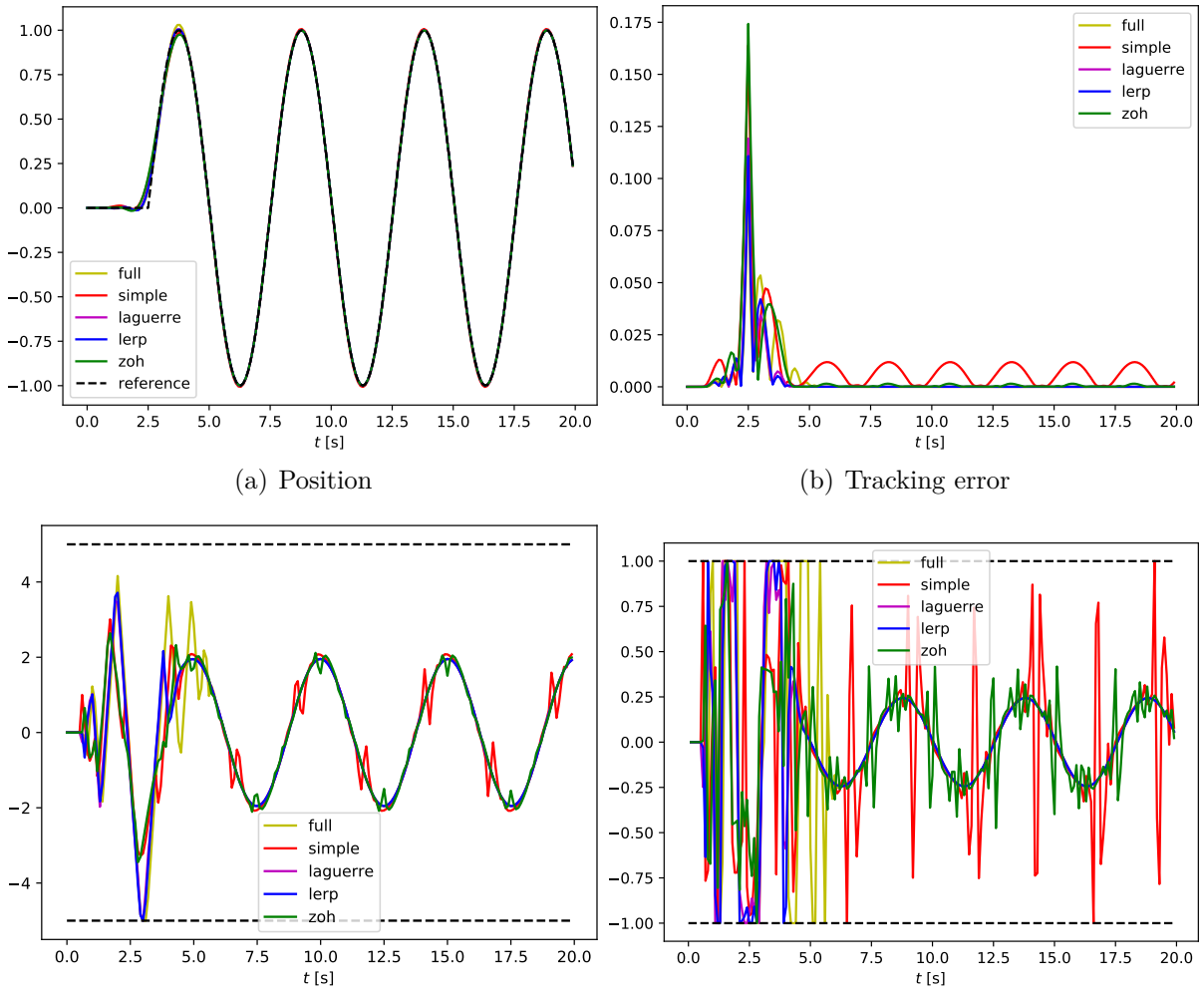
First test: 6 parameters

In this first test, the decision vector in the optimization has size $n_\pi = 6$. Since the control input is a scalar signal, this means that the control horizon n_c is set to the same value for the simple parameterization, while ZOH and LERP parameterizations feature 6 free-samples each. In particular, these samples were set to $k_{free} = 0, 1, 3, 5, 10, 15$ for the holder parameterization and to $k_{free} = 0, 2, 5, 10, 15, 19$ when using LERP. Finally, a total of $n_b = 6$ Laguerre's functions have been used.

A simulation was performed for each parameterization, recording the position of the system as well as the tracking error, which are shown in Figures 4.5(a) and 4.5(b) respec-

tively. In addition, the control signal used to command the system and its variation are depicted with their respective bounds in Figures 4.5(c) and 4.5(d).

Note that, in addition to the data associated to the four parameterizations detailed above, Figure 4.5 includes the results obtained with a classical scheme characterized by $n_\pi = n_c = n_p = 20$. Since it features the highest number of degrees of freedom, it is named "full" parameterization for brevity. It provides the theoretically best case for the MPC scheme, and is thus meant to serve as a reference to better interpret simulations results.



(a) Position

(b) Tracking error

(c) Control input

(d) Control variation

Figure 4.5 – Control of a triple integrator using a simple (red), ZOH (green), LERP (blue) and Laguerre (magenta) parameterizations, all having $n_\pi = 6$ parameters. An additional case is reported for comparison, consisting in a full parameterization with $n_\pi = n_p$.

It is interesting to notice that parameterizations positively influence the evolution

of the system. Focusing on the tracking error depicted in Figure 4.5(b) in the different cases, the three proposed parameterizations exhibit, after the initial transient phase, a lower error compared to the classical case. Furthermore, an interesting result is related to the control inputs. When the reference changes from a constant value to a sinusoidal profile, all optimized controls tend to vary significantly, becoming smoother only after this transient phase. Nonetheless, the classical strategy features few spots in which the control inputs experience rather large and abrupt changes even after the transient. The ZOH parameterization suffers from a similar problem, but with more modest oscillations in the control signal. Finally, LERP and Laguerre parameterizations nicely adapt to the output and produce smooth and regular control inputs, apparently reaching steady-behavior even before the full case.

Second test: 3 parameters

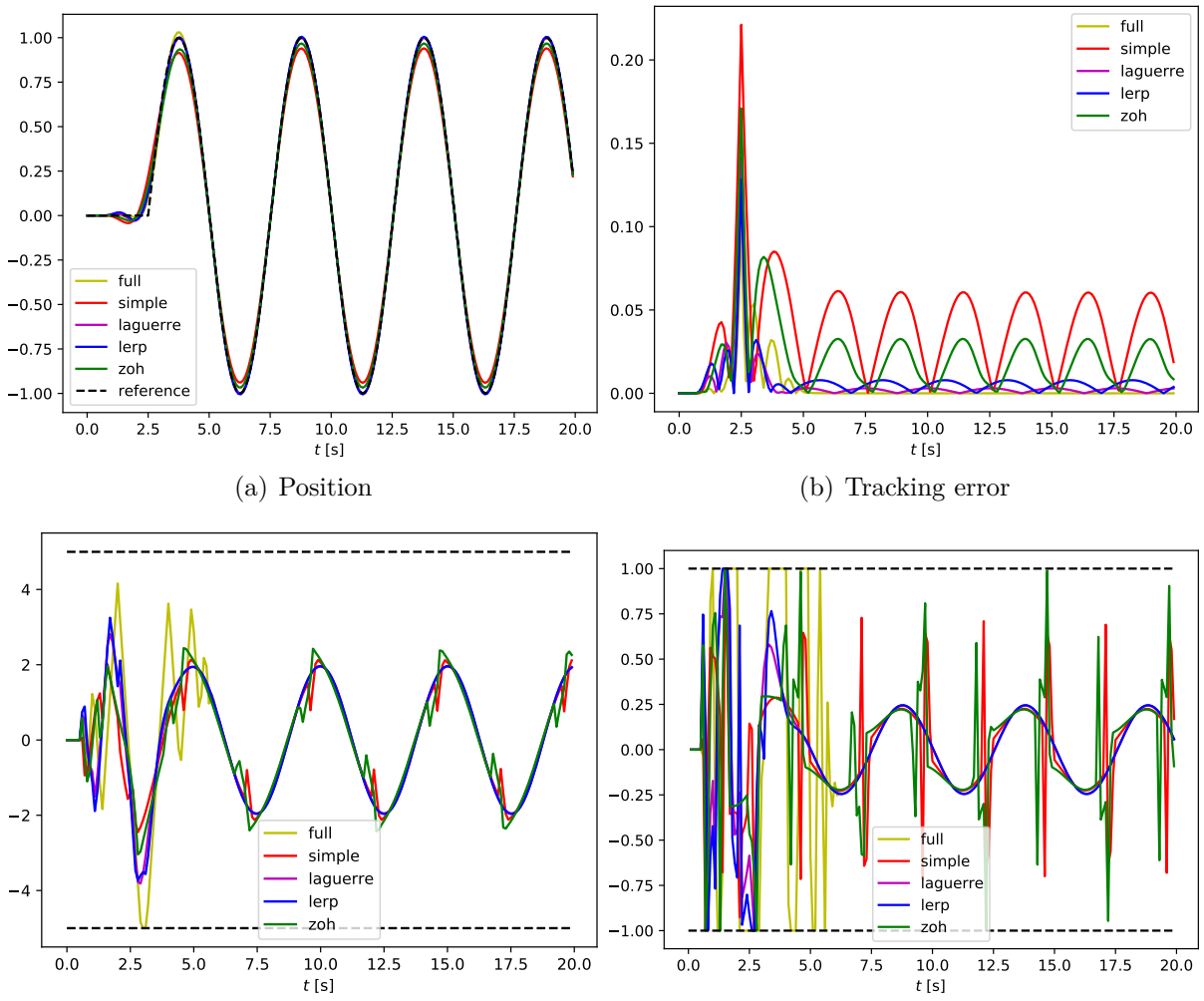
The simulations proposed above were all repeated, reducing the degrees of freedom to just 3. The free samples were changed to $k_{free} = 0, 2, 5$ and $k_{free} = 0, 4, 8$ for the ZOH and LERP parameterizations respectively, while the coefficient ε was left unchanged for the Laguerre parameterization.

Having halved the total number of parameters with respect to the previous simulations, clearer differences appear (see Figure 4.6). In particular, the classical approach provides the least satisfactory performances, with peak errors that are almost twice as large as those obtained with a holder parameterization. LERP and Laguerre parameterizations perform rather nicely, with reduced errors and smoother control inputs, even though differences with respect to the full parameterization are more evident.

Third test: 2 parameters

This third and last test considers as few as 2 degrees of freedom for the optimization, with the free-samples for the holder and LERP parameterizations set to $k_{free} = 0, 2$ and $k_{free} = 0, 10$.

The results are reported in Figure 4.7, in the same format used for previous simulations. The holder parameterization presents large tracking errors in the first half of the test, while in the end of the simulation it stabilizes closer to the reference. Other parameterizations required less time to settle to a steady behavior, with the best performances being those of LERP and Laguerre parameterizations.



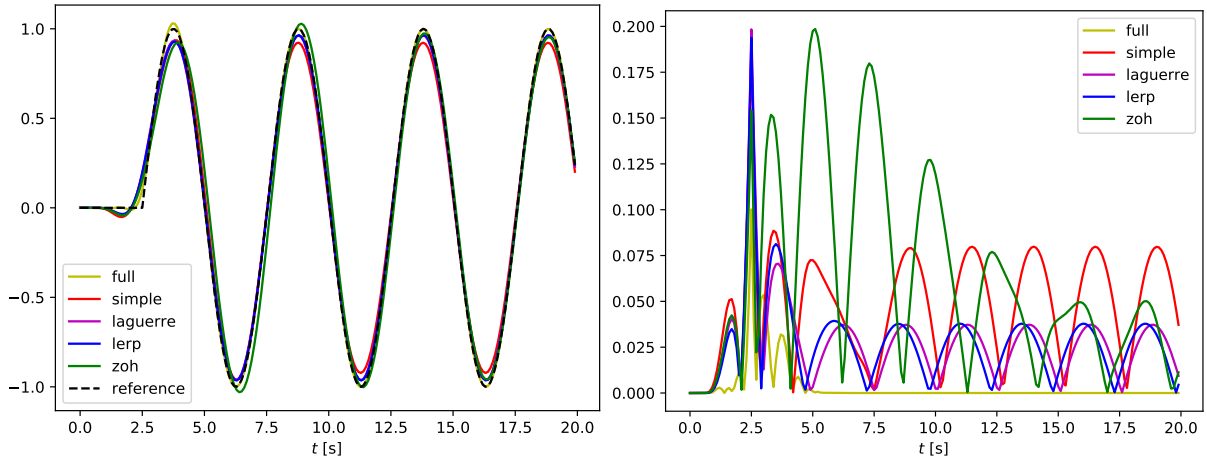
(a) Position

(b) Tracking error

(c) Control input

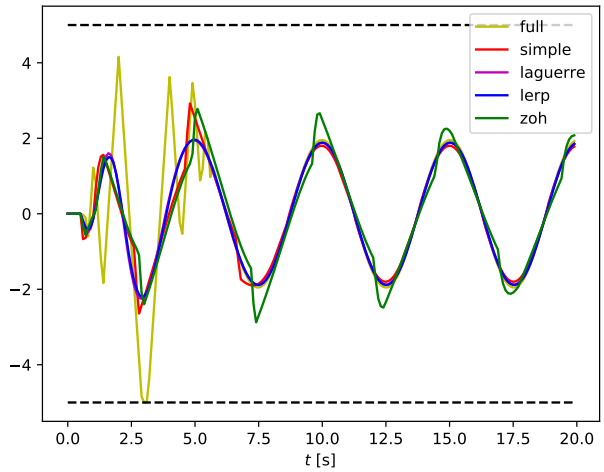
(d) Control variation

Figure 4.6 – Control of a triple integrator using a simple (red), ZOH (green), LERP (blue) and Laguerre (magenta) parameterizations, all having $n_\pi = 3$ parameters. An additional case is reported for comparison, consisting in a full parameterization with $n_\pi = n_p$.

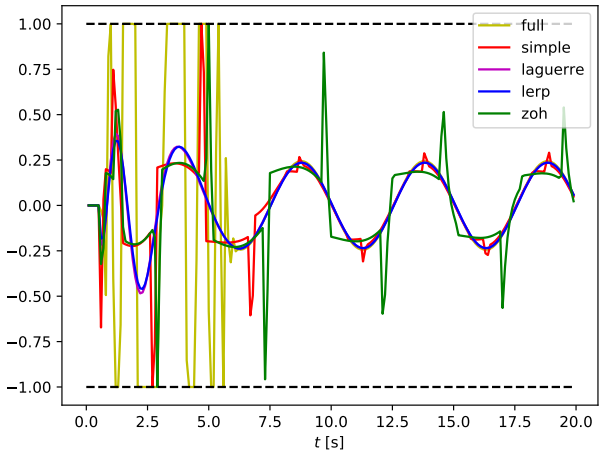


(a) Position

(b) Tracking error



(c) Control input



(d) Control variation

Figure 4.7 – Control of a triple integrator using a simple (red), ZOH (green), LERP (blue) and Laguerre (magenta) parameterizations, all having $n_\pi = 2$ parameters. An additional case is reported for comparison, consisting in a full parameterization with $n_\pi = n_p$.

Final Comparison

To conclude the analysis of the parameterizations used to control a triple integrator, a synthesis of the data collected from several simulations is now presented.

First of all, it must be noted that, for a given number of parameters, there is no reason to expect significant differences when solving the optimization problem (4.20) with different parameterizations, given that the matrices should be of equal size. However, this is just a heuristic estimation, and the times might fluctuate depending on which constraints are activated and deactivated to reach the optimum, as well as the current CPU load. For this reason, tests similar to those included in the previous sections were run multiple times to gather statistical information related to the time required to complete the optimization. The resulting data is included in Figure 4.8, which shows, as solid lines, the median optimization time of each parameterization in function of the number of parameters n_π. Note that the times are scaled relative to the median time obtained when using the full parameterization. In addition, semi-transparent regions are added to provide the maximum and minimum times for each case. The presented data shows that the optimization times obtained with each approach can indeed fluctuate, but within a relatively small range. The LERP parameterization seems to be the best performing one, while the Laguerre and simple parameterizations are the worst, the former when 5 or less parameters are used and the latter when $n_\pi \geq 6$. It is believed that one reason for the initial gap between the performances of the Laguerre parameterization with respect to the other strategies is to be found in the number of constraints. In fact, while control limits can be applied simply at the free samples to ensure that other values will remain within the given bounds, this is not necessarily the case when using the Basis function approach. This means that in this case the number of control constraints is proportional to n_p rather than n_π as it is with simple, ZOH and LERP parameterizations.

The three sets of simulations proposed before suggest that the LERP and Laguerre parameterizations lead to better results in terms of tracking performances, but it is hard to quantify how much the performances depend on the number of parameters. For this reason, average errors were calculated for different parameterizations with an increasing dimension of the optimization variables. The resulting curves were then normalized with respect to the average error obtained with the full parameterization, and are reported in Figure 4.9. It must be noted that the solution to the optimization problem is deterministic, and thus no statistical analysis was performed in this case.

The reported results support the claim that parameterizations bear improvements to

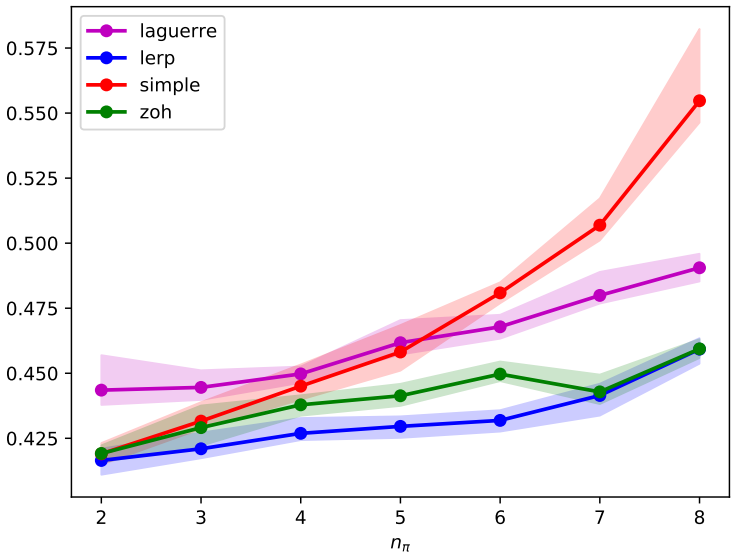


Figure 4.8 – Optimization times obtained when controlling a triple integrator using different parameterizations, as function of the number of parameters n_π. Solid lines represent median times, while semi-transparent regions are used to show maximum and minimum times. Provided values are relative to the median time required for the optimization using the full parameterization.

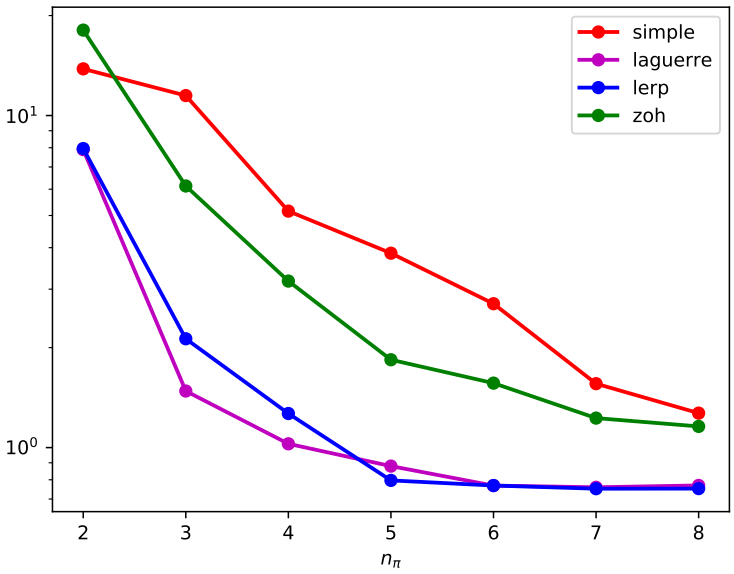


Figure 4.9 – Average tracking errors obtained when controlling a triple integrator using different parameterizations, as function of the number of parameters n_π. Provided values are relative to the average error obtained when using the full parameterization.

MPC. In particular, LERP and Laguerre parameterizations provide the best performances, with an average error that is comparable to the one of the full parameterization already with as few as 5 parameters. The Zero-Order-Holder parameterization does not provide results that are as nice as those of the other two approaches, but nonetheless outperforms the simple parameterization almost everywhere.

4.2.3 Control of a Mass-Spring-Damper

A linear system that often appears in robotics applications is the mass-spring-damper model, which can be described by the following ordinary differential equation:

$$m\ddot{x} = -2m\xi\omega_n\dot{x} - m\omega_n^2 x + u \tag{4.23}$$

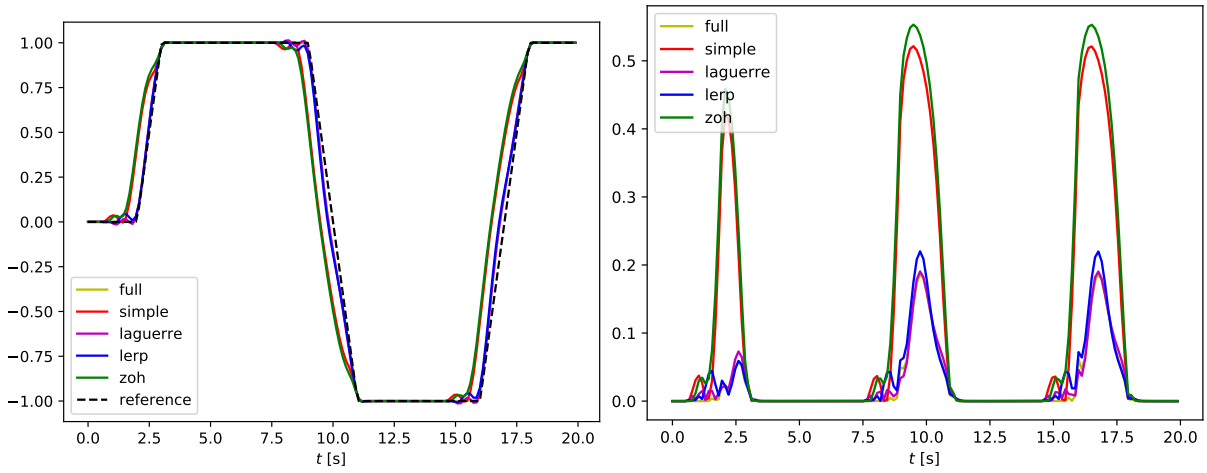
wherein m, ω_n and ξ are respectively the mass, the natural frequency and the damping coefficient of the system. The variable x represents the position of the mass, and the input u corresponds to a force applied to the body.

This section briefly analyzes the use of parameterized MPC to control such kind of system, with the objective of showing that the good performances obtained using the parameterized approach were not limited to the case of the triple integrator, but are indeed a characteristic of the proposed method. To simplify the test, the mass is selected as $m = 1\,\mathrm{kg}$, while the natural frequency is chosen as $\omega_n = 5\,\mathrm{s}^{-1}$. The damping coefficient is selected as $\xi = 0.3$, thus leading to an under-damped system. The model is converted in state-space representation and discretized, assuming a Zero-Order-Holder device for the input, via matrix exponentiation. The frequency is chosen as 100 Hz, while the sampling time used for predictions is set to $\Delta t = 0.05\,\mathrm{s}$. The prediction horizon is set to $n_p = 30$, thus allowing to foresee within a time interval of $T_p = 1.5\,\mathrm{s}$. Control inputs are bounded in the interval $[-30\,\mathrm{N}, 30\,\mathrm{N}]$, with a maximum variation $\Delta\boldsymbol{u}_{lim} = 1\,\mathrm{N}$.

The considered parameterizations are again ZOH, LERP and Laguerre ones, in addition to the classical MPC parameterization. The objective is to track a variable reference position, which corresponds to a "trapezoidal wave". It starts at level 0 m, and shortly after rises at a rate of $1\,\mathrm{m\,s}^{-1}$ until it has reached the value of 1 m. After 5 s, it starts decreasing, attaining the value -1 m in 2 s, and the process is repeated periodically.

First test: 6 parameters

In this first simulation, the optimization has to find the optimal control sequence using a total of 6 parameters. The tuning of the parameterizations for this test was almost the same as the one used in the first test shown with the triple integrator. The main difference is that the Laguerre parameterization here uses one damped polynomial less than before. Nonetheless, it features an additional static offset whose height is to be chosen by the optimizer, thus still using a total of 6 basis functions.

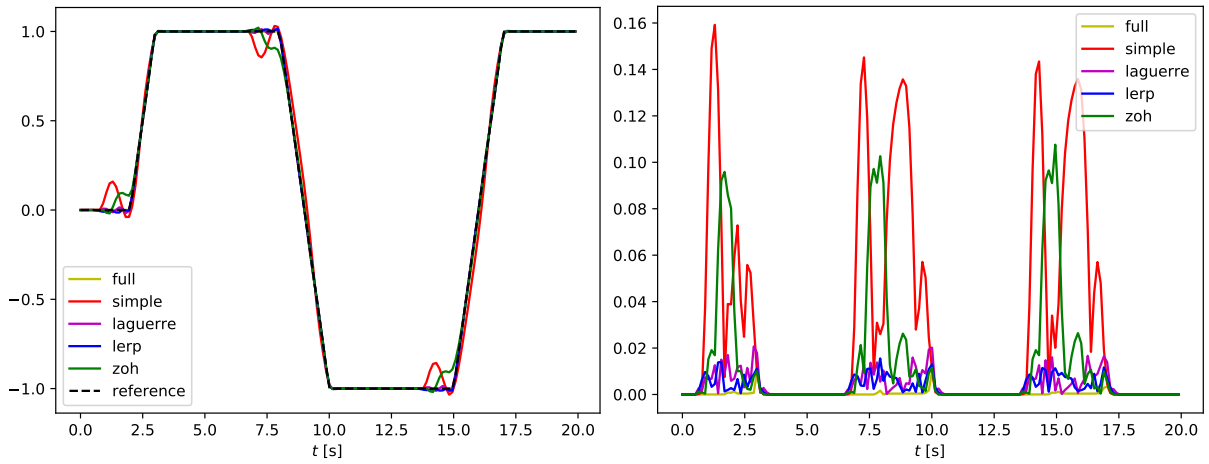


(a) Position (b) Tracking error

Figure 4.10 – Control of a mass-spring-damper system using a simple (red), ZOH (green), LERP (blue) and Laguerre (magenta) parameterizations with $n_\pi = 6$ parameters, as well as a full parameterization (yellow) with $n_\pi = n_p = 30$.

Results from the simulation are reported in Figure 4.10, which shows the position of the mass along time and the tracking error obtained with the different parameterizations. The simple and ZOH parameterization do not perform as desired, moving the mass too early with respect to the desired reference. This fact can be explained by the tight control variation limits, which, as briefly anticipated earlier in this chapter, can greatly reduce the range of achievable control values, especially when using simple and ZOH parameterizations. To support this claim, the same simulation has been run also with increased variation limits ($\Delta \boldsymbol{u}_{lim} = 5\,\mathrm{N}$). As visible in Figure 4.11, when these constraints are relaxed, all parameterizations provide better results in terms of convergence. In particular, the issue previously encountered with the simple and ZOH parameterizations is no longer present, and the mass is no longer moved before it is necessary. However, the relative per-

formances are more or less unaffected, with the LERP and Laguerre parameterizations that once again provide an evolution that is quite close to that of the full parameterization.



(a) Position (b) Tracking error

Figure 4.11 – Control of a mass-spring-damper using $n_\pi = 6$ parameters. The control variation limit has been increased to $\Delta \boldsymbol{u}_{lim} = 5\,\mathrm{N}$.

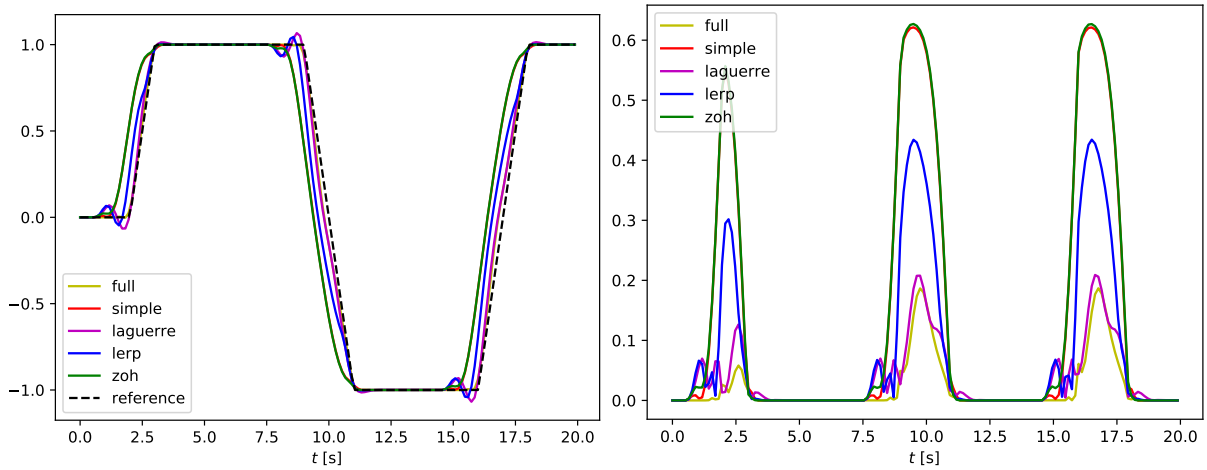
Second test: 3 parameters

A final simulation is now presented to conclude the tests on linear systems. In here, the variation limit of the control input is restored to its original value, $\Delta \boldsymbol{u}_{lim} = 1\,\mathrm{N}$. The parameters are reduced to $n_\pi = 3$, which is worth remarking is just a tenth of the samples used for the prediction.

Figure 4.12 shows the results obtained with the discussed setup. The LERP parameterization shows issues similar to those of simple and ZOH ones, forcing the mass to move in advance and thus resulting in peak errors that are twice as large as those obtained with the full parameterization. On the other hand, the parameterization based on basis functions seems almost unaffected by the decrease in the parameters number, with tracking errors that closely resemble those obtained with the full parameterization.

Final Comparison

The last two sets of simulations offered a glimpse at the potential benefits coming from the use of parameterizations that allow to cover broader sets of control samples along the prediction horizon, especially when the distance between subsequent values is



(a) Position (b) Tracking error

Figure 4.12 – Control of a mass-spring-damper system using a simple (red), ZOH (green), LERP (blue) and Laguerre (magenta) parameterizations with $n_\pi = 3$ parameters, as well as a full parameterization (yellow) with $n_\pi = n_p = 30$.

tightly constrained. In addition, so as to provide a wider analysis of the performances, Figure 4.13 includes for each parameterization the average error (normalized by the one obtained with the full parameterization) over the number of parameters used in the test.

Even at a first glance result are clear, with the Laguerre parameterization raking first and followed by the LERP one. Moreover, the gap between these two strategies and the classical one is much larger than in the case of the triple integrator. This is encouraging, especially considering that the number of parameters used here is rather limited, confirming the interest in further investigating the topic of Parameterized Model Predictive Control.

Another fundamental remark is related to the ZOH parameterization being always outperformed by the simple one. A possible cause for this could simply be that the holder parameterization might not be well suited for this case of scenario, which involves a long prediction horizon and small variation limits. This argument is supported by the evidence that, when constraints are loosen, this parameterization can perform much better (recall Figure 4.11). Moreover, this is the parameterization whose range of achievable inputs is the smallest, as it had been depicted in Figure 4.3. With these two facts in mind, the conclusion might be as stated: the holder parameterization is simply worse than the simple one in this scenario. However, there is also another factor that plays an important role and that has not been discussed yet: the choice of the “characteristic parameters” of the

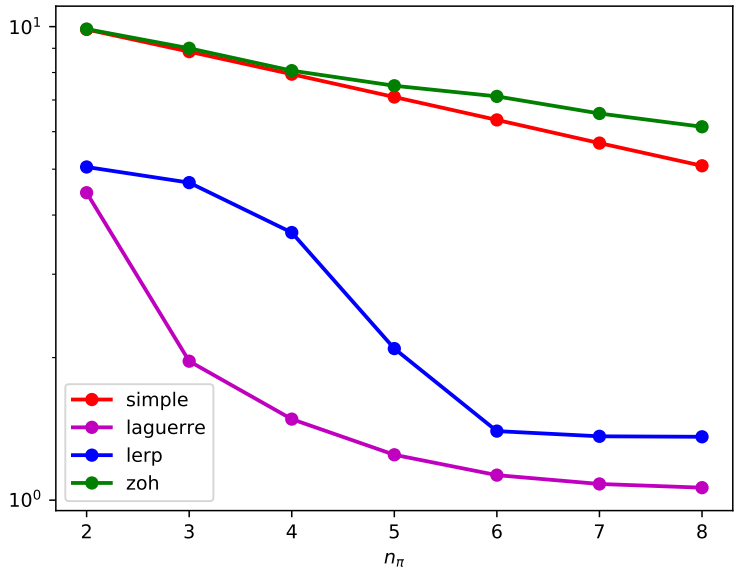


Figure 4.13 – Average tracking errors obtained when controlling a mass-spring-damper system using different parameterizations, as function of the number of parameters n_π. Provided values are relative to the average error obtained when using the full parameterization.

parameterization, *i.e.*, the free samples. As mentioned before, these values were chosen by trial and error. This means that only a subset of the total possibilities has been analyzed, and that a "human factor" was necessary in the process to heuristically choose which sets of indices to consider for the trials. One better approach would be to design a systematic tuning protocol, which nonetheless presents some critical aspects. The major issue is that the set of possible unique choices of the free samples can be incredibly large. In fact, once the prediction horizon and the dimension of the parameter vector have been fixed, the total number of unique free samples selections amounts to[2] $\binom{n_p-1}{n_\pi-1} - 1$. As an example, in the first test presented in this section, where $n_p = 30$ and $n_\pi = 6$ were used, a total of 118754 combinations was possible. With such a large set of potential candidates, it is only natural to rule out brute force methods that explore all the possibilities and choose the best one. It would be possible to design some sort of automatic procedure that, with the help of some heuristic criteria, could reduce the set of candidates to an acceptable amount. However, it does not seem worth the trouble, especially considering that, at least

2. The formula comes from the fact that the first free sample is always located at $k = 0$, and thus one has to choose the other $n_\pi - 1$ samples from a total of $n_p - 1$ remaining possibilities. In addition, one choice is removed, in which all samples are consecutive, since it would correspond to the simple parameterization.

up to now, the LERP and Laguerre parameterizations can lead to better results without the need of complicated tuning procedures.

4.3 Nonlinear Model Predictive Control with Parameterizations

As shown in the previous section, the introduction of parameterizations positively affects the results obtained with MPC schemes designed for linear systems. As a natural extension of previous results, this section considers the broader class of nonlinear dynamical systems. In particular, the following section briefly details the optimization problem resulting from the parameterized formulation, while Section 4.3.2 focuses on simulations that were performed on a simple nonlinear system. Finally, Section 4.3.3 applies the parameterization to visual servoing, considering a free-flying camera that has to perform a given trajectory in front of a simple target.

4.3.1 Problem Formulation

The nonlinear problems arising from the use of parameterizations in predictive control are not so different from those that were formulated in the beginning of Chapter 3 when discussing direct methods.

A first possible formulation is the equivalent of single shooting methods when dealing with parameterizations. In this case, given a particular value for the parameter vector $\boldsymbol{p}$, a first control sample can be evaluated using (4.1), which in turn is used to evaluate the first predicted state. At the next step, (4.1) is used again to obtain a second control input, and a new prediction is formulated from its value and that of the first predicted state sample. The procedure is repeated for the whole prediction vector, and is conceptually equivalent to the one that had been outlined in (3.11). Assuming the quadratic stage cost (3.9), the objective function and the constraints can be evaluated on top of the parameterization, and thus the optimization problem can be transcribed as:

$$\min_{\boldsymbol{p}} \sum_{k} C(\boldsymbol{x}_{k+1}(\boldsymbol{x}_0, \boldsymbol{p}), \boldsymbol{u}_k(\boldsymbol{p}), k) \tag{4.24a}$$

subject to:

$$\boldsymbol{x}_k(\boldsymbol{x}_0, \boldsymbol{p}) \in \mathcal{X} \quad \forall k = 1, \cdots, n_p \tag{4.24b}$$

$$\boldsymbol{u}_k(\boldsymbol{p}) \in \mathcal{U} \quad \forall k = 0, \cdots, n_p - 1 \tag{4.24c}$$

wherein the explicit dependencies on $\boldsymbol{p}$ and $\boldsymbol{x}_0$ have been used to remind that $\underline{\boldsymbol{u}}$ and $\underline{\boldsymbol{x}}$ are fully defined by the parameters and the initial state of the system.

A second possible problem formulation consists in extending multiple-shooting algorithms to the case of parameterizations. As outlined in Section 3.1.2, the core concept is to "promote" dependent variables to the role of decision variables, adding adequate equality constraints to ensure that they are coherently linked to the original optimization variables according to their previous definition. In practice, the whole optimization problem could be written as:

$$\min_{\boldsymbol{p}, \underline{\boldsymbol{x}}, \underline{\boldsymbol{u}}} \sum_k C(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k, k) \tag{4.25a}$$

subject to:

$$\boldsymbol{x}_k \in \mathcal{X} \quad \forall k = 1, \cdots, n_p \tag{4.25b}$$

$$\boldsymbol{u}_k \in \mathcal{U} \quad \forall k = 0, \cdots, n_p - 1 \tag{4.25c}$$

$$\boldsymbol{x}_{k+1} - \boldsymbol{f}_d(\boldsymbol{x}_k, \boldsymbol{u}_k) = \boldsymbol{0} \quad \forall k = 0, \cdots, n_p - 1 \tag{4.25d}$$

$$\boldsymbol{u}_k - \boldsymbol{\pi}_k(\boldsymbol{p}) = \boldsymbol{0} \quad \forall k = 0, \cdots, n_p - 1 \tag{4.25e}$$

Further options could be considered, *e.g.*, by adding only either $\underline{\boldsymbol{x}}$ or $\underline{\boldsymbol{u}}$ to the decision vector (and consequently removing (4.25d) or (4.25e) as needed). In addition, inspired by what was done in [Gif+18], hybrid multiple shooting variants of problem (4.25) could be considered in which only a subset of the control and state samples are added to the decision variables.

The simulations presented in the next section all exploit the first formulation, and are solved using a custom implementation of the Gauss-Newton minimization algorithm (*cfr.* Appendix B.4), which can be viewed as a special case of Sequential Quadratic Programming. Only linear parameterizations have been used up to now, subject to control bounds and control variation limits. As a consequence, all constraints are linear, which simplifies the problem at hand, ensuring that the returned parameter values are always

within the valid domain.

Implementations based on the multiple-shooting formulation have yet to be tested, mainly since the optimization algorithm needs to be created ad-hoc in order to properly exploit the sparsity of the problem and to enhance parallelization.

4.3.2 Control of an Inverted Crane-Pendulum

We consider here an example in which an inverted crane is to be controlled. The system, represented in Figure 4.14, consists in a moving support of mass M which can slide along a rail under the action of a horizontal force u. A pendulum, which is able to freely rotate thanks to a frictionless joint, is attached to the base. The rod of the pendulum has length ℓ and is considered to be massless, while the terminal weight has given mass m and inertia I. The system is underactuated, as the rotation of the pendulum is not directly controlled by any actuation system.

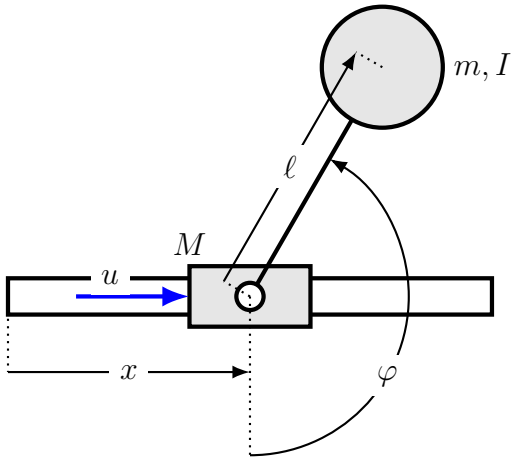


Figure 4.14 – Sketch of the crane-pendulum system.

The state of the system consists in the horizontal position of the base, denoted as x, and the angle of rotation of the pendulum, φ. The derivatives of these two variables are also part of the state vector, while the horizontal force $\boldsymbol{u}$ applied at the base corresponds to the unique input of the system.

To obtain the equations of motions, the Lagrangian mechanics is used. In particular, the kinetic energies of the moving support and of the pendulum write as:

$$E_1 = \frac{1}{2} M \dot{x}^2 \tag{4.26a}$$

$$E_2 = \frac{1}{2}\left(m\left\|\begin{bmatrix}\dot{x}+\ell\dot{\varphi}\cos\varphi\\ \ell\dot{\varphi}\sin\varphi\end{bmatrix}\right\|^2 + I\dot{\varphi}^2\right) = \frac{1}{2}\left(m\dot{x}^2 + (m\ell^2+I)\dot{\varphi}^2 + 2m\ell\,\dot{x}\dot{\varphi}\cos\varphi\right) \tag{4.26b}$$

Regarding the potential energy, since the moving support moves horizontally the only contribution that needs to be considered is the one of the disc, and corresponds to:

$$U_2 = -mg\ell\cos\varphi \tag{4.27}$$

The Lagrangian $\mathcal{L}$ of the system is defined as the difference between the total kinetic and potential energy, which, for the crane-pendulum, results in:

$$\mathcal{L} = E_1 + E_2 - U_2 = \frac{1}{2}\underbrace{(M+m)}_{m_t}\dot{x}^2 + \frac{1}{2}\underbrace{\left(m\ell^2+I\right)}_{I_0}\dot{\varphi}^2 + m\ell\left(\dot{x}\dot{\varphi}+g\right)\cos\varphi \tag{4.28}$$

The following step consists in the evaluation of the derivatives of $\mathcal{L}$ with respect to the generalized coordinates $\boldsymbol{\varrho} \doteq \begin{bmatrix}x & \varphi\end{bmatrix}^T$:

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{\varrho}} = \begin{bmatrix}0\\ -s_\varphi m\ell\left(\dot{\varphi}\dot{x}+g\right)\end{bmatrix}^T \tag{4.29a}$$

$$\frac{\partial\mathcal{L}}{\partial\dot{\boldsymbol{\varrho}}} = \begin{bmatrix}\dot{\varphi}c_\varphi m\ell + \dot{x}m_t\\ I_0\dot{\varphi} + \dot{x}c_\varphi m\ell\end{bmatrix}^T \tag{4.29b}$$

$$\frac{\partial}{\partial t}\frac{\partial\mathcal{L}}{\partial\dot{\boldsymbol{\varrho}}} = \begin{bmatrix}\ddot{\varphi}c_\varphi m\ell + \ddot{x}m_t - \dot{\varphi}^2 s_\varphi m\ell\\ I_0\ddot{\varphi} + \ddot{x}c_\varphi m\ell - \dot{\varphi}\dot{x}s_\varphi m\ell\end{bmatrix}^T \tag{4.29c}$$

wherein s_φ and c_φ have been used to shorten the notation, and correspond respectively to the sine and cosine of the angle φ. Knowing these derivatives, the inverse dynamics writes as:

$$\begin{bmatrix}u\\0\end{bmatrix} = \frac{\partial}{\partial t}\frac{\partial\mathcal{L}}{\partial\dot{\boldsymbol{\varrho}}}^T - \frac{\partial\mathcal{L}}{\partial\boldsymbol{\varrho}}^T = \begin{bmatrix}m_t & c_\varphi m\ell\\ c_\varphi m\ell & I_0\end{bmatrix}\begin{bmatrix}\ddot{x}\\ \ddot{\varphi}\end{bmatrix} + \begin{bmatrix}-\dot{\varphi}^2 s_\varphi m\ell\\ g s_\varphi m\ell\end{bmatrix} \tag{4.30}$$

By inversion of said model, the Direct Dynamics Equations of the system are finally found:

$$\ddot{x} = \frac{I_0 \left(\dot{\varphi}^2 s_\varphi m\ell + u\right) + gm^2\ell^2 s_\varphi c_\varphi}{I_0 m_t - c_\varphi^2 m^2 \ell^2} \tag{4.31a}$$

$$\ddot{\varphi} = -\frac{m\ell \left(c_\varphi \left(\dot{\varphi}^2 s_\varphi m\ell + u\right) + gm_t s_\varphi\right)}{I_0 m_t - c_\varphi^2 m^2 \ell^2} \tag{4.31b}$$

Despite this being a rather simple model with reduced dimensionality and a single control input, the equations of motion present several highly nonlinear terms which justify the use of this model as a first benchmark for nonlinear optimal control.

The crane was simulated in Gazebo via the Bullet physics engine, using the geometric and physical properties reported in Table 4.1. To obtain more realistic results, the simulation was updated at a much higher frequency (10 kHz) than the one used for the control (50 Hz), holding the same input for the entire duration of a control period. A total of $n_p = 20$ samples was used for predictions, with sampling time of 50 ms. Control inputs were limited to a maximum of 0.1 N in absolute value, with a maximal allowed variation of 0.02 N between consecutive control samples.

Table 4.1 – Constants used in Gazebo to simulate the crane-pendulum.

M	m	I	ℓ	m_t	I_0
0.25 kg	0.1178 kg	0.001 25 kg m^2	0.3 m	0.3678 kg	0.011 852 kg m^2

The objective of the controller in each simulation is to move the base of the crane, switching between the two positions $x_- = 0\,\text{m}$ and $x_+ = 0.3\,\text{m}$. While doing so, the pendulum should remain vertical and pointing upward, *i.e.*, $\varphi^\star = \pi\,\text{rad}$. Since the goal is to stabilize the system at fixed configurations, the desired linear and angular velocities $\dot{x}^\star$ and $\dot{\varphi}$ are both set to zero. To mitigate the relative importance of each task, in the objective the weights of the different state variables were set to 200, 10, 7 and 2 respectively for the base position, the pendulum orientation, the linear velocity of the base and the rotation rate of the pendulum.

In addition to the full parameterization used as reference for comparisons, the usual four parameterizations have been considered:

1. The simple parameterization from classical MPC theory.
2. ZOH parameterizations with free samples located at various spots along the prediction horizon. The goal is to have enough samples to steer both in the short and

long term.

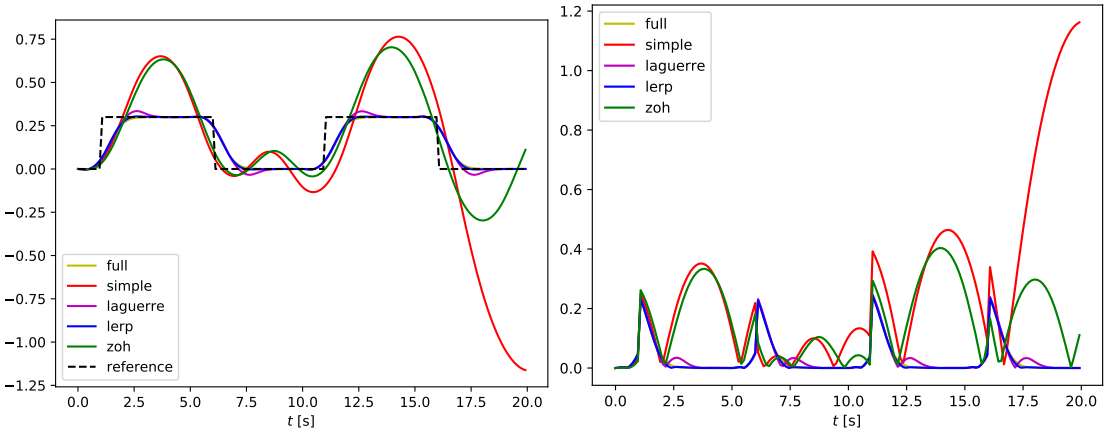
3. LERP parameterizations with free samples located similarly to ZOH.

4. Laguerre parameterization with $\varepsilon = 1.5$.

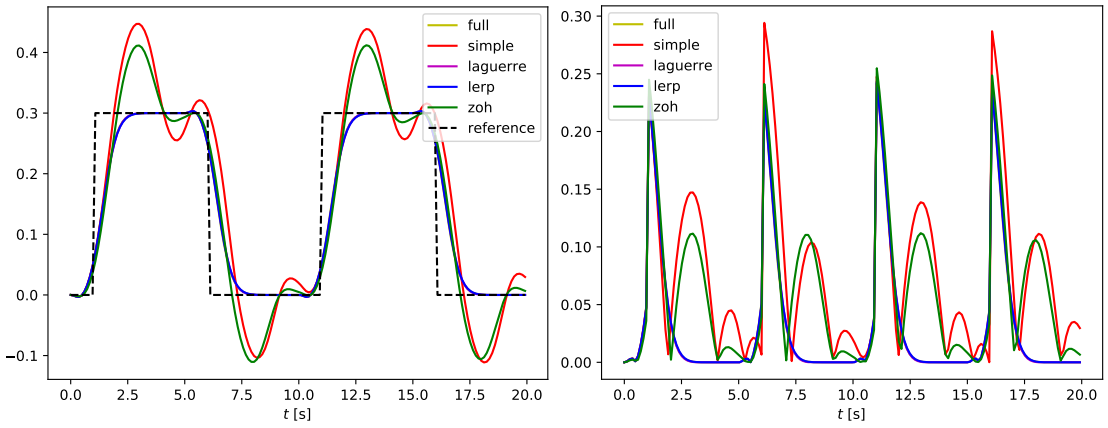
Similarly to the simulations performed with linear systems, the parameterizations have been compared using different values of n_π. In particular, Figure 4.15 reports the results from three specific cases, in which the number of parameters is respectively 3, 6 and 8. A first necessary observation is related to the test presented in Figures 4.15(a) and 4.15(b), in which $n_\pi = 3$. In this test, the simple parameterization is the only one that is not able to properly stabilize the crane, with the base position deviating more and more from the desired reference. It must also be remarked that the holder parameterization, despite being able to at least bound the oscillations in the tracking error, provides results which are far from satisfactory, with peak errors that reach the 0.4 m. On the other hand, the Laguerre parameterization and especially the LERP one perform really well. Indeed, the former features only small overshooting when a desired position is reached, while the latter presents a behavior that is nearly indistinguishable from that of the full parameterization.

Increasing the number of parameters allows the simple parameterization to find stabilizing controls, but, other than that, the results confirm what was found in the first simulation. More precisely, in both remaining simulations, the ZOH parameterization slightly outperforms the simple one, even though the maximum overshooting is still too large, *i.e.*, around 11 cm and 3.5 cm when using $n_\pi = 6$ and $n_\pi = 8$ respectively. On the other hand, the LERP and Laguerre parameterizations perform as well as the full parameterization in both remaining simulations.

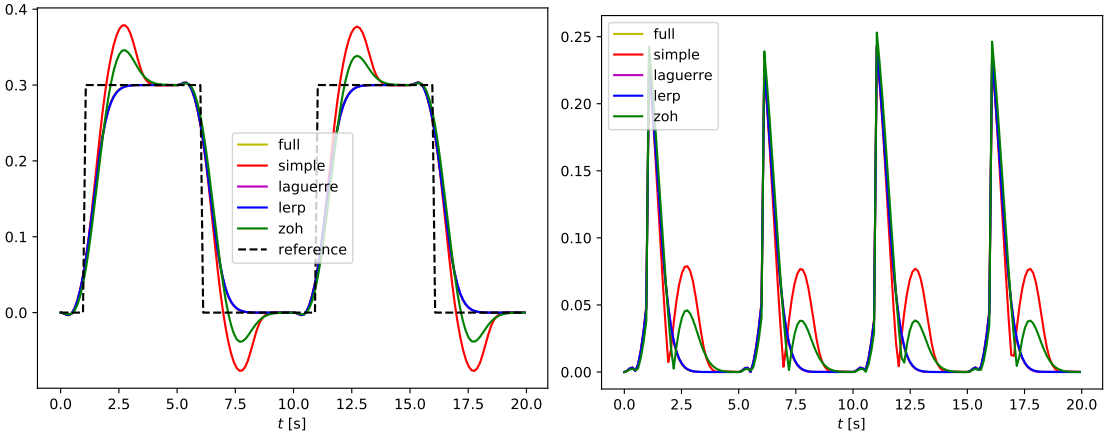
To conclude the present analysis, an overall comparison of the average position error as function of the number of parameters is provided in Figure 4.16. The overall performances reflect what shown in the selected simulations, and provide similar conclusions to those found when dealing with linear systems. The holder parameterization can provide some benefits, but they are often marginal and less consistent. On the other hand, the advantages in using a LERP or a Laguerre parameterization are clear, with performances that are comparable to those of the full parameterization, but using as few as 3 or 4 parameters, *i.e.*, 80 % less decisions variables than those used in a full parameterization.



(a) Position ($n_\pi = 3$) (b) Tracking error ($n_\pi = 3$)



(c) Position ($n_\pi = 6$) (d) Tracking error ($n_\pi = 6$)



(e) Position ($n_\pi = 8$) (f) Tracking error ($n_\pi = 8$)

Figure 4.15 – Simulation results from the crane-pendulum control problem. Each row corresponds to a different simulation set, with increasing number of parameters. For each set, the position of the base and the corresponding tracking error are shown respectively on the left and right graphs.

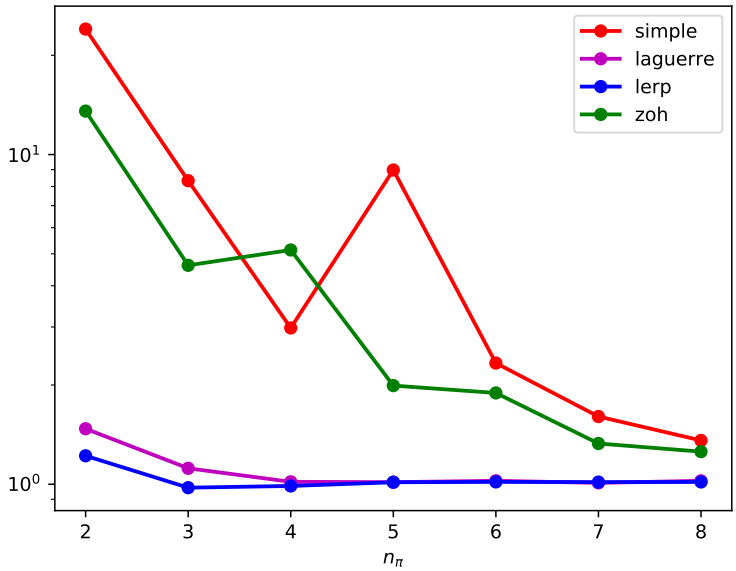


Figure 4.16 – Comparison of four parameterizations while controlling a crane-pendulum system. The four signals represent the average position tracking error of each parameterization as function of the number of parameters n_π, normalized with respect to the full parameterization.

4.3.3 Control of a Free-Flying Camera

In this last section, the use of parameterizations in a VPC scheme is investigated. In particular, a set of simulations was run considering a free-flying camera with 6 dof. The control input is assumed to be the sensor twist projected in its own frame, *i.e.*, $\mathbf{v}$. The considered prediction model is a local model that updates both the features $\boldsymbol{s}$ and the parameters $\boldsymbol{z}$ using the interaction matrix evaluated at each iteration. The model is thus simpler than those considered in the previous chapter, since the manipulator and its kinematics are not taken into account.

The observed object is once again a simple target with four black dots printed on it, and the normalized coordinates of the four centers are used as features. As done in all previous servoing simulations and experiments, the object is assumed to be motionless.

In each simulation, the camera is initially positioned in front of the target at a distance of 2.5 m, with the task of performing a circular trajectory in front of the observed object (the diameter of this circular motion being of 1 m). In terms of predicted visual features, the objective is to ensure that the coordinates $\boldsymbol{s}_i$ of the i-th feature point follow the desired

profile along the prediction horizon:

$$\boldsymbol{s}^{\star}_{i,k} = \boldsymbol{s}^{(0)}_{i} + R \begin{bmatrix} \cos\left(t + k\Delta t\right) \\ \sin\left(t + k\Delta t\right) \end{bmatrix} \tag{4.32}$$

wherein $\boldsymbol{s}^{(0)}_{i}$ are the coordinates of the i-th point when the camera is located in front of the object, at the beginning of the simulation. The parameter R is proportional to the radius of the camera trajectory ($R = 0.2$ was used in the simulations), while t corresponds to the time at which the current MPC problem has to be defined and solved. Finally, Δt is the sampling time used in the predictors. The desired features thus change continuously during the simulation, as opposed to the cases considered in the previous chapter, in which the values of $\boldsymbol{s}^{\star}_{i,k}$ were kept constant. As explained before, a generic visual task constrains 6 dof, allowing in this case to fully determine the control input (the 6-dimensional sensor twist) without the need of a secondary task, as it was done when controlling the 7 dof LWR4+ manipulator.

The simulation was run with a control frequency of 50 Hz to simulate typical camera acquisition frequencies. Since the task to be performed corresponds to the tracking of a time-varying trajectory, it is important to use a rather precise sampling step for the prediction. In particular, $\Delta t = \Delta t_c = 20\,\text{ms}$ was used in all simulations discussed in the following. If the tracking is performed properly, the prediction horizon needs not to be extremely large. However, it would be desirable to still have a sufficiently long prediction vector in case any perturbation leads to an increase in the feature error. This can happen, *e.g.*, if the observed object performs some kind of motion, since this is not modeled in the predictor. It was decided to consider a prediction time of $T_p = 1.5\,\text{s}$, thus leading to a prediction horizon of $n_p = 75$ samples.

All parameterizations described so far were tested in simulation, with an increasing number of decision variables. In particular, the simple parameterization was considered with a control horizon n_c ranging from 2 to 10, and thus the number of parameters n_π varies from 12 to 60 (since a single control sample has dimension $m = 6$). Similarly, ZOH and LERP parameterizations were considered, each featuring a number of free samples ranging from 2 to 10, with the free samples being uniformly spread along the prediction horizon. Finally, Laguerre parameterizations were also used, each featuring a static offset (like in the case of the mass-spring-damper of Section 4.2.3) and as many basis functions as needed to have the same amount of parameters as other control parameterizations. The value $\varepsilon = 0.5$ was used to tune the damped Laguerre polynomials.

The optimization problems were formulated using the single-shooting formulation (4.24) and solved using an iterative Gauss-Newton method. Constraints on the control inputs were added to limit the linear and angular velocities to $0.5\,\mathrm{m\,s^{-1}}$ and $1\,\mathrm{s^{-1}}$ respectively. In addition, the maximum variations were set to $0.017\,\mathrm{m\,s^{-1}}$ and $0.033\,\mathrm{s^{-1}}$, approximately corresponding to acceleration limits of $0.83\,\mathrm{m\,s^{-2}}$ and $1.67\,\mathrm{s^{-2}}$.

A synthetic representation of the results is represented in Figure 4.17, which provides the features error norm along time for different parameterization types and number of parameters. In particular, each type of parameterization is plotted with a specific hue component, with darker colors representing the use of fewer parameters. For ease of comparison, the results obtained using the maximum number of optimization variables ($n_\pi = m \cdot n_p = 6 \cdot 75 = 450$) are included as well.

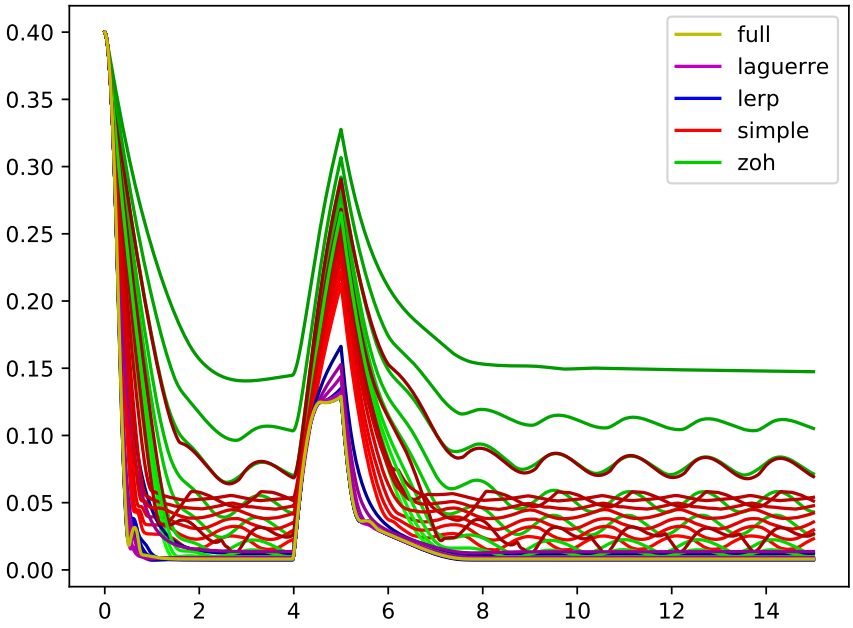


Figure 4.17 – Comparison of tracking error norms for different parameterizations in the visual servoing simulations. The value of n_π was $m \cdot j = 6j$, with j ranging from 2 ($n_\pi = 12$) to 10 ($n_\pi = 60$). In the graph, darker colors correspond to smaller numbers of parameters.

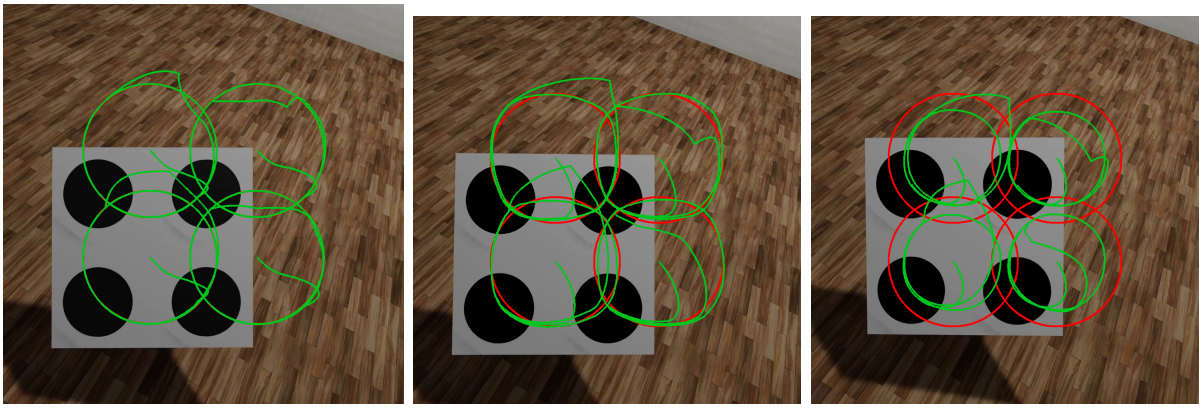
As visible, all parameterization feature two distinct peaks in the features error vector. The first one is obtained at the very beginning of the simulation, and is simply due to the features being initially located at the centers of the desired circular trajectories. In addition, a second peak appears after 4 s of simulation. This is caused by the object, which performs a rotational motion of 70° between $t = 4\,\mathrm{s}$ and $t = 5\,\mathrm{s}$. The center of rotation corresponds to the origin of the object itself and is performed around the unit

axis $\left(\sqrt{1/6}, \sqrt{1/6}, \sqrt{2/3}\right)$ at constant velocity. As mentioned before, this motion is not modeled in the prediction and thus it cannot be anticipated. As a result, it leads to an increased tracking error.

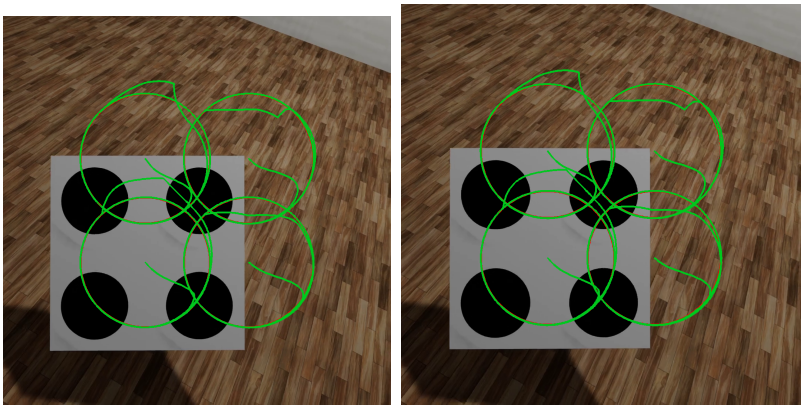
Two observations can be made from the results depicted in Figure 4.17. First of all, the ZOH parameterization does not lead to satisfactory performances when a small amount of parameters is used. The tracking error after transient becomes smaller than that of the simple parameterization only if at least 8 free samples are used. As already discussed, given the extremely high number of possible choices of the free samples, the existence of a better selection cannot be easily ruled out. However, this once again shows one (and perhaps the main) weakness of this parameterization, which seems to be effective only after careful selection of the free samples locations.

The second observation is that very few parameters are needed to obtain good performances with LERP and Laguerre parameterizations. In particular, using just 3 free samples in a LERP parameterization leads to tracking errors that are almost equivalent to those obtained with the full parameterization, which uses instead 75 samples. A slightly worse, but definitely very close, behavior is obtained with Laguerre parameterizations. This can be seen also in Figure 4.18, which compares the features trajectories obtained with the different parameterizations when 3 parameters per input coordinate are used. In particular, the motions obtained using the full 4.18(a), LERP 4.18(d) and Laguerre 4.18(e) are really close in the image. Moreover, it must be noted that the rate at which peak errors are reduced is relatively high. Indeed, it generally takes around 0.5 s to almost nullify the tracking error when LERP and Laguerre parameterizations are used, given that a sufficient amount of degrees of freedom are used (again, this roughly corresponds to a minimum of 18 or 24 total decision variables).

Finally, from a practical point of view it must be noted that even with the reduced number of $n_\pi = 18$ parameters used to obtain the results of Figure 4.18 the optimization time required to evaluate the control at each iteration is about 300 ms, which is evidently too large for a real-time implementation. Nonetheless, it corresponds to just slightly more than the 5 % of the time required when using the full parameterization, and thus it provides a very significant relative improvement. Furthermore, the results were obtained with a simple implementation written entirely in Python, and it seems reasonable that real-time execution could be achieved with some additional code optimization and by translating the source to a compiled language.



(a) Full (b) Simple (c) ZOH



(d) LERP (e) Laguerre

Figure 4.18 – View of the target as observed from the free-flying camera, with the desired (red) and actual (green) trajectories of the dots. The first four images correspond to parameterizations using 3 parameters for each control coordinate, for a total of $n_\pi = 18$ parameters. The last image corresponds to a full parameterization with $n_\pi = 450$ parameters.

4.4 Conclusions

This chapter focused on the concept of control input parameterizations, which allows to synthesize whole command sequences from a low-dimensional set of parameters. After detailing some possible kinds of parameterizations, their performances were tested in simulation, both in the case of linear and nonlinear systems, including a simple example of visual predictive control. The discussed results show that better performances than the classical formulation can be obtained even when using a small amount of parameters with respect to the number of prediction samples.

This topic has not yet been considered extensively in robotics yet, but it is believed that it deserves to be studied more in details, with few open questions that need to be answered. First of all, the use of parameterizations for more complex nonlinear systems should be considered in order to asses their performances in practical scenarios. Examples might include underactuated systems or serial manipulators performing a visual servoing task.

Another direction of investigation should focus on different formulations of the nonlinear optimization problem. In fact, as discussed before, only single-shooting methods have been investigated so far, while it is worth to extend the analysis to multiple-shooting approaches as well.

Moreover, the tuning problem should be addressed as well. All parameterizations contain a certain number of "design constants" that are to be set before the optimization. This task is currently performed by the user, which has to resort to his/her experience, knowledge of the system and some intuition as well. Instead, it would be desirable to have some tuning criteria or even some automatic procedures that can reduce the amount of human intervention. In this sense, a first attempt could be to include the values that define a parameterization within the decision variables of the optimization. An example could be to consider nonlinear Laguerre parameterizations in which the scaling factor ε is also optimized by the nonlinear programming routine.

Last but definitely not least, it would be of great interest to mix the concepts illustrated here with those presented in Chapter 3, assessing the performances of parameterized VPC schemes based on second-order feature models.

Conclusions

This thesis dealt with the objective of increasing the productivity of manipulators performing visual servoing tasks. In particular, different ways to allow faster execution of the relative positioning of a sensor with respect to an observed object were investigated. To address the problem, several strategies were considered, either involving the use of new models to describe the interaction between the controlled robot and its environment, or the employment of advanced control schemes that try to optimize the motion of the robot. The contributions were validated via simulations and experiments, using mainly a Kuka LWR4+ as target platform. It was shown that the proposed methods are well suited for the control of such system, allowing to reach medium speeds and relatively short execution times.

Contributions

More than few existing works suggest that a key to unlock high-speed performances is the use of higher-order models that allow to establish a link between the features and the acceleration and actuation efforts of the controlled system. Following this suggestion, this thesis firstly investigated second-order models for visual servoing. In particular, a simple procedure was proposed that allows to obtain these models for a generic eye-in-hand system, detailed in the beginning of Chapter 2. It allowed to obtain second-order models for various sets of features, *cfr.* Appendix A, including image points, polar coordinates and planar image moments.

The use of second order models allows to create new classes of control schemes which use the acceleration – either of the sensor or of the manipulator depending on the formulation – as input. In these controllers, the state is not only given by the features, but also by their velocities, which can in turn be estimated using existing first-order models. One of the main interests in obtaining an acceleration control signal directly from the features observations is that it can be used in conjunction with the dynamic model of the controlled manipulator to establish a Computed Torque Control directly in the feature space. As it was shown by means of simulations in [FKM19], such controller is less af-

fected by sensor noise and allows to obtain satisfactory performances in terms of features convergence.

One limitation of the feedback controllers proposed in Chapter 2 is that, similarly to classical proportional velocity control laws, they constrain the features evolution in a very specific way. In particular, controllers attempt to steer the features along straight hyper-lines connecting the current configuration to the desired one, which can result in unsatisfactory motions for the sensor in SE(3). In addition, the rate at which this motion is performed is dictated by an exponential decay. As a consequence, if the control gains are increased to reduce execution times, very high accelerations and velocities might be produced for a short period of time, while near completion the sensor will tend to move rather slowly.

To overcome these issues, predictive control strategies were examined. The principal advantage over feedback controllers is the ability of MPC schemes to explore a set of solutions and select the control inputs which optimally steers the system towards the desired configuration. By properly shaping the cost functional of the optimization it is possible to favor shorter paths and faster motions, both contributing in a reduction of the execution time for a task. Moreover, as the control computation is the result of a nonlinear optimization, arbitrary constraints can be added to the problem definition, thus ensuring physical feasibility of the returned solution.

The main drawback of MPC is the computational burden that it introduces: rather heavy nonlinearly constrained nonlinear minimization problems in several variables must be solved in few milliseconds in order to allow feasible real-time implementations. To make the problems manageable in this sense, two different strategies have been considered that improve state of the art VPC techniques.

The first attempt was inspired by works on second-order models, which underlined that features and sensor accelerations become non-negligible in more dynamic contexts. It was decided to consider predictors that exploit also the second derivative of the features to predict their future values. These models allow to use coarser time samplings without sacrificing reliability of the predictions. As a consequence, less samples need to be considered so as to cover a target period of time. The complexity of each optimization problem can be reduced, and the problem can be solved within the allotted time with sufficient precision. This is supported via several simulations and experiments, as presented in Chapter 1 and [FKM20]. In particular, the results show that models that incorporate acceleration information are associated with generally better and shorter motions, and

can outperform predictive strategies based on velocity models. Moreover, the use of different features sets in real experiments suggest that this result is due to the proposed methodology and not specific to a single type of descriptive features.

A second way to deal with the complexity of predictive strategies was analyzed as well, based on the ideas introduced in [Ala06]. As opposed to classical predictive strategies, which try to reduce the dimensionality of the optimization by using as decision variables the first samples of the control sequence, this theory proposes to introduce a parameterization for the control inputs so that the "degrees of freedom" are better distributed along the prediction horizon. Given that the concept was rarely applied to robotics and very few related works are available, the majority of Chapter 4 has been dedicated to a thorough analysis of these parameterizations on simple systems. In particular, they were used in simulation to control two common linear systems. The results were discussed, underlining the advantages coming from the use of two parameterizations in particular. The method was then applied for the control of a simple nonlinear system – an inverted crane-pendulum – and the results confirmed what was previously obtained when considering linear systems. The parameterized approach was finally exploited to perform a simulated visual servoing task. In particular, a free-flying perspective camera was controlled in velocity using different types of control parameterizations, subject to actuation constraints. A relevant result was obtained in this case: using just a 25^{th} of the maximum parameters, it is possible to obtain very satisfactory tracking performances, comparable to those obtained when all control samples are individually optimized. This allowed to obtain reduced errors and fast responses in a short time (compared to that needed by the full parameterization). The presented results have not been published yet, but a journal article is currently in preparation in order to share these findings with the scientific community and valorize the thesis work.

Perspectives

This thesis contributed to the enhancement of visual servoing performances by following three different directions. Even if the outcomes of these investigations were positive and allowed to control sensor displacements at higher speed, there are still several aspects that need deeper exploration.

First of all, it must be noted that the models and controls presented in Chapter 2 depend on the derivatives of the feature vector. In the present study, it was simply as-

sumed that such derivative could be estimated by multiplication of the feature Jacobian by the joint velocity, *i.e.*, $\dot{\boldsymbol{s}} = \mathbf{J}_{\boldsymbol{s}}\dot{\boldsymbol{q}}$. The assumption was not unreasonable in the presented simulations since perfect knowledge of the geometric model of the manipulator was assumed, and the noise in the sensor measurements was relatively small. However, a second possibility could be considered, consisting in using an estimation scheme to evaluate $\dot{\boldsymbol{s}}$ from fusion of several measurements. As an example, an Inertial Measurement Unit (IMU) could be used to obtain an estimation of the end-effector velocity, and an optical flow strategy could provide a direct estimation of $\dot{\boldsymbol{s}}$ from image measurements. Fusing all these signals altogether could provide a better overall estimation of the feature velocity.

As explained in Chapter 2, simulations were run using a very high sampling time, assuming nonetheless that a rather reliable estimation of both the features and parameters would have been available. Therefore, in order to validate the presented simulations also on a real platform, a good estimation of the feature vector itself would be essential. To obtain it, different solutions could be considered. A first possibility would be to use a virtual visual servoing strategy similar to that presented in [Dah+12], or to use a solution based on an extended Kalman filter which exploits both the first and second-order features interaction models to perform simultaneous estimation of the features and of their velocity.

Simulations and experiments presented in this manuscript did not consider high-speed systems, such as the parallel robots used in [Dah+12; Ozg+13]. Extending the results obtained in this thesis to such class of manipulators would be of high interest, since their mechanical design allows to reach very high speeds and accelerations. Of course, they constitute a higher challenge with respect to the control of serial manipulators such as the LWR4+ Kuka arm that was used in most of the simulations and experiments presented in this thesis. As an example, the visual servoing controller based on the Inverse Dynamic Model presented in this thesis did not take into account some behaviors that typically appears in highly dynamic contexts such as flexibility in the joints and links. It would be interesting to assess whether the performances obtained with the serial manipulators hold also with higher dynamic excitation, or if more accurate models are needed. Furthermore, if motions become significantly faster, the quality of acquired images might deteriorate. As an example, motion blur and rolling shutter might rend the visual feedback highly unreliable. A potential solution to this problem could consist in the use of event-based and hybrid cameras such as the Dynamic and Active-pixel Vision Sensor (DAVIS) [Bra+14; Ted+16].

Several improvements to predictive strategies are also possible. First of all, up to now

only simple geometric primitives have been used as features, and the object used for simulations and experiments was simplified to facilitate features extraction. Even though image points can be used to represent many real-life objects, to deal with complex shapes it might be better to consider image moments as descriptors. To extend the domain of applicability of the presented predictive strategies, a relevant improvement would thus be to develop new predictors for this class of features. This poses some challenges for few reasons. A first, simple, one is the difficulty in obtaining second order models for these descriptors. To support this claim, the reader is referred to Appendix A.5, which contains the second order model of a generic moment feature $m_{i/j}$, which is composed of several terms. Fortunately, despite the large amount of factors which make the modeling task tedious and error-prone, the algorithmic procedure detailed in Chapter 2 allows to evaluate the necessary quantities with the help of symbolic tools. Nonetheless, a second issue complicates the definition of predictors according to the formalism proposed in Chapter 3: each moment $m_{i/j}$ depends on moments of higher order such as $m_{i+1/j}$. As a consequence, to predict $m_{i/j}$ one has to evaluate $m_{i+1/j}$ as well, which in turns requires to predict $m_{i+2/j}$. This chain of dependencies on higher-order moments is infinite, and thus the predictor needs to be defined differently. One possibility would be, *e.g.*, to use a constant value for all moments of a specific order, or to use a constant interaction matrix along the prediction horizon.

Another interesting development would consist in the application of parameterized MPC to systems with unstable behaviors, such as an aerial vehicle performing a servoing task. Being an underactuated and inherently unstable system, this case is similar to that of the inverted pendulum of Section 4.3.2. In addition, such systems often have a limited computational power. This makes them a perfect target for the methods presented in the last chapter, whose aim is precisely to reduce the time required to evaluate control inputs.

Finally, several other technical improvements could be attempted. For instance, the use of nonlinear parameterizations should be considered, as well as trying to solve the optimization problem using a multiple-shooting technique. Moreover, a mix between the parameterized approach and the new predictors presented in Chapter 3 should be considered, with the objective of further improving the performances during the execution of servoing tasks.

Appendix A

Second Order Models of Visual Features

In the following some second-order interaction models are reported. In particular, each section focuses on a specific feature set, detailing what are the features $\boldsymbol{s}$ and the parameters $\boldsymbol{z}$ (if any), the corresponding interaction matrices $\mathbf{L}_{\boldsymbol{s}}$ and $\mathbf{L}_{\boldsymbol{z}}$, and finally the $\mathbf{G}$-matrices that define the $\mathbf{h}_{\boldsymbol{s}}$ and $\mathbf{h}_{\boldsymbol{z}}$ vectors appearing in the second-order models.

A.1 3D Points

In this case, the 3D coordinates of a point are used as features, and no parameters are present. The three $\mathbf{G}$-matrices can easily be obtained by following the procedure detailed in Chapter 2.

$$\boldsymbol{s}^T = \begin{bmatrix} X & Y & Z \end{bmatrix} \tag{A.1}$$

$$\mathbf{L}_{\boldsymbol{s}} = \begin{bmatrix} -1 & 0 & 0 & 0 & -Z & Y \\ 0 & -1 & 0 & Z & 0 & -X \\ 0 & 0 & -1 & -Y & X & 0 \end{bmatrix} \tag{A.2}$$

$$\mathbf{G}_X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{Y}{2} & \frac{Z}{2} \\ 0 & 0 & 1 & \frac{Y}{2} & -X & 0 \\ 0 & -1 & 0 & \frac{Z}{2} & 0 & -X \end{bmatrix} \tag{A.3}$$

$$\mathbf{G}_Y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & -Y & \frac{X}{2} & 0 \\ 0 & 0 & 0 & \frac{X}{2} & 0 & \frac{Z}{2} \\ 1 & 0 & 0 & 0 & \frac{Z}{2} & -Y \end{bmatrix} \tag{A.4}$$

$$\mathbf{G}_Z = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -Z & 0 & \frac{X}{2} \\ -1 & 0 & 0 & 0 & -Z & \frac{Y}{2} \\ 0 & 0 & 0 & \frac{X}{2} & \frac{Y}{2} & 0 \end{bmatrix} \tag{A.5}$$

A.2 Image Points

In the case of normalized image points, the interaction matrix is function not only of the features coordinates x and y, but also of the 3D depth of the observed point. Therefore, in order to obtain the required second-order model, the interaction matrix of the depth is to be considered. It simply consists in the last row of the interaction matrix of a 3D point, in which the Cartesian coordinates X and Y are replaced by xZ and yZ. In this way, also $\mathbf{L}_{\boldsymbol{z}}$ depends only on $\boldsymbol{s}$ and $\boldsymbol{z}$, and the matrices $\mathbf{G}_x$ and $\mathbf{G}_y$ can finally be obtained. The expression of $\mathbf{G}_Z$ is also reported, since it is required by the predictors presented in Section 3.3.

$$\boldsymbol{s}^T = \begin{bmatrix} x & y \end{bmatrix} \tag{A.6}$$

$$\boldsymbol{z} = \begin{bmatrix} Z \end{bmatrix} \tag{A.7}$$

$$\mathbf{L}_{\boldsymbol{s}} = \begin{bmatrix} -\frac{1}{Z} & 0 & \frac{x}{Z} & xy & -x^2-1 & y \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & y^2+1 & -xy & -x \end{bmatrix} \tag{A.8}$$

$$\mathbf{L}_{\boldsymbol{z}} = \begin{bmatrix} 0 & 0 & -1 & -Zy & Zx & 0 \end{bmatrix} \tag{A.9}$$

$$\mathbf{G}_x = \begin{bmatrix} 0 & 0 & -\frac{1}{Z^2} & -\frac{y}{Z} & \frac{2x}{Z} & 0 \\ 0 & 0 & 0 & -\frac{x}{Z} & 0 & -\frac{1}{Z} \\ -\frac{1}{Z^2} & 0 & \frac{2x}{Z^2} & \frac{2xy}{Z} & -\frac{2x^2}{Z} & \frac{y}{Z} \\ -\frac{y}{Z} & -\frac{x}{Z} & \frac{2xy}{Z} & x\left(2y^2+1\right) & -\frac{y\left(4x^2+1\right)}{2} & -\frac{x^2}{2}+y^2+\frac{1}{2} \\ \frac{2x}{Z} & 0 & -\frac{2x^2}{Z} & -\frac{y\left(4x^2+1\right)}{2} & 2x\left(x^2+1\right) & -\frac{3xy}{2} \\ 0 & -\frac{1}{Z} & \frac{y}{Z} & -\frac{x^2}{2}+y^2+\frac{1}{2} & -\frac{3xy}{2} & -x \end{bmatrix} \tag{A.10}$$

$$\mathbf{G}_y = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{y}{Z} & \frac{1}{Z} \\ 0 & 0 & -\frac{1}{Z^2} & -\frac{2y}{Z} & \frac{x}{Z} & 0 \\ 0 & -\frac{1}{Z^2} & \frac{2y}{Z^2} & \frac{2y^2}{Z} & -\frac{2xy}{Z} & -\frac{x}{Z} \\ 0 & -\frac{2y}{Z} & \frac{2y^2}{Z} & 2y\left(y^2+1\right) & -\frac{x\left(4y^2+1\right)}{2} & -\frac{3xy}{2} \\ \frac{y}{Z} & \frac{x}{Z} & -\frac{2xy}{Z} & -\frac{x\left(4y^2+1\right)}{2} & y\left(2x^2+1\right) & x^2-\frac{y^2}{2}+\frac{1}{2} \\ \frac{1}{Z} & 0 & -\frac{x}{Z} & -\frac{3xy}{2} & x^2-\frac{y^2}{2}+\frac{1}{2} & -y \end{bmatrix} \tag{A.11}$$

$$\mathbf{G}_Z = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -Z & 0 & \frac{Zx}{2} \\ -1 & 0 & 0 & 0 & -Z & \frac{Zy}{2} \\ 0 & 0 & 0 & \frac{Zx}{2} & \frac{Zy}{2} & 0 \end{bmatrix} \tag{A.12}$$

A.3 Polar Coordinates

This case is similar to the previous one, in which normalized points were used, and the parameter vector still consists in the 3D depth of the observed point. Polar coordinates and image points are linked by the well-known relation

$$x = \rho \cos\theta \tag{A.13}$$

$$y = \rho \sin\theta \tag{A.14}$$

which can be used to obtain an expression for $\boldsymbol{L}_{\boldsymbol{z}}$ and $\mathbf{G}_Z$ that depends only on ρ, θ and Z.

$$\boldsymbol{s}^T = \begin{bmatrix} \rho & \theta \end{bmatrix} \tag{A.15}$$

$$\boldsymbol{z} = \begin{bmatrix} Z \end{bmatrix} \tag{A.16}$$

$$\mathbf{L}_{\boldsymbol{s}} = \begin{bmatrix} -\frac{\cos(\theta)}{Z} & -\frac{\sin(\theta)}{Z} & \frac{\rho}{Z} & (\rho^2+1)\sin(\theta) & (-\rho^2-1)\cos(\theta) & 0 \\ \frac{\sin(\theta)}{Z\rho} & -\frac{\cos(\theta)}{Z\rho} & 0 & \frac{\cos(\theta)}{\rho} & \frac{\sin(\theta)}{\rho} & -1 \end{bmatrix} \tag{A.17}$$

$$\mathbf{L}_{\boldsymbol{z}} = \begin{bmatrix} 0 & 0 & -1 & -Z\rho\sin(\theta) & Z\rho\cos(\theta) & 0 \end{bmatrix} \tag{A.18}$$

$$\mathbf{G}_\rho = \begin{bmatrix} \frac{\sin^2(\theta)}{Z^2\rho} & -\frac{\sin(2\theta)}{2Z^2\rho} & -\frac{\cos(\theta)}{Z^2} \\ -\frac{\sin(2\theta)}{2Z^2\rho} & \frac{\cos^2(\theta)}{Z^2\rho} & -\frac{\sin(\theta)}{Z^2} \\ -\frac{\cos(\theta)}{Z^2} & -\frac{\sin(\theta)}{Z^2} & \frac{2\rho}{Z^2} \\ -\frac{\left(\rho^2-1\right)\sin(2\theta)}{2Z\rho} & \frac{\rho^2\cos^2(\theta)-2\rho^2-\cos^2(\theta)}{Z\rho} & \frac{2\rho^2\sin(\theta)}{Z} \\ \frac{-\rho^2\sin^2(\theta)+2\rho^2+\sin^2(\theta)}{Z\rho} & \frac{\left(\rho^2-1\right)\sin(2\theta)}{2Z\rho} & -\frac{2\rho^2\cos(\theta)}{Z} \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} -\frac{\left(\rho^2-1\right)\sin(2\theta)}{2Z\rho} & \frac{-\rho^2\sin^2(\theta)+2\rho^2+\sin^2(\theta)}{Z\rho} & 0 \\ \frac{\rho^2\cos^2(\theta)-2\rho^2-\cos^2(\theta)}{Z\rho} & \frac{\left(\rho^2-1\right)\sin(2\theta)}{2Z\rho} & 0 \\ \frac{2\rho^2\sin(\theta)}{Z} & -\frac{2\rho^2\cos(\theta)}{Z} & 0 \\ \frac{\left(\rho^2+1\right)\left(2\rho^2\sin^2(\theta)+\cos^2(\theta)\right)}{\rho} & -\frac{\left(2\rho^4+\rho^2-1\right)\sin(2\theta)}{2\rho} & -\frac{\left(\rho^2+1\right)\cos(\theta)}{2} \\ -\frac{\left(2\rho^4+\rho^2-1\right)\sin(2\theta)}{2\rho} & \frac{\left(\rho^2+1\right)\left(2\rho^2\cos^2(\theta)+\sin^2(\theta)\right)}{\rho} & -\frac{\left(\rho^2+1\right)\sin(\theta)}{2} \\ -\frac{\left(\rho^2+1\right)\cos(\theta)}{2} & -\frac{\left(\rho^2+1\right)\sin(\theta)}{2} & 0 \end{bmatrix} \tag{A.19}$$

$$\mathbf{G}_\theta = \begin{bmatrix} \frac{\sin(2\theta)}{Z^2\rho^2} & -\frac{\cos(2\theta)}{Z^2\rho^2} & 0 & \frac{\cos(2\theta)}{Z\rho^2} & \frac{\sin(2\theta)}{Z\rho^2} & 0 \\ -\frac{\cos(2\theta)}{Z^2\rho^2} & -\frac{\sin(2\theta)}{Z^2\rho^2} & 0 & \frac{\sin(2\theta)}{Z\rho^2} & -\frac{\cos(2\theta)}{Z\rho^2} & 0 \\ 0 & 0 & 0 & -\frac{\cos(\theta)}{Z\rho} & -\frac{\sin(\theta)}{Z\rho} & 0 \\ \frac{\cos(2\theta)}{Z\rho^2} & \frac{\sin(2\theta)}{Z\rho^2} & -\frac{\cos(\theta)}{Z\rho} & -\frac{\left(\rho^2+2\right)\sin(2\theta)}{2\rho^2} & \frac{\left(\rho^2+2\right)\cos(2\theta)}{2\rho^2} & \frac{\sin(\theta)}{2\rho} \\ \frac{\sin(2\theta)}{Z\rho^2} & -\frac{\cos(2\theta)}{Z\rho^2} & -\frac{\sin(\theta)}{Z\rho} & \frac{\left(\rho^2+2\right)\cos(2\theta)}{2\rho^2} & \frac{\left(\rho^2+2\right)\sin(2\theta)}{2\rho^2} & -\frac{\cos(\theta)}{2\rho} \\ 0 & 0 & 0 & \frac{\sin(\theta)}{2\rho} & -\frac{\cos(\theta)}{2\rho} & 0 \end{bmatrix} \tag{A.20}$$

$$\mathbf{G}_Z = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -Z & 0 & \frac{Z\rho\cos(\theta)}{2} \\ -1 & 0 & 0 & 0 & -Z & \frac{Z\rho\sin(\theta)}{2} \\ 0 & 0 & 0 & \frac{Z\rho\cos(\theta)}{2} & \frac{Z\rho\sin(\theta)}{2} & 0 \end{bmatrix} \tag{A.21}$$

A.4 Planes

3D Planes can be represented in Cartesian Space by the equation

$$n_x X + n_y Y + n_z Z = d \quad \left(n_x^2 + n_y^2 + n_z^2 = 1\right) \tag{A.22}$$

wherein (X, Y, Z) is any point belonging to the plane, $d > 0$ represents the distance of the plane from the origin of the sensor frame and $\boldsymbol{n}^T = \begin{bmatrix} n_x & n_y & n_z \end{bmatrix}$ is the normal of the plane. There are no parameters in this case, and the feature vector consists in the normal and the value of d.

$$\boldsymbol{s}^T = \begin{bmatrix} n_x & n_y & n_z & d \end{bmatrix} \tag{A.23}$$

$$\mathbf{L}_{\boldsymbol{s}} = \begin{bmatrix} \mathbf{0} & [\boldsymbol{n}]_\times \\ -\boldsymbol{n}^T & \mathbf{0} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & -n_z & n_y \\ 0 & 0 & 0 & n_z & 0 & -n_x \\ 0 & 0 & 0 & -n_y & n_x & 0 \\ -n_x & -n_y & -n_z & 0 & 0 & 0 \end{bmatrix} \tag{A.24}$$

$$\mathbf{G}_{n_x} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{n_y}{2} & \frac{n_z}{2} \\ 0 & 0 & 0 & \frac{n_y}{2} & -n_x & 0 \\ 0 & 0 & 0 & \frac{n_z}{2} & 0 & -n_x \end{bmatrix} \tag{A.25}$$

$$\mathbf{G}_{n_y} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -n_y & \frac{n_x}{2} & 0 \\ 0 & 0 & 0 & \frac{n_x}{2} & 0 & \frac{n_z}{2} \\ 0 & 0 & 0 & 0 & \frac{n_z}{2} & -n_y \end{bmatrix} \tag{A.26}$$

$$\mathbf{G}_{n_z} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -n_z & 0 & \frac{n_x}{2} \\ 0 & 0 & 0 & 0 & -n_z & \frac{n_y}{2} \\ 0 & 0 & 0 & \frac{n_x}{2} & \frac{n_y}{2} & 0 \end{bmatrix} \tag{A.27}$$

$$\mathbf{G}_d = \mathbf{O} \tag{A.28}$$

An alternative, minimal, representation of a 3D plane can exploited, relating the coordinates of normalized points as:

$$Ax + By + C = 1/Z \tag{A.29}$$

It is linked to the previous definition of planes by

$$A = \frac{n_x}{d} \quad B = \frac{n_y}{d} \quad C = \frac{n_z}{d} \tag{A.30}$$

The interest in considering this representation, and in obtaining the corresponding second-order interaction models, is justified by the fact that these values appear as well in the

interaction matrices of planar image moments.

$$\boldsymbol{s}^T = \begin{bmatrix} A & B & C \end{bmatrix} \tag{A.31}$$

$$\mathbf{L}_{\boldsymbol{s}} = \begin{bmatrix} \boldsymbol{s}\boldsymbol{s}^T & -[\boldsymbol{s}]_\times \end{bmatrix} = \begin{bmatrix} A^2 & AB & AC & 0 & -C & B \\ AB & B^2 & BC & C & 0 & -A \\ AC & BC & C^2 & -B & A & 0 \end{bmatrix} \tag{A.32}$$

$$\mathbf{G}_A = \begin{bmatrix} 2A^3 & 2A^2B & 2A^2C & 0 & -AC & AB \\ 2A^2B & 2AB^2 & 2ABC & 0 & -BC & B^2 \\ 2A^2C & 2ABC & 2AC^2 & 0 & -C^2 & BC \\ 0 & 0 & 0 & 0 & \frac{B}{2} & \frac{C}{2} \\ -AC & -BC & -C^2 & \frac{B}{2} & -A & 0 \\ AB & B^2 & BC & \frac{C}{2} & 0 & -A \end{bmatrix} \tag{A.33}$$

$$\mathbf{G}_B = \begin{bmatrix} 2A^2B & 2AB^2 & 2ABC & AC & 0 & -A^2 \\ 2AB^2 & 2B^3 & 2B^2C & BC & 0 & -AB \\ 2ABC & 2B^2C & 2BC^2 & C^2 & 0 & -AC \\ AC & BC & C^2 & -B & \frac{A}{2} & 0 \\ 0 & 0 & 0 & \frac{A}{2} & 0 & \frac{C}{2} \\ -A^2 & -AB & -AC & 0 & \frac{C}{2} & -B \end{bmatrix} \tag{A.34}$$

$$\mathbf{G}_C = \begin{bmatrix} 2A^2C & 2ABC & 2AC^2 & -AB & A^2 & 0 \\ 2ABC & 2B^2C & 2BC^2 & -B^2 & AB & 0 \\ 2AC^2 & 2BC^2 & 2C^3 & -BC & AC & 0 \\ -AB & -B^2 & -BC & -C & 0 & \frac{A}{2} \\ A^2 & AB & AC & 0 & -C & \frac{B}{2} \\ 0 & 0 & 0 & \frac{A}{2} & \frac{B}{2} & 0 \end{bmatrix} \tag{A.35}$$

A.5 Image Moments

We conclude this section by reporting the matrices of the second-order model of planar image moments. Even though they were not used in the work presented in this thesis, they have been included here for reference since extensions of the predictors considered in [FKM20] could be based on these models. For the planar moment $m_{i/j}$, the parameter vector is significantly larger than those found for other features, and includes other image

moments. This leads to second-order models containing several terms. In particular, all entries of $\mathbf{G}_{m_{i/j}}$ can be expressed as a linear combination of image moments whose order varies from $i+j-2$ to $i+j+2$. Therefore, for each entry of (A.39), the linear combination coefficients are listed, and the overall expression is then given. Note that only the upper triangular portion of (A.39) has been reported, since the remaining terms follow from the symmetry of $\mathbf{G}_{m_{i/j}}$. In addition, following the notation adopted in [TC05], the parameter δ is used, whose value equals 1 or 0 respectively when dense or discrete moments are used.

$$\boldsymbol{s} = \begin{bmatrix} m_{i/j} \end{bmatrix} \tag{A.36}$$

$$\boldsymbol{z}^T = \begin{bmatrix} m_{i/j-1} & m_{i/j+1} & m_{i-1/j} & m_{i+1/j} & m_{i-1/j+1} & m_{i+1/j-1} & A & B & C \end{bmatrix} \tag{A.37}$$

$$\mathbf{L}_{\boldsymbol{s}}^T = \begin{bmatrix} -A\delta m_{i/j} - i\left(Am_{i/j} + Bm_{i-1/j+1} + Cm_{i-1/j}\right) \\ -B\delta m_{i/j} - j\left(Am_{i+1/j-1} + Bm_{i/j} + Cm_{i/j-1}\right) \\ -C\delta m_{i/j} + (3\delta + i + j)\left(Am_{i+1/j} + Bm_{i/j+1} + Cm_{i/j}\right) \\ jm_{i/j-1} + m_{i/j+1}\left(3\delta + i + j\right) \\ -im_{i-1/j} + m_{i+1/j}\left(-3\delta - i - j\right) \\ im_{i-1/j+1} - jm_{i+1/j-1} \end{bmatrix} \tag{A.38}$$

$$\mathbf{G}_{m_{i/j}} = \begin{bmatrix} G_{v_x,v_x} & G_{v_x,v_y} & G_{v_x,v_z} & G_{v_x,w_x} & G_{v_x,w_y} & G_{v_x,w_z} \\ & G_{v_y,v_y} & G_{v_y,v_z} & G_{v_y,w_x} & G_{v_y,w_y} & G_{v_y,w_z} \\ & & G_{v_z,v_z} & G_{v_z,w_x} & G_{v_z,w_y} & G_{v_z,w_z} \\ & & & G_{w_x,w_x} & G_{w_x,w_y} & G_{w_x,w_z} \\ & & & & G_{w_y,w_y} & G_{w_y,w_z} \\ & & & & & G_{w_z,w_z} \end{bmatrix} \tag{A.39}$$

$$\begin{aligned} G_{v_x,v_x} = {} & G_{v_x,v_x}^{m_{i/j}} m_{i/j} + G_{v_x,v_x}^{m_{i-1/j}} m_{i-1/j} + G_{v_x,v_x}^{m_{i-1/j+1}} m_{i-1/j+1} \\ & + G_{v_x,v_x}^{m_{i-2/j}} m_{i-2/j} + G_{v_x,v_x}^{m_{i-2/j+1}} m_{i-2/j+1} + G_{v_x,v_x}^{m_{i-2/j+2}} m_{i-2/j+2} \end{aligned} \tag{A.40a}$$

$$G_{v_x,v_x}^{m_{i/j}} = A^2\left(-\delta - i + (\delta + i)^2\right) \tag{A.40b}$$
$$G_{v_x,v_x}^{m_{i-1/j}} = 2ACi\,(\delta + i - 1) \tag{A.40c}$$
$$G_{v_x,v_x}^{m_{i-1/j+1}} = 2ABi\,(\delta + i - 1) \tag{A.40d}$$
$$G_{v_x,v_x}^{m_{i-2/j}} = C^2 i\,(i - 1) \tag{A.40e}$$
$$G_{v_x,v_x}^{m_{i-2/j+1}} = 2BCi\,(i - 1) \tag{A.40f}$$
$$G_{v_x,v_x}^{m_{i-2/j+2}} = B^2 i\,(i - 1) \tag{A.40g}$$

$$\begin{aligned} G_{v_x,v_y} = {} & G_{v_x,v_y}^{m_{i/j}} m_{i/j} + G_{v_x,v_y}^{m_{i/j-1}} m_{i/j-1} + G_{v_x,v_y}^{m_{i-1/j}} m_{i-1/j} \\ & + G_{v_x,v_y}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{v_x,v_y}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{v_x,v_y}^{m_{i-1/j-1}} m_{i-1/j-1} \end{aligned} \tag{A.41a}$$

$$G_{v_x,v_y}^{m_{i/j}} = AB\left(\delta^2 + \delta i + \delta j - \delta + 2ij\right) \tag{A.41b}$$
$$G_{v_x,v_y}^{m_{i/j-1}} = ACj\,(\delta + 2i) \tag{A.41c}$$
$$G_{v_x,v_y}^{m_{i-1/j}} = BCi\,(\delta + 2j) \tag{A.41d}$$
$$G_{v_x,v_y}^{m_{i-1/j+1}} = B^2 i\,(\delta + j) \tag{A.41e}$$
$$G_{v_x,v_y}^{m_{i+1/j-1}} = A^2 j\,(\delta + i) \tag{A.41f}$$
$$G_{v_x,v_y}^{m_{i-1/j-1}} = C^2 ij \tag{A.41g}$$

$$\begin{aligned} G_{v_x,v_z} = {} & G_{v_x,v_z}^{m_{i/j}} m_{i/j} + G_{v_x,v_z}^{m_{i/j+1}} m_{i/j+1} + G_{v_x,v_z}^{m_{i-1/j}} m_{i-1/j} \\ & + G_{v_x,v_z}^{m_{i+1/j}} m_{i+1/j} + G_{v_x,v_z}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{v_x,v_z}^{m_{i-1/j+2}} m_{i-1/j+2} \end{aligned} \tag{A.42a}$$

$$G_{v_x,v_z}^{m_{i/j}} = -AC\left(2\delta^2 + 6\delta i + \delta j + \delta + 2i^2 + 2ij\right) \tag{A.42b}$$
$$G_{v_x,v_z}^{m_{i/j+1}} = -AB\,(\delta + 2i)\,(3\delta + i + j) \tag{A.42c}$$
$$G_{v_x,v_z}^{m_{i-1/j}} = -C^2 i\,(2\delta + i + j) \tag{A.42d}$$
$$G_{v_x,v_z}^{m_{i+1/j}} = -A^2\,(\delta + i)\,(3\delta + i + j) \tag{A.42e}$$
$$G_{v_x,v_z}^{m_{i-1/j+1}} = -BCi\,(5\delta + 2i + 2j) \tag{A.42f}$$
$$G_{v_x,v_z}^{m_{i-1/j+2}} = -B^2 i\,(3\delta + i + j) \tag{A.42g}$$

$$G_{v_x,w_x} = G_{v_x,w_x}^{m_{i/j-1}} m_{i/j-1} + G_{v_x,w_x}^{m_{i/j+1}} m_{i/j+1} + G_{v_x,w_x}^{m_{i-1/j}} m_{i-1/j} + G_{v_x,w_x}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{v_x,w_x}^{m_{i-1/j-1}} m_{i-1/j-1} + G_{v_x,w_x}^{m_{i-1/j+2}} m_{i-1/j+2} \quad \text{(A.43a)}$$

$$G_{v_x,w_x}^{m_{i/j-1}} = -Aj\,(\delta + i) \quad \text{(A.43b)}$$

$$G_{v_x,w_x}^{m_{i/j+1}} = -A\,(\delta + i)\,(3\delta + i + j) \quad \text{(A.43c)}$$

$$G_{v_x,w_x}^{m_{i-1/j}} = -Bij \quad \text{(A.43d)}$$

$$G_{v_x,w_x}^{m_{i-1/j+1}} = -Ci\,(3\delta + i + j) \quad \text{(A.43e)}$$

$$G_{v_x,w_x}^{m_{i-1/j-1}} = -Cij \quad \text{(A.43f)}$$

$$G_{v_x,w_x}^{m_{i-1/j+2}} = -Bi\,(3\delta + i + j) \quad \text{(A.43g)}$$

$$G_{v_x,w_y} = G_{v_x,w_y}^{m_{i/j}} m_{i/j} + G_{v_x,w_y}^{m_{i/j+1}} m_{i/j+1} + G_{v_x,w_y}^{m_{i-1/j}} m_{i-1/j} + G_{v_x,w_y}^{m_{i+1/j}} m_{i+1/j} + G_{v_x,w_y}^{m_{i-2/j}} m_{i-2/j} + G_{v_x,w_y}^{m_{i-2/j+1}} m_{i-2/j+1} \quad \text{(A.44a)}$$

$$G_{v_x,w_y}^{m_{i/j}} = C\left(3\delta i + 3\delta + i^2 + ij + i + j\right) \quad \text{(A.44b)}$$

$$G_{v_x,w_y}^{m_{i/j+1}} = B\,(i+1)\,(3\delta + i + j) \quad \text{(A.44c)}$$

$$G_{v_x,w_y}^{m_{i-1/j}} = Ai\,(\delta + i - 1) \quad \text{(A.44d)}$$

$$G_{v_x,w_y}^{m_{i+1/j}} = A\,(\delta + i + 1)\,(3\delta + i + j) \quad \text{(A.44e)}$$

$$G_{v_x,w_y}^{m_{i-2/j}} = Ci\,(i-1) \quad \text{(A.44f)}$$

$$G_{v_x,w_y}^{m_{i-2/j+1}} = Bi\,(i-1) \quad \text{(A.44g)}$$

$$G_{v_x,w_z} = G_{v_x,w_z}^{m_{i/j}} m_{i/j} + G_{v_x,w_z}^{m_{i/j-1}} m_{i/j-1} + G_{v_x,w_z}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{v_x,w_z}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{v_x,w_z}^{m_{i-2/j+1}} m_{i-2/j+1} + G_{v_x,w_z}^{m_{i-2/j+2}} m_{i-2/j+2} \quad \text{(A.45a)}$$

$$G_{v_x,w_z}^{m_{i/j}} = Bj\,(i+1) \tag{A.45b}$$
$$G_{v_x,w_z}^{m_{i/j-1}} = Cj\,(i+1) \tag{A.45c}$$
$$G_{v_x,w_z}^{m_{i-1/j+1}} = Ai\,(-\delta - i + 1) \tag{A.45d}$$
$$G_{v_x,w_z}^{m_{i+1/j-1}} = Aj\,(\delta + i + 1) \tag{A.45e}$$
$$G_{v_x,w_z}^{m_{i-2/j+1}} = Ci\,(1 - i) \tag{A.45f}$$
$$G_{v_x,w_z}^{m_{i-2/j+2}} = Bi\,(1 - i) \tag{A.45g}$$

$$\begin{aligned} G_{v_y,v_y} = {} & G_{v_y,v_y}^{m_{i/j}} m_{i/j} + G_{v_y,v_y}^{m_{i/j-1}} m_{i/j-1} + G_{v_y,v_y}^{m_{i+1/j-1}} m_{i+1/j-1} \\ & + G_{v_y,v_y}^{m_{i/j-2}} m_{i/j-2} + G_{v_y,v_y}^{m_{i+1/j-2}} m_{i+1/j-2} + G_{v_y,v_y}^{m_{i+2/j-2}} m_{i+2/j-2} \end{aligned} \tag{A.46a}$$

$$G_{v_y,v_y}^{m_{i/j}} = B^2\left(-\delta - j + (\delta + j)^2\right) \tag{A.46b}$$
$$G_{v_y,v_y}^{m_{i/j-1}} = 2BCj\,(\delta + j - 1) \tag{A.46c}$$
$$G_{v_y,v_y}^{m_{i+1/j-1}} = 2ABj\,(\delta + j - 1) \tag{A.46d}$$
$$G_{v_y,v_y}^{m_{i/j-2}} = C^2 j\,(j - 1) \tag{A.46e}$$
$$G_{v_y,v_y}^{m_{i+1/j-2}} = 2ACj\,(j - 1) \tag{A.46f}$$
$$G_{v_y,v_y}^{m_{i+2/j-2}} = A^2 j\,(j - 1) \tag{A.46g}$$

$$\begin{aligned} G_{v_y,v_z} = {} & G_{v_y,v_z}^{m_{i/j}} m_{i/j} + G_{v_y,v_z}^{m_{i/j-1}} m_{i/j-1} + G_{v_y,v_z}^{m_{i/j+1}} m_{i/j+1} \\ & + G_{v_y,v_z}^{m_{i+1/j}} m_{i+1/j} + G_{v_y,v_z}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{v_y,v_z}^{m_{i+2/j-1}} m_{i+2/j-1} \end{aligned} \tag{A.47a}$$

$$G_{v_y,v_z}^{m_{i/j}} = -BC\left(2\delta^2 + \delta i + 6\delta j + \delta + 2ij + 2j^2\right) \tag{A.47b}$$
$$G_{v_y,v_z}^{m_{i/j-1}} = -C^2 j\,(2\delta + i + j) \tag{A.47c}$$
$$G_{v_y,v_z}^{m_{i/j+1}} = -B^2\,(\delta + j)\,(3\delta + i + j) \tag{A.47d}$$
$$G_{v_y,v_z}^{m_{i+1/j}} = -AB\,(\delta + 2j)\,(3\delta + i + j) \tag{A.47e}$$
$$G_{v_y,v_z}^{m_{i+1/j-1}} = -ACj\,(5\delta + 2i + 2j) \tag{A.47f}$$
$$G_{v_y,v_z}^{m_{i+2/j-1}} = -A^2 j\,(3\delta + i + j) \tag{A.47g}$$

$$G_{v_y,w_x} = G_{v_y,w_x}^{m_{i/j}} m_{i/j} + G_{v_y,w_x}^{m_{i/j-1}} m_{i/j-1} + G_{v_y,w_x}^{m_{i/j+1}} m_{i/j+1} + G_{v_y,w_x}^{m_{i+1/j}} m_{i+1/j} + G_{v_y,w_x}^{m_{i/j-2}} m_{i/j-2} + G_{v_y,w_x}^{m_{i+1/j-2}} m_{i+1/j-2} \quad \text{(A.48a)}$$

$$G_{v_y,w_x}^{m_{i/j}} = -C\left(3\delta j + 3\delta + ij + i + j^2 + j\right) \quad \text{(A.48b)}$$

$$G_{v_y,w_x}^{m_{i/j-1}} = Bj\left(-\delta - j + 1\right) \quad \text{(A.48c)}$$

$$G_{v_y,w_x}^{m_{i/j+1}} = -B\left(\delta + j + 1\right)\left(3\delta + i + j\right) \quad \text{(A.48d)}$$

$$G_{v_y,w_x}^{m_{i+1/j}} = -A\left(j + 1\right)\left(3\delta + i + j\right) \quad \text{(A.48e)}$$

$$G_{v_y,w_x}^{m_{i/j-2}} = Cj\left(1 - j\right) \quad \text{(A.48f)}$$

$$G_{v_y,w_x}^{m_{i+1/j-2}} = Aj\left(1 - j\right) \quad \text{(A.48g)}$$

$$G_{v_y,w_y} = G_{v_y,w_y}^{m_{i/j-1}} m_{i/j-1} + G_{v_y,w_y}^{m_{i-1/j}} m_{i-1/j} + G_{v_y,w_y}^{m_{i+1/j}} m_{i+1/j} + G_{v_y,w_y}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{v_y,w_y}^{m_{i-1/j-1}} m_{i-1/j-1} + G_{v_y,w_y}^{m_{i+2/j-1}} m_{i+2/j-1} \quad \text{(A.49a)}$$

$$G_{v_y,w_y}^{m_{i/j-1}} = Aij \quad \text{(A.49b)}$$

$$G_{v_y,w_y}^{m_{i-1/j}} = Bi\left(\delta + j\right) \quad \text{(A.49c)}$$

$$G_{v_y,w_y}^{m_{i+1/j}} = B\left(\delta + j\right)\left(3\delta + i + j\right) \quad \text{(A.49d)}$$

$$G_{v_y,w_y}^{m_{i+1/j-1}} = Cj\left(3\delta + i + j\right) \quad \text{(A.49e)}$$

$$G_{v_y,w_y}^{m_{i-1/j-1}} = Cij \quad \text{(A.49f)}$$

$$G_{v_y,w_y}^{m_{i+2/j-1}} = Aj\left(3\delta + i + j\right) \quad \text{(A.49g)}$$

$$G_{v_y,w_z} = G_{v_y,w_z}^{m_{i/j}} m_{i/j} + G_{v_y,w_z}^{m_{i-1/j}} m_{i-1/j} + G_{v_y,w_z}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{v_y,w_z}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{v_y,w_z}^{m_{i+1/j-2}} m_{i+1/j-2} + G_{v_y,w_z}^{m_{i+2/j-2}} m_{i+2/j-2} \quad \text{(A.50a)}$$

$$G_{v_y,w_z}^{m_{i/j}} = -Ai\,(j+1) \tag{A.50b}$$
$$G_{v_y,w_z}^{m_{i-1/j}} = -Ci\,(j+1) \tag{A.50c}$$
$$G_{v_y,w_z}^{m_{i-1/j+1}} = -Bi\,(\delta+j+1) \tag{A.50d}$$
$$G_{v_y,w_z}^{m_{i+1/j-1}} = Bj\,(\delta+j-1) \tag{A.50e}$$
$$G_{v_y,w_z}^{m_{i+1/j-2}} = Cj\,(j-1) \tag{A.50f}$$
$$G_{v_y,w_z}^{m_{i+2/j-2}} = Aj\,(j-1) \tag{A.50g}$$

$$\begin{aligned} G_{v_z,v_z} = G_{v_z,v_z}^{m_{i/j}} m_{i/j} + G_{v_z,v_z}^{m_{i/j+1}} m_{i/j+1} + G_{v_z,v_z}^{m_{i+1/j}} m_{i+1/j} \\ + G_{v_z,v_z}^{m_{i/j+2}} m_{i/j+2} + G_{v_z,v_z}^{m_{i+1/j+1}} m_{i+1/j+1} + G_{v_z,v_z}^{m_{i+2/j}} m_{i+2/j} \end{aligned} \tag{A.51a}$$

$$G_{v_z,v_z}^{m_{i/j}} = C^2\,(2\delta+i+j)\,(2\delta+i+j+1) \tag{A.51b}$$
$$G_{v_z,v_z}^{m_{i/j+1}} = 2BC\,(3\delta+i+j)\,(2\delta+i+j+1) \tag{A.51c}$$
$$G_{v_z,v_z}^{m_{i+1/j}} = 2AC\,(3\delta+i+j)\,(2\delta+i+j+1) \tag{A.51d}$$
$$G_{v_z,v_z}^{m_{i/j+2}} = B^2\,(3\delta+i+j)\,(3\delta+i+j+1) \tag{A.51e}$$
$$G_{v_z,v_z}^{m_{i+1/j+1}} = 2AB\,(3\delta+i+j)\,(3\delta+i+j+1) \tag{A.51f}$$
$$G_{v_z,v_z}^{m_{i+2/j}} = A^2\,(3\delta+i+j)\,(3\delta+i+j+1) \tag{A.51g}$$

$$\begin{aligned} G_{v_z,w_x} = G_{v_z,w_x}^{m_{i/j}} m_{i/j} + G_{v_z,w_x}^{m_{i/j-1}} m_{i/j-1} + G_{v_z,w_x}^{m_{i/j+1}} m_{i/j+1} \\ + G_{v_z,w_x}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{v_z,w_x}^{m_{i/j+2}} m_{i/j+2} + G_{v_z,w_x}^{m_{i+1/j+1}} m_{i+1/j+1} \end{aligned} \tag{A.52a}$$

$$G_{v_z,w_x}^{m_{i/j}} = Bj\,(3\delta+i+j-1) \tag{A.52b}$$
$$G_{v_z,w_x}^{m_{i/j-1}} = Cj\,(2\delta+i+j-1) \tag{A.52c}$$
$$G_{v_z,w_x}^{m_{i/j+1}} = C\,(3\delta+i+j)\,(2\delta+i+j+1) \tag{A.52d}$$
$$G_{v_z,w_x}^{m_{i+1/j-1}} = Aj\,(3\delta+i+j-1) \tag{A.52e}$$
$$G_{v_z,w_x}^{m_{i/j+2}} = B\,(3\delta+i+j)\,(3\delta+i+j+1) \tag{A.52f}$$
$$G_{v_z,w_x}^{m_{i+1/j+1}} = A\,(3\delta+i+j)\,(3\delta+i+j+1) \tag{A.52g}$$

$$G_{v_z,w_y} = G_{v_z,w_y}^{m_{i/j}} m_{i/j} + G_{v_z,w_y}^{m_{i-1/j}} m_{i-1/j} + G_{v_z,w_y}^{m_{i+1/j}} m_{i+1/j} + G_{v_z,w_y}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{v_z,w_y}^{m_{i+1/j+1}} m_{i+1/j+1} + G_{v_z,w_y}^{m_{i+2/j}} m_{i+2/j} \quad \text{(A.53a)}$$

$$G_{v_z,w_y}^{m_{i/j}} = Ai\left(-3\delta - i - j + 1\right) \quad \text{(A.53b)}$$

$$G_{v_z,w_y}^{m_{i-1/j}} = Ci\left(-2\delta - i - j + 1\right) \quad \text{(A.53c)}$$

$$G_{v_z,w_y}^{m_{i+1/j}} = -C\left(3\delta + i + j\right)\left(2\delta + i + j + 1\right) \quad \text{(A.53d)}$$

$$G_{v_z,w_y}^{m_{i-1/j+1}} = Bi\left(-3\delta - i - j + 1\right) \quad \text{(A.53e)}$$

$$G_{v_z,w_y}^{m_{i+1/j+1}} = -B\left(3\delta + i + j\right)\left(3\delta + i + j + 1\right) \quad \text{(A.53f)}$$

$$G_{v_z,w_y}^{m_{i+2/j}} = -A\left(3\delta + i + j\right)\left(3\delta + i + j + 1\right) \quad \text{(A.53g)}$$

$$G_{v_z,w_z} = G_{v_z,w_z}^{m_{i/j+1}} m_{i/j+1} + G_{v_z,w_z}^{m_{i+1/j}} m_{i+1/j} + G_{v_z,w_z}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{v_z,w_z}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{v_z,w_z}^{m_{i-1/j+2}} m_{i-1/j+2} + G_{v_z,w_z}^{m_{i+2/j-1}} m_{i+2/j-1} \quad \text{(A.54a)}$$

$$G_{v_z,w_z}^{m_{i/j+1}} = Ai\left(3\delta + i + j\right) \quad \text{(A.54b)}$$

$$G_{v_z,w_z}^{m_{i+1/j}} = -Bj\left(3\delta + i + j\right) \quad \text{(A.54c)}$$

$$G_{v_z,w_z}^{m_{i-1/j+1}} = Ci\left(2\delta + i + j\right) \quad \text{(A.54d)}$$

$$G_{v_z,w_z}^{m_{i+1/j-1}} = -Cj\left(2\delta + i + j\right) \quad \text{(A.54e)}$$

$$G_{v_z,w_z}^{m_{i-1/j+2}} = Bi\left(3\delta + i + j\right) \quad \text{(A.54f)}$$

$$G_{v_z,w_z}^{m_{i+2/j-1}} = -Aj\left(3\delta + i + j\right) \quad \text{(A.54g)}$$

$$G_{w_x,w_x} = G_{w_x,w_x}^{m_{i/j}} m_{i/j} + G_{w_x,w_x}^{m_{i/j-2}} m_{i/j-2} + G_{w_x,w_x}^{m_{i/j+2}} m_{i/j+2} \quad \text{(A.55a)}$$

$$G_{w_x,w_x}^{m_{i/j}} = j\left(3\delta + i + j - 1\right) + \left(j + 1\right)\left(3\delta + i + j\right) \quad \text{(A.55b)}$$

$$G_{w_x,w_x}^{m_{i/j-2}} = j\left(j - 1\right) \quad \text{(A.55c)}$$

$$G_{w_x,w_x}^{m_{i/j+2}} = \left(3\delta + i + j\right)\left(3\delta + i + j + 1\right) \quad \text{(A.55d)}$$

$$G_{w_x,w_y} = G_{w_x,w_y}^{m_{i-1/j+1}} m_{i-1/j+1} + G_{w_x,w_y}^{m_{i+1/j-1}} m_{i+1/j-1} + G_{w_x,w_y}^{m_{i-1/j-1}} m_{i-1/j-1} + G_{w_x,w_y}^{m_{i+1/j+1}} m_{i+1/j+1} \quad \text{(A.56a)}$$

$$G_{w_x,w_y}^{m_{i-1/j+1}} = \frac{i\,(-6\delta - 2i - 2j + 1)}{2} \quad \text{(A.56b)}$$

$$G_{w_x,w_y}^{m_{i+1/j-1}} = \frac{j\,(-6\delta - 2i - 2j + 1)}{2} \quad \text{(A.56c)}$$

$$G_{w_x,w_y}^{m_{i-1/j-1}} = -ij \quad \text{(A.56d)}$$

$$G_{w_x,w_y}^{m_{i+1/j+1}} = -\,(3\delta + i + j)\,(3\delta + i + j + 1) \quad \text{(A.56e)}$$

$$G_{w_x,w_z} = G_{w_x,w_z}^{m_{i-1/j}} m_{i-1/j} + G_{w_x,w_z}^{m_{i+1/j}} m_{i+1/j} + G_{w_x,w_z}^{m_{i+1/j-2}} m_{i+1/j-2} + G_{w_x,w_z}^{m_{i-1/j+2}} m_{i-1/j+2} \quad \text{(A.57a)}$$

$$G_{w_x,w_z}^{m_{i-1/j}} = i\left(j + \frac{1}{2}\right) \quad \text{(A.57b)}$$

$$G_{w_x,w_z}^{m_{i+1/j}} = -\frac{(2j+1)\,(3\delta + i + j)}{2} \quad \text{(A.57c)}$$

$$G_{w_x,w_z}^{m_{i+1/j-2}} = j\,(1-j) \quad \text{(A.57d)}$$

$$G_{w_x,w_z}^{m_{i-1/j+2}} = i\,(3\delta + i + j) \quad \text{(A.57e)}$$

$$G_{w_y,w_y} = G_{w_y,w_y}^{m_{i/j}} m_{i/j} + G_{w_y,w_y}^{m_{i-2/j}} m_{i-2/j} + G_{w_y,w_y}^{m_{i+2/j}} m_{i+2/j} \quad \text{(A.58a)}$$

$$G_{w_y,w_y}^{m_{i/j}} = i\,(3\delta + i + j - 1) + (i+1)\,(3\delta + i + j) \quad \text{(A.58b)}$$

$$G_{w_y,w_y}^{m_{i-2/j}} = i\,(i-1) \quad \text{(A.58c)}$$

$$G_{w_y,w_y}^{m_{i+2/j}} = (3\delta + i + j)\,(3\delta + i + j + 1) \quad \text{(A.58d)}$$

$$G_{w_y,w_z} = G_{w_y,w_z}^{m_{i/j-1}} m_{i/j-1} + G_{w_y,w_z}^{m_{i/j+1}} m_{i/j+1} + G_{w_y,w_z}^{m_{i-2/j+1}} m_{i-2/j+1} + G_{w_y,w_z}^{m_{i+2/j-1}} m_{i+2/j-1} \quad \text{(A.59a)}$$

$$G_{w_y,w_z}^{m_{i/j-1}} = j\left(i + \frac{1}{2}\right) \quad \text{(A.59b)}$$

$$G_{w_y,w_z}^{m_{i/j+1}} = -\frac{(2i+1)(3\delta + i + j)}{2} \quad \text{(A.59c)}$$

$$G_{w_y,w_z}^{m_{i-2/j+1}} = i(1-i) \quad \text{(A.59d)}$$

$$G_{w_y,w_z}^{m_{i+2/j-1}} = j(3\delta + i + j) \quad \text{(A.59e)}$$

$$G_{w_z,w_z} = G_{w_z,w_z}^{m_{i/j}} m_{i/j} + G_{w_z,w_z}^{m_{i-2/j+2}} m_{i-2/j+2} + G_{w_z,w_z}^{m_{i+2/j-2}} m_{i+2/j-2} \quad \text{(A.60a)}$$

$$G_{w_z,w_z}^{m_{i/j}} = -2ij - i - j \quad \text{(A.60b)}$$

$$G_{w_z,w_z}^{m_{i-2/j+2}} = i(i-1) \quad \text{(A.60c)}$$

$$G_{w_z,w_z}^{m_{i+2/j-2}} = j(j-1) \quad \text{(A.60d)}$$

Appendix B

Optimization Algorithms for Model Predictive Control

This Appendix details the main algorithms that were used to solve the optimization problems arising in MPC schemes. In particular, three algorithms are presented: a Quadratic Programming method based on active set, a SQP for the minimization of generic nonlinear problems and a Gauss-Newton algorithm for the solution of constrained nonlinear least squares minimization.

B.1 Standard Problem & KKT Conditions

For the remainder of this Appendix, minimization problems are assumed to be written in the following standard form:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{B.1a}$$

subject to:

$$\boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{0} \tag{B.1b}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \boldsymbol{0} \tag{B.1c}$$

wherein $\boldsymbol{x} \in \mathbb{R}^n$ is the decision vector, while f, $\boldsymbol{g}$ and $\boldsymbol{h}$ are twice-differentiable functions that model the objective function and the inequality/equality constraints.

So as to determine the optima of the minimization, a fundamental quantity that is defined for the problem is the Lagrangian function:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\mu}^T \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x}) \tag{B.2}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are known as "Lagrange multipliers" or "dual variables" associated to the

problem (B.1). This function is fundamental for the solution of constrained minimization, since it can be used to determine a lower bound on the optima of the original problem. In particular, denoting with $\mathcal{D}$ the set of points in $\mathbb{R}^n$ that satisfy the constraints (B.1b) and (B.1c), the following holds:

$$\min_{\boldsymbol{x}\in\mathcal{D}} f(\boldsymbol{x}) \geq \inf_{\boldsymbol{x}\in\mathcal{D}} \mathcal{L}(\boldsymbol{x},\boldsymbol{\mu},\boldsymbol{\lambda}) \quad \forall \boldsymbol{\mu}\geq \mathbf{0}, \boldsymbol{\lambda} \tag{B.3}$$

Furthermore, the Lagrangian can be used to define four necessary conditions that a triplet $\boldsymbol{x}^\star, \boldsymbol{\mu}^\star, \boldsymbol{\lambda}^\star$ must satisfy to be qualified as an optimum of (B.1). These conditions are known as Karush-Kuhn-Tucker (KKT) conditions, and write as:

$$\text{stationarity} \qquad \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^\star} + \boldsymbol{\mu}^{\star T} \left.\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^\star} + \boldsymbol{\lambda}^{\star T} \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^\star} = \mathbf{0} \tag{B.4a}$$

$$\text{primal feasibility} \qquad \begin{aligned} &\boldsymbol{g}(\boldsymbol{x}^\star) \leq \mathbf{0} \\ &\boldsymbol{h}(\boldsymbol{x}^\star) = \mathbf{0} \end{aligned} \tag{B.4b}$$

$$\text{dual feasibility} \qquad \boldsymbol{\mu}^\star \geq \mathbf{0} \tag{B.4c}$$

$$\text{complementary slackness} \qquad \boldsymbol{\mu}^\star \boldsymbol{g}(\boldsymbol{x}^\star) = \mathbf{0} \tag{B.4d}$$

The first condition can be written synthetically as $\left.\frac{\partial \mathcal{L}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}^\star,\boldsymbol{\mu}^\star,\boldsymbol{\lambda}^\star} = \mathbf{0}$, and has a geometric interpretation that will be explained in the following, while the second condition trivially states that if an optimum $\boldsymbol{x}^\star$ exists, it must belong to $\mathcal{D}$, *i.e.*, the set of feasible points for the problem. The meaning of the third and fourth conditions will also be explained below.

To understand the first condition, let's focus for a moment on the case of problems featuring equality constraints only. Condition (B.4a) states that, for $\boldsymbol{x}^\star$ to be an optimum, the gradient of the objective function must be expressible as a linear combination of the gradients of the constraints, *i.e.*,

$$\nabla f \in \operatorname{span}(\nabla \boldsymbol{h}) \tag{B.5}$$

Or, equivalently, ∇f must not have any component that lies in the tangent space of the constraints. An intuitive explanation is provided in Figure B.1(a): since the gradient of the objective has a component that is tangent to the constraint, the depicted point cannot be a constrained optimum. In fact, one could imagine to perform a small motion $\delta\boldsymbol{x}_\parallel$ in a direction that is tangent to the constraint curve. In this case, the value of the constraint

is not expected to change:

$$h(\boldsymbol{x}^\star + \delta\boldsymbol{x}_\parallel) \simeq h(\boldsymbol{x}^\star) + \delta\boldsymbol{x}_\parallel^T \nabla h(\boldsymbol{x}^\star) \tag{B.6}$$

Since $\boldsymbol{x}^\star$ is a feasible point, it follows that $h(\boldsymbol{x}^\star + \delta\boldsymbol{x}) \simeq h(\boldsymbol{x}^\star) = \mathbf{0}$, and thus the new point $\boldsymbol{x}^\star + \delta\boldsymbol{x}_\parallel$ should be feasible as well. However, using a similar argument, the value of the objective is expected to change of a quantity $\delta\boldsymbol{x}_\parallel^T \nabla f(\boldsymbol{x}^\star)$, which cannot be null since $\nabla f \nparallel \nabla h$. This shows that there is always the possibility to locally change the value of f, and in particular to decrease it, which is equivalent to say that $\boldsymbol{x}^\star$ cannot be a minimum. On the contrary, if ∇f has no component in the tangent space of the constraints as in Figure B.1(b), then $\delta\boldsymbol{x}_\parallel^T \nabla f(\boldsymbol{x}^\star) = 0$ and thus $\boldsymbol{x}^\star$ is a candidate optimum point. Note that one can also interpret the condition in terms of forces acting on a point mass. In practice $-\nabla f$ acts as a force pulling the point towards unconstrained minima, while $-\lambda\nabla h$ as the reaction force that is necessary to keep the mass "attached" to the constraint curve. In this sense, a minimum corresponds to a stable static equilibrium point for the mass.

The previous explanation can be extended to inequality constraints, allowing to understand the remaining KKT conditions. First of all, note that if a point $\boldsymbol{x}^\star$ does not lie on the boundary defined by an inequality constraint, then there exists a whole neighborhood of $\boldsymbol{x}^\star$ which satisfies the constraint. In other words, at least locally, the constraint is said to be inactive, and can thus be ignored. Using the analogy of gradients acting as forces on a mass, the force produced by an inequality constraint behaves as a ground reaction, the "ground" being the locus where the inequality turns into an equality. While the mass is "in the air", the ground cannot exert any reaction, and thus the associated Lagrange multiplier must be zero. On the contrary, when the mass reaches contact with the ground, the force activates, explaining the last of the KKT conditions. In addition, sticking to the analogy of ground reaction forces, an active inequality constraint can exert a force only in the direction opposite to the ground. As an example, in Figure B.1(c), the constraint consists in keeping $\boldsymbol{x}$ over the depicted curve. At the represented point, the "objective force" $-\nabla f$ pulls the mass down, and thus the constraint can react with a force in the direction $-\nabla g$, whose magnitude is regulated by the *positive* scalar μ. On the contrary, in Figure B.1(d) the constraint requires the point to stay below the curve. To keep the point exactly on the curve, the constraint force would require to pull the mass upward, which requires $\mu < 0$. However, this is not possible: the ground reaction can only push the mass, which leads to (B.4c).

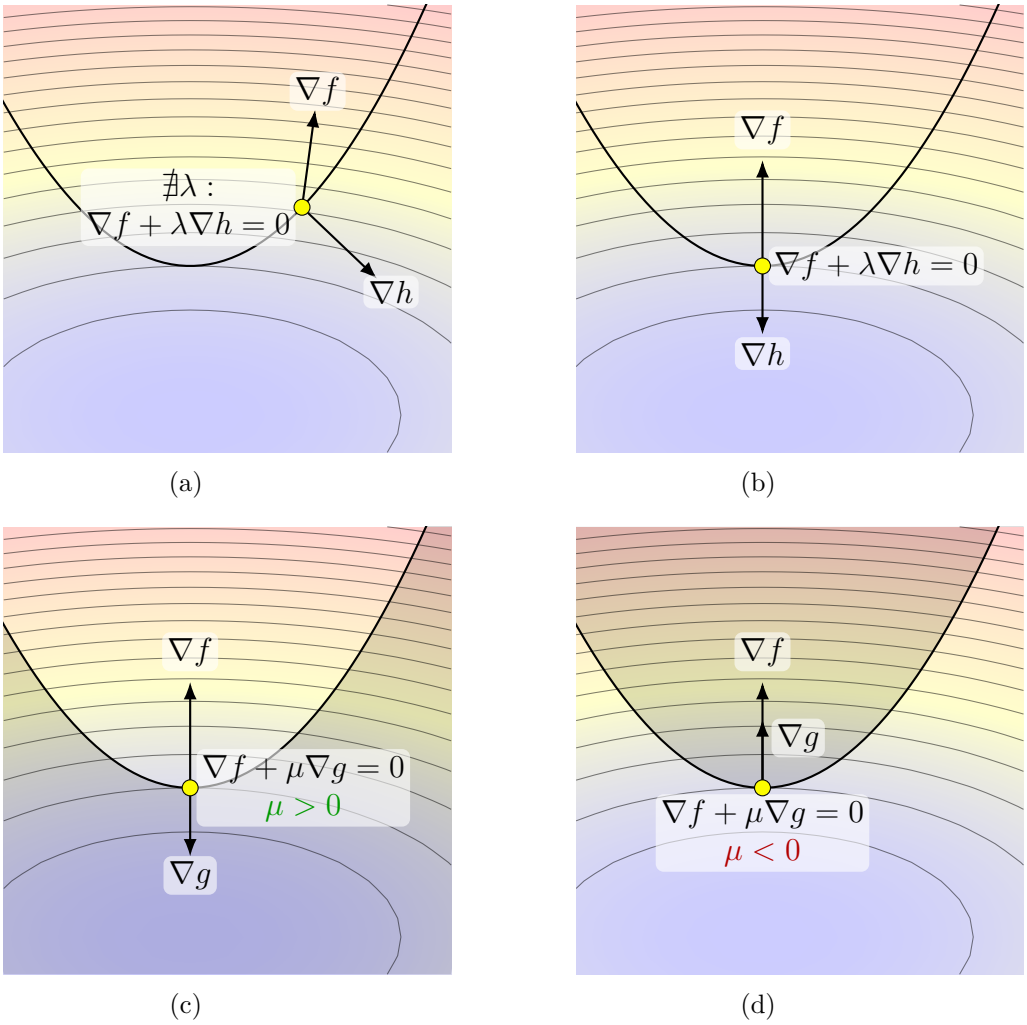


Figure B.1 – Pictorial explanation of the KKT conditions. The solid black line represents an equality ((a) & (b)) or inequality (((c) & (d))) constraint. A point on the constraint boundary is an optimum only if the gradient of the objective function can be obtained as a linear combination of the constraint gradients. Additionally, the multipliers must be positive in the case of inequality constraints.

B.2 Active Set Quadratic Programming

A quadratic minimization problem is a special case of nonlinear optimization in which the objective function is linear-quadratic in the decision vector. In addition, unless otherwise specified, it is almost always assumed that constraints are linear. According to this definition, the goal of Quadratic Programming is to find the solution to the following special form of (B.1):

$$\min_{\boldsymbol{x}} \frac{1}{2}\boldsymbol{x}^T\mathbf{Q}\boldsymbol{x} - \boldsymbol{r}^T\boldsymbol{x} \tag{B.7a}$$

subject to:

$$\mathbf{C}\boldsymbol{x} \leq \boldsymbol{d} \tag{B.7b}$$

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b} \tag{B.7c}$$

wherein the squared matrix $\mathbf{Q}$ is assumed to be symmetric and positive semidefinite.

One strategy that can be employed to solve this special class of problems is the one known as *Active Set Quadratic Programming*. It is an iterative algorithm which works by taking steps within $\mathcal{D}$ while activating or deactivating inequality constraints as needed and, as it will be shown in the following, it is closely related to the KTT conditions illustrated before. In each iteration, given a valid point $\boldsymbol{x}_{i-1}$, the procedure mainly consists in the following steps:

1. Determine a step direction $\boldsymbol{z}_i$.
2. Evaluate a step length α_i.
3. Perform the step: $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \alpha_i\boldsymbol{z}_i$.
4. Activate or deactivate constraints as needed.

B.2.1 Solution Algorithm

Determine a step direction $\boldsymbol{z}_i$.

In order to find the step direction $\boldsymbol{z}_i$, a change of variable is performed in the original quadratic problem, introducing the variable $\boldsymbol{z}$ as in $\boldsymbol{x} = \boldsymbol{x}_{i-1} + \boldsymbol{z}$. After substitution into (B.7), the new (quadratic) sub-problem is obtained:

$$\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T\mathbf{Q}\boldsymbol{z} - \boldsymbol{s}_i^T\boldsymbol{z} \tag{B.8a}$$

subject to:

$$\mathbf{C}\boldsymbol{z} \leq \boldsymbol{e}_i \tag{B.8b}$$

$$\mathbf{A}\boldsymbol{z} = \mathbf{0} \tag{B.8c}$$

wherein in the objective $\boldsymbol{s}_i \doteq \boldsymbol{r} - \mathbf{Q}\boldsymbol{x}_{i-1}$ and the constant term $\frac{1}{2}\boldsymbol{x}_{i-1}^T\mathbf{Q}\boldsymbol{x}_i - \boldsymbol{r}^T\boldsymbol{x}_{i-1}$ has been discarded, since it does not change the solution. Regarding the constraints, $\boldsymbol{e}_i \doteq \boldsymbol{d} - \mathbf{C}\boldsymbol{x}_i$, while the fact that $\mathbf{A}\boldsymbol{x}_{i-1} = \boldsymbol{b}$ ($\boldsymbol{x}_{i-1} \in \mathcal{D}$) has been used to obtain the equality constraints.

The KKT condition of stationarity and primal feasibility for this problem can be written altogether as:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{C}^T & \mathbf{A}^T \\ \mathbf{C} & & \\ \mathbf{A} & & \end{bmatrix} \begin{bmatrix} \boldsymbol{z}^\star \\ \boldsymbol{\mu}^\star \\ \boldsymbol{\lambda}^\star \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_i \\ \leq \boldsymbol{e}_i \\ \mathbf{0} \end{bmatrix} \tag{B.9}$$

wherein trivial zeros have been removed for readability. As stated in the previous section, inequality constraints can be divided into inactive ones (in which the inequality is strict) and actives (which locally act as equality constraints). One can thus partition, at the current point $\boldsymbol{x}_{i-1}$, the inequality constraints as:

$$\begin{aligned} \mathbf{C}_{a,i}\boldsymbol{z} &= \mathbf{e}_{a,i} \\ \mathbf{C}_{na,i}\boldsymbol{z} &< \mathbf{e}_{na,i} \end{aligned} \tag{B.10}$$

Note that also the associated Lagrange multipliers are to be partitioned in a similar way. In addition, since the constraints are all smooth, the partition will be constant at least in a neighborhood of $\boldsymbol{z} = \mathbf{0}$. Using this fact, and considering condition (B.4d) (complementary slackness), it directly follows that $\boldsymbol{\mu}_{a,i}^\star = \mathbf{0}$, ultimately leading to a simplification of (B.9) to:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{C}_{a,i}^T & \mathbf{A}^T \\ \mathbf{C}_{a,i} & & \\ \mathbf{A} & & \end{bmatrix} \begin{bmatrix} \boldsymbol{z}^\star \\ \boldsymbol{\mu}_a^\star \\ \boldsymbol{\lambda}^\star \end{bmatrix} = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{e}_{a,i} \\ \mathbf{0} \end{bmatrix} \tag{B.11}$$

Solving this linear system of equations leads to the optimal solution of the sub-problem, with the primal optimum $\boldsymbol{z}^\star$ being the desired step direction $\boldsymbol{z}_i$.

Determine a step length α_i

The solution $\boldsymbol{z}_i$ found at the previous step is valid only if the partition of active and passive inequality constraints does not change. As mentioned above, it is true at least in a neighborhood of $\boldsymbol{z} = \boldsymbol{0}$, but the question is whether it holds also at $\boldsymbol{z}_i$. Finding an answer to this question is equivalent to determine if there exist a value $\alpha \in [0, 1]$ such that the set of active constraints changes at $\boldsymbol{z} = \alpha \boldsymbol{z}_i$.

To solve the current problem, each of the currently **inactive** constraints is written as $\boldsymbol{c}_j^T \boldsymbol{z} \leq e_j$. While "moving" from the origin to $\boldsymbol{z}_i$, the current configuration $\boldsymbol{z}$ could either approach the boundary of an inactive constraint or not. The second case corresponds to situations in which $\boldsymbol{c}_j^T \boldsymbol{z}_i$ does not move towards the value e_j, which can happen only if $\boldsymbol{c}_j^T \boldsymbol{z}_i \leq 0$. In fact, by definition, $e_j \doteq d_j - \boldsymbol{c}_j^T \boldsymbol{x}_i$. Moreover, by assumption $\boldsymbol{x}_i$ is a feasible point such that $\boldsymbol{c}_j^T \boldsymbol{x}_i \leq d_j$, which implies $0 \leq d_j - \boldsymbol{c}_j^T \boldsymbol{x}_i$, *i.e.*, $e_j \geq 0$. Since in this step we are considering $\boldsymbol{z} = \alpha \boldsymbol{z}_i$, with positive α, then $\boldsymbol{c}_j^T \boldsymbol{z}_i \leq 0$ implies $\alpha \boldsymbol{c}_j^T \boldsymbol{z}_i \leq 0 \leq e_j$.

One case has yet to be checked, *i.e.*, when $\boldsymbol{c}_j^T \boldsymbol{z}_i > 0$. Recall that the problem at hand is to determine if at some point $\boldsymbol{z} = \alpha \boldsymbol{z}_i$ there are any constraints that activate. By definition, for each j such that $\boldsymbol{c}_j^T \boldsymbol{z}_i > 0$ we must ensure $\alpha \boldsymbol{c}_j^T \boldsymbol{z}_i \leq e_j$, which corresponds to $\alpha \leq \frac{e_j}{\boldsymbol{c}_j^T \boldsymbol{z}_i}$. To satisfy this system of inequalities, it suffices to take the smallest of the right hand sides. In addition, since $\alpha \in [0, 1]$, the desired value is finally:

$$\alpha_i = \min \left(1, \min \left\{ \frac{e_j}{\boldsymbol{c}_j^T \boldsymbol{z}_i} \text{ s.t. } \boldsymbol{c}_j^T \boldsymbol{z}_i \geq 0 \right\} \right) \tag{B.12}$$

Note however that there exist a "degenerate case", so as to speak, in which $\boldsymbol{z}_i = \boldsymbol{0}$. In this case, looking for a step size using the outlined procedure is pointless, and $\alpha_i = 1$ can directly be selected. In practice, any value in the interval $[0, 1]$ could be chosen. However, using 1 allows the following step to be performed "safely" without having to treat this case separately.

Activating or deactivating constraints

The new iterate $\boldsymbol{x}_i$ can now be formed as $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \alpha_i \boldsymbol{z}_i$. By construction, it is a valid point and is such that $f(\boldsymbol{x}_i) \leq f(\boldsymbol{x}_{i-1})$. Before continuing with the new iteration, it is necessary to check the current active set of constraints.

If $\alpha_i < 1$, then the current iterate $\boldsymbol{x}_i$ has reached the boundary of one of the inequality constraints. In other words, it is has now become active, and it is necessary to include it in the active set.

If instead $\alpha_i = 1$, then the "full step" did not cause any constraint activation. In principle this could mean that the optimum has been found. However, there is still one condition to check: according to the dual feasibility (B.4c), all Lagrange multipliers associated to inequality constraints must be non-negative at the constrained optimum. Therefore, if any entry of $\boldsymbol{\mu}_a$ is negative, the associated constraint must be removed and a new iteration has to be performed. Otherwise, if $\boldsymbol{\mu}_a \geq \mathbf{0}$, the algorithm can terminate, returning $\boldsymbol{x}^\star = \boldsymbol{x}_i$.

Note that, from a practical point of view, the active set must be modified only at this step. This is necessary because, when defining the partition (B.10), one must not include constraints that have just deactivated. Based only on their value, these constraints would evaluate as 0 and erroneously selected as part of the active set.

B.2.2 Practical Notes

Initialization

The detailed strategy based on the active set iterates by stepping within the valid domain $\mathcal{D}$. While this is enforced by construction for the values $\boldsymbol{x}_i$ with $i > 0$, it is crucial to find a valid point $\boldsymbol{x}_0$ to start the algorithm. One way to do it would consist in solving a simple linear optimization problem similar to the following:

$$\boldsymbol{x}_0 = \arg\min_{\boldsymbol{x}} 0 \tag{B.13a}$$

subject to

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b} \tag{B.13b}$$

$$\mathbf{C}\boldsymbol{x} \leq \boldsymbol{d} \tag{B.13c}$$

If instead the user was able to provide an initial guess $\boldsymbol{x}_u$, it could be desirable to initialize the algorithm using the following procedure:

$$\boldsymbol{x}_0 = \boldsymbol{x}_u + \|\boldsymbol{z}\|_\infty \tag{B.14a}$$

subject to

$$\mathbf{A}\boldsymbol{z} = \mathbf{0} \tag{B.14b}$$

$$\mathbf{C}\boldsymbol{z} \leq \boldsymbol{d} - \mathbf{C}\boldsymbol{x}_u \tag{B.14c}$$

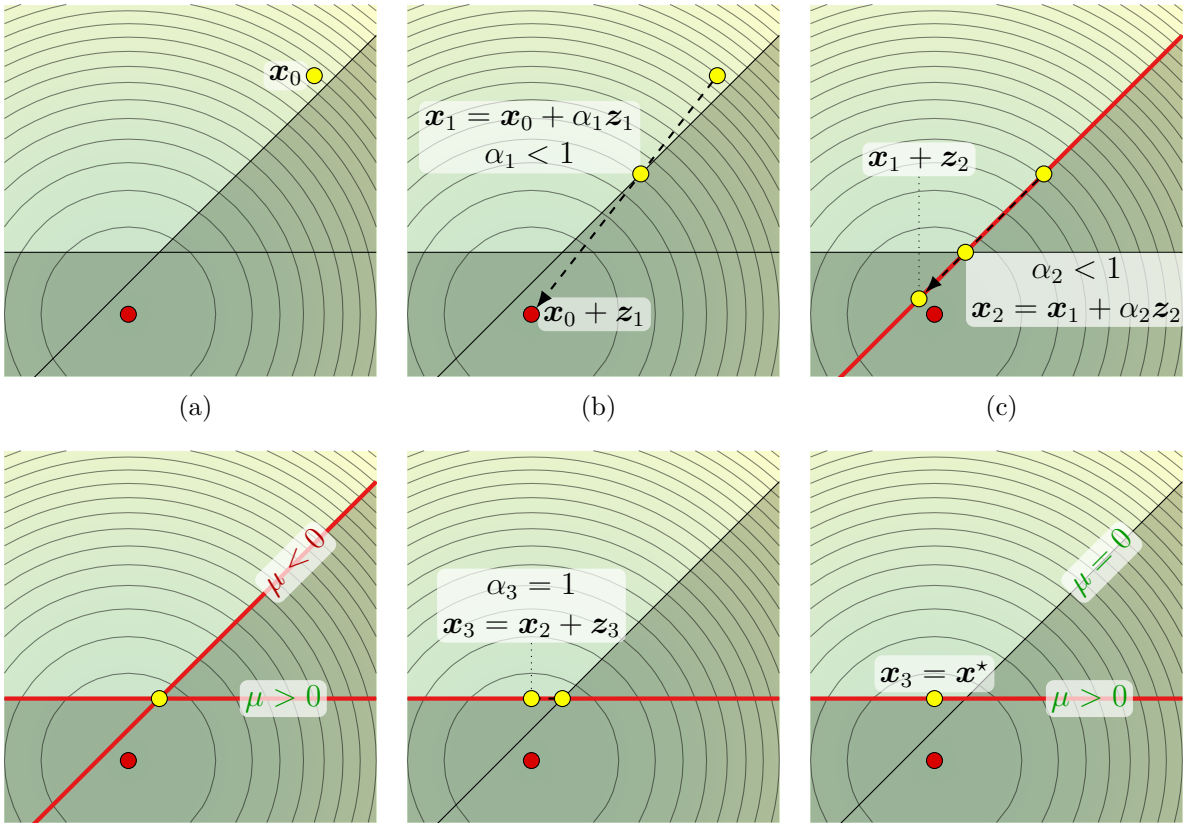


Figure B.2 – Different phases during the solution of a constrained quadratic optimization. The dark regions represent the invalid domain according to two linear constraints. The minimum value of the objective is found at the red point, which violates the constraints. The optimization starts at the yellow point shown in (a), and moves within the valid domain until the constrained optimum is reached in (f). Along the process, some constraints need to be activated (they are highlighted in red) and deactivated.

which can be useful especially if $\mathbf{Q}$ is only positive semidefinite and thus admits multiple solutions. In fact, it initializes $\boldsymbol{x}_0$ so that it is as close as possible to the user-supplied value $\boldsymbol{x}_u$, and hopefully returns a final solution which is in turn closer to it.

Finally, if similar quadratic problems have to be solved in succession, a last option is available, consisting in using the last active set from a previous solution to define the seed iterate $\boldsymbol{x}_0$. In practice, by denoting as $\mathbf{C}_{a,w}$ and $\boldsymbol{d}_{a,w}$ the matrix and vector corresponding to the last active constraints, one possibility is to use

$$\boldsymbol{x}_0 = \begin{bmatrix} \mathbf{C}_{a,w} \\ \mathbf{A} \end{bmatrix}^{+} \begin{bmatrix} \boldsymbol{d}_{a,w} \\ \boldsymbol{b} \end{bmatrix} \tag{B.15}$$

as starting point. Of course, the value should be tested to confirm that it does in fact correspond to a feasible point according the remaining constraints, and, if this is not the case, one of the solutions above can be used instead.

Finally, this warm start can be used not only to evaluate an initial solution $\boldsymbol{x}_0$, but also to initialize the active set. This can greatly reduce the computation times, *e.g.*, when QP is used for the solution of linear MPC problems – in which most of the constraints are likely to remain active between successive iterations – or when the active set algorithm is used for the solution of a series of sub-problems, which is true in the case of SQP and Gauss-Newton methods presented later.

Efficient handling of equality constraints

Since the equality constraints (B.7c) of the original problem do not change across subsequent iterations, it is possible to perform a simple substitution that reduces the dimension of the sub-problems to be solved at each iteration. In particular, the decision vector can be parameterized as:

$$\boldsymbol{x} = \boldsymbol{x}_{eq} + \mathbf{Z}_{\mathbf{A}} \boldsymbol{y} \tag{B.16}$$

wherein $\boldsymbol{x}_{eq}$ is any vector that solves the equality constraints, *i.e.*, $\mathbf{A}\boldsymbol{x}_{eq} = \boldsymbol{b}$, while $\mathbf{Z}_{\mathbf{A}}$ is a basis of the kernel of $\mathbf{A}$ and $\boldsymbol{y}$ is a new vector of free variables. With this parameterization, problem (B.7c) can be formulated in the lower-dimensional variable $\boldsymbol{y}$ as:

$$\min_{\boldsymbol{y}} \frac{1}{2} \boldsymbol{y}^T \mathbf{Z}_{\mathbf{A}}^T \mathbf{Q} \mathbf{Z}_{\mathbf{A}} \boldsymbol{y} - \left(\boldsymbol{r} - \boldsymbol{x}_{eq}\mathbf{Q}\right)^T \mathbf{Z}_{\mathbf{A}} \boldsymbol{y} \tag{B.17a}$$

subject to:

$$\mathbf{C}\mathbf{Z}_{\mathbf{A}}\boldsymbol{y} \leq \boldsymbol{d} - \mathbf{C}\boldsymbol{x}_{eq} \tag{B.17b}$$

Solving the quadratic sub-problem

While detailing the overall active set algorithm, it was simply stated that at each iteration the step $\boldsymbol{z}_i$ defined by the quadratic sub-problem is given by the solution of the following linear system:

$$\begin{bmatrix} \mathbf{Q} & \mathbf{C}_{a,i}^T & \mathbf{A}^T \\ \mathbf{C}_{a,i} & & \\ \mathbf{A} & & \end{bmatrix} \begin{bmatrix} \boldsymbol{z}^\star \\ \boldsymbol{\mu}_a^\star \\ \boldsymbol{\lambda}^\star \end{bmatrix} = \begin{bmatrix} \boldsymbol{s} \\ \boldsymbol{e}_{a,i} \\ \mathbf{0} \end{bmatrix} \tag{B.18}$$

without providing hints on how to solve it. In practice, there are two main possible solutions, the first one being to perform a Cholesky decomposition with pivoting (also known as *LDLT* decomposition) on the overall matrix, and to solve the problem using that factorization.

An alternative approach is similar to the one detailed in the previous section, and consists in firstly finding a particular solution $\boldsymbol{z}_{eq}$ to the sub-system

$$\begin{bmatrix} \mathbf{C}_{a,i} \\ \mathbf{A} \end{bmatrix} \boldsymbol{z} = \begin{bmatrix} \boldsymbol{e}_{a,i} \\ \mathbf{0} \end{bmatrix} \tag{B.19}$$

and then to perform the change of variable defined by:

$$\boldsymbol{z} = \boldsymbol{z}_{eq} + \mathbf{Z}_{eq}\boldsymbol{y} \qquad \text{s.t.} \quad \begin{bmatrix} \mathbf{C}_{a,i} \\ \mathbf{A} \end{bmatrix} \mathbf{Z}_{eq} = \mathbf{O} \tag{B.20}$$

This new parameterization can then be injected in the original sub-problem and a solution can be obtained in the lower-dimensional vector $\boldsymbol{y}$.

B.3 Sequential Quadratic Programming

As already mentioned in Chapter 3, Sequential Quadratic Programming is an optimization algorithm that can be used to solve generic non-linearly constrained nonlinear minimization problems. The principle of this method is to locally approximate the problem at hand with a quadratic minimization, which can be solved using any QP algorithm.

The obtained solution allows to update the current guess for the minimum point of the problem, and the whole procedure is sequentially repeated until termination criteria are met.

In particular, at each iteration i the value $\boldsymbol{x}_{i-1}$ is assumed as an approximate solution of the problem, and, similarly to what was done in the iterations of the active set QP algorithm, the goal is to find a step direction $\boldsymbol{z}_i$ and a step length α_i to produce a new iterate in the form $\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \alpha_i \boldsymbol{z}_i$.

First of all, the objective function f is approximated in a neighborhood of $\boldsymbol{x}_{i-1}$ via a second-order Taylor expansion as:

$$f(\boldsymbol{x}_{i-1} + \boldsymbol{z}) \simeq f(\boldsymbol{x}_{i-1}) + \frac{\partial f}{\partial \boldsymbol{x}}\boldsymbol{z} + \frac{1}{2}\boldsymbol{z}^T \frac{\partial^2 f}{\partial \boldsymbol{x}^2}\boldsymbol{z} \tag{B.21}$$

wherein all derivatives are evaluated at $\boldsymbol{x}_{i-1}$. In addition, with some abuse of notation, $\frac{\partial^2 f}{\partial \boldsymbol{x}^2}$ has been used to compactly represent the hessian of f, whose correct expression should be written as $\frac{\partial}{\partial \boldsymbol{x}}\left(\frac{\partial f}{\partial \boldsymbol{x}}\right)^T$.

On top of the quadratic approximation of the objective function, a minimization problem in the step variable $\boldsymbol{z}$ can be formulated. Constraints are approximated as well, using a linear-affine first order expansion centered at $\boldsymbol{x}_{i-1}$. The i-th sub-problem thus writes as:

$$\min_{\boldsymbol{z}} \frac{1}{2}\boldsymbol{z}^T \frac{\partial^2 f}{\partial \boldsymbol{x}^2}\boldsymbol{z} + \frac{\partial f}{\partial \boldsymbol{x}}\boldsymbol{z} \tag{B.22a}$$

subject to:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\boldsymbol{z} \leq -\boldsymbol{g}(\boldsymbol{x}_{i-1}) \tag{B.22b}$$

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\boldsymbol{z} = -\boldsymbol{h}(\boldsymbol{x}_{i-1}) \tag{B.22c}$$

The problem is clearly quadratic in $\boldsymbol{z}$, and, assuming the hessian to be at least positive semidefinite, can be solved by standard QP algorithms to produce the step $\boldsymbol{z}_i$. This fact is crucial, and might be false in general. For this reason, an approximation of the hessian is often used [Pow78; Kra88], which is ensured to always be positive definite and also depends only on first-order derivatives of the objective.

The second phase in the SQP algorithm consists in determining the step length $\alpha_i \in [0, 1]$ to be used to update the current iterate. In practice, a cost (or merit) function is

used to assign a score to new points. An example of such cost is given by [Han77]:

$$m(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_j r_j \max(g_j(\boldsymbol{x}), 0) + \sum_j s_j \left| h_j(\boldsymbol{x}) \right| \tag{B.23}$$

wherein g_j and h_j are the j-th inequality and equality constraint. The values r_j and s_j are instead weights that penalize constraints infringement. They can be simply assigned the value of the j-th dual variable, *i.e.*, $r_j = \mu_i$ and $s_i = \lambda_i$. Alternatively, [Pow78] proposes to use weights that depend also on their past values r_j^{past} and s_j^{past}:

$$\begin{aligned} r_j &= \max\left(\mu_j, \frac{\mu_j + r_j^{past}}{2}\right) \\ s_j &= \max\left(\left|\lambda_j\right|, \frac{\left|\lambda_j\right| + s_j^{past}}{2}\right) \end{aligned} \tag{B.24}$$

The step size is finally obtained by a line search in the following form:

$$\alpha_i = \arg\min_{\alpha \in [0,1]} m(\boldsymbol{x}_{i-1} + \alpha \boldsymbol{z}_i) \tag{B.25}$$

and the new iterate $\boldsymbol{x}_i$ is set to $\boldsymbol{x}_{i-1} + \alpha_i \boldsymbol{z}_i$.

The procedure is repeated until one or more termination criteria are met, common choices in this sense being to set a time limit or a maximum number of iterations. In addition, it is possible to determine if a constrained local minimum has been attained by checking the norm of the update step $\alpha_i \boldsymbol{z}_i$ or by checking the KKT conditions.

One last important remark is related to handling of linear constraints. In the formulation of (B.22) it was assumed that $\boldsymbol{g}$ and $\boldsymbol{h}$ were generic, possibly but not necessarily nonlinear, functions. However, it is more efficient to make an explicit distinction between linear and nonlinear constraints. In fact, no linearization is needed when formulating the quadratic sub-problem (B.22). Furthermore, and possibly more importantly, they do not contribute to the value of the merit function. In fact, assuming that $\boldsymbol{x}_{i-1}$ is in the valid domain defined by linear (in)equalities, it follows that the point $\boldsymbol{x}_{i-1} + \boldsymbol{z}_i$ will also be valid since the QP subroutine always enforces the validity of subsequent iterates. Finally, since domains defined by linear equalities and inequalities are convex, it follows that any point in the form $\boldsymbol{x}_{i-1} + \alpha \boldsymbol{z}_i$ is necessarily valid for $\alpha \in [0, 1]$, and therefore they do not participate to (B.23).

B.4 Iterative Gauss-Newton Method

This method can be used to solve nonlinear optimization problems in the following special form:

$$\min_{\boldsymbol{x}} \|\boldsymbol{\phi}(\boldsymbol{x})\|^2 \tag{B.26a}$$

subject to:

$$\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \tag{B.26b}$$

$$\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \tag{B.26c}$$

wherein $\boldsymbol{\phi}$ is a generic vector-valued function. The method can be viewed as a special case of the SQP algorithm, as it will be shown below, and works by formulating the quadratic sub-problem using a slightly different methodology. In practice, a first-order approximation of $\boldsymbol{\phi}$ around $\boldsymbol{x}_{i-1}$ is performed and substituted in the objective:

$$\|\boldsymbol{\phi}(\boldsymbol{x}_{i-1} + \boldsymbol{z})\|^2 \simeq \left\| \boldsymbol{\phi}(\boldsymbol{x}_{i-1}) + \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} \boldsymbol{z} \right\|^2 \tag{B.27}$$

After expansion, and by approximating the constraints as in the general SQP method, the minimization sub-problem is obtained:

$$\min_{\boldsymbol{z}} \boldsymbol{z}^T \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} \boldsymbol{z} + 2\boldsymbol{\phi}^T \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} \boldsymbol{z} \tag{B.28a}$$

subject to:

$$\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}} \boldsymbol{z} \leq -\boldsymbol{g}(\boldsymbol{x}_{i-1}) \tag{B.28b}$$

$$\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \boldsymbol{z} = -\boldsymbol{h}(\boldsymbol{x}_{i-1}) \tag{B.28c}$$

To see that it indeed corresponds to a SQP method, the derivatives of the objective

can be evaluated:

$$\frac{\partial}{\partial \boldsymbol{x}} \left\| \boldsymbol{\phi} \right\|^2 = 2\boldsymbol{\phi}^T \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} \tag{B.29a}$$

$$\frac{\partial^2}{\partial \boldsymbol{x}^2} \left\| \boldsymbol{\phi} \right\|^2 = 2\frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}}^T \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{x}} + 2\sum_j \phi_j \frac{\partial^2 \phi_j}{\partial \boldsymbol{x}^2} \tag{B.29b}$$

wherein ϕ_j represents the j-th component of $\boldsymbol{\phi}$. Comparing (B.22a) and (B.28a) reveals that this Gauss-Newton method corresponds to a SQP one in which the hessian is approximated by discarding the second summation appearing in (B.29b). Note also that the approximated hessian is by definition at least positive semidefinite, and thus it can be used in alternative to the approximation method proposed in [Pow78].

BIBLIOGRAPHY

[AA+06] Omar Ait-Aider, Nicolas Andreff, Jean Marc Lavest, and Philippe Martinet, « Simultaneous object pose and velocity computation using a single view from a rolling shutter camera », *in*: *Lecture Notes in Computer Science*, vol. 3952 LNCS, Springer, Berlin, Heidelberg, 2006, pp. 56–68.

[AC09] Guillaume Allibert and Estelle Courtial, « What can prediction bring to image-based visual servoing? », *in*: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis (USA), 2009, pp. 5210–5215.

[ACC10] Guillaume Allibert, Estelle Courtial, and François Chaumette, « Predictive control for constrained image-based visual servoing », *in*: *IEEE Transactions on Robotics* 26.*5* (2010), pp. 933–939.

[ACT07] Guillaume Allibert, Estelle Courtial, and Youssoufi Touré, « A flat Model Predictive Controller for trajectory tracking in Image Based Visual Servoing », *in*: *IFAC Proceedings Volumes*, vol. 40, 12, Elsevier, 2007, pp. 993–998.

[ACT08] Guillaume Allibert, Estelle Courtial, and Youssoufi Touré, « Visual predictive control for manipulators with catadioptric camera », *in*: *IEEE International Conference on Robotics and Automation*, Pasadena (CA), 2008, pp. 510–515.

[AF05] Benedetto Allotta and Duccio Fioravanti, « 3D Motion planning for image-based visual servoing tasks », *in*: *IEEE International Conference on Robotics and Automation*, vol. 2005, IEEE, 2005, pp. 2173–2178.

[Agr+17] Don Joven Agravante, Giovanni Claudio, Fabien Spindler, and François Chaumette, « Visual Servoing in an Optimization Framework for the Whole-Body Control of Humanoid Robots », *in*: *IEEE Robotics and Automation Letters* 2.*2* (2017), pp. 608–615.

[Ala06] Mazen Alamir, *Stabilization of nonlinear systems using receding-horizon control schemes: a parametrized approach for fast systems*, vol. 339, Lecture Notes in Control and Information Sciences, Springer, 2006.

[Ala13] Mazen Alamir, *A pragmatic story of model predictive control: self-contained algorithms and case-studies*, CreateSpace Independent Publishing Platform, 2013.

[BC95] F. Bensalah and F. Chaumette, « Compensation of abrupt motion changes in target tracking by visual servoing », *in*: *Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, vol. 1, IEEE Comput. Soc. Press, 1995, pp. 181–187.

[BCL14] Adrian Burlacu, Cosmin Copot, and Corneliu Lazar, « Predictive control architecture for real-time image moments based servoing of robot manipulators », *in*: *Journal of Intelligent Manufacturing*, vol. 25, 5, Elsevier, 2014, pp. 1125–1134.

[Bra+14] Christian Brandli, Raphael Berner, Minhao Yang, Shih Chii Liu, and Tobi Delbruck, « A 240 × 180 130 dB 3 μs latency global shutter spatiotemporal vision sensor », *in*: *IEEE Journal of Solid-State Circuits* 49.*10* (2014), pp. 2333–2341.

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool, « SURF: Speeded up robust features », *in*: *European conference on computer vision*, vol. 3951 LNCS, Springer, Berlin, Heidelberg, 2006, pp. 404–417.

[Car+19] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard, « The Pinocchio C ++ library – A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives », *in*: *IEEE International Symposium on System Integrations (SII)*, Paris, 2019.

[CBL11] Cosmin Copot, Adrian Burlacu, and Corneliu Lazar, « Visual predictive control architecture based on image moments for manipulator robots », *in*: *IEEE International Symposium on Industrial Electronics*, 2011, pp. 963–968.

[CG96] Peter I. Corke and Malcolm C. Good, « Dynamic effects in visual closed-loop systems », *in*: *IEEE Transactions on Robotics and Automation* 12.*5* (1996), pp. 671–683.

[CH01] Peter I. Corke and Seth A. Hutchinson, « A new partitioned approach to image-based visual servo control », *in*: *IEEE Transactions on Robotics and Automation* 17.*4* (2001), pp. 507–515.

[CH06] François Chaumette and Seth Hutchinson, « Visual servo control. I. Basic approaches », *in*: *IEEE Robotics and Automation Magazine* 13.*4* (2006), pp. 82–90.

[CH07] François Chaumette and Seth Hutchinson, « Visual servo control. II. Advanced approaches », *in*: *IEEE Robotics and Automation Magazine* 14.*1* (2007), pp. 109–118.

[Cha04] François Chaumette, « Image Moments: A General and Useful Set of Features for Visual Servoing », *in*: *IEEE Transactions on Robotics* 20.*4* (2004), pp. 713–723.

[Cha98] François Chaumette, « Potential problems of stability and convergence in image-based and position-based visual servoing », *in*: *The confluence of vision and control*, Springer, London, 1998, pp. 66–78.

[CLB12] C Copot, C Lazar, and A Burlacu, « Predictive control of nonlinear visual servoing systems using image moments », *in*: *IET control theory & applications* 6.*10* (2012), pp. 1486–1496.

[CM01] F. Chaumette and Marchand Éric, « A redundancy-based iterative approach for avoiding joint limits: Application to visual servoing », *in*: *IEEE Transactions on Robotics and Automation* 17.*5* (2001), pp. 719–730.

[CM18] Justin Carpentier and Nicolas Mansard, « Analytical Derivatives of Rigid Body Dynamics Algorithms », *in*: *Robotics: Science and Systems (RSS 2018)*, 2018.

[CMT87] David W Clarke, Coorous Mohtadi, and P S Tuffs, « Generalized predictive control—Part I. The basic algorithm », *in*: *Automatica* 23.*2* (1987), pp. 137–148.

[Com+06] Andrew I. Comport, Eric Marchand, Muriel Pressigout, and François Chaumette, « Real-time markerless tracking for augmented reality: The virtual visual servoing framework », *in*: *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, 4, 2006, pp. 615–628.

[CRE93] François Chaumette, Patrick Rives, and Bernard Espiau, « Classification and realization of the different vision-based tasks », *in*: *Visual Servoing: Real-Time Control of Robot Manipulators Based on Visual Sensory Feedback*, World Scientific, 1993, pp. 199–228.

[CSC16] Giovanni Claudio, Fabien Spindler, and Francois Chaumette, « Vision-based manipulation with the humanoid robot Romeo », *in*: *IEEE-RAS International Conference on Humanoid Robots*, IEEE Computer Society, 2016, pp. 286–293.

[Dah+08] Redwan Dahmouche, Omar Ait-Aider, Nicolas Andreff, and Youcef Mezouar, « High-speed pose and velocity measurement from vision », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, 2008, pp. 107–112.

[Dah+09] Redwan Dahmouche, Nicolas Andreff, Youcef Mezouar, and Philippe Martinet, « 3D pose and velocity visual tracking based on sequential region of interest acquisition », *in*: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, IEEE, 2009, pp. 5426–5431.

[Dah+10] Redwan Dahmouche, Nicolas Andreff, Youcef Mezouar, and Philippe Martinet, « Efficient high-speed vision-based computed torque control of the orthoglide parallel robot », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 644–649.

[Dah+12] Redwan Dahmouche, Nicolas Andreff, Youcef Mezouar, Omar Ait-Aider, and Philippe Martinet, « Dynamic visual servoing from sequential regions of interest acquisition », *in*: *International Journal of Robotics Research* 31.*4* (2012), pp. 520–537.

[DD95] Daniel F. Dementhon and Larry S. Davis, « Model-based object pose in 25 lines of code », *in*: *International Journal of Computer Vision* 15.*1-2* (1995), pp. 123–141.

[Die+06] M. Diehl, H. G. Bock, H. Diedam, and P. B. Wieber, « Fast direct multiple shooting algorithms for optimal robot control », *in*: *Lecture Notes in Control and Information Sciences*, vol. 340, Springer, Berlin, Heidelberg, 2006, pp. 65–93.

[DO18] A. De Luca and G. Oriolo, « Efficient Dynamic Resolution of Robot Redundancy », *in*: *1990 American Control Conference*, IEEE, 2018, pp. 221–227.

[DOG07] Alessandro De Luca, Giuseppe Oriolo, and Paolo Robuffo Giordano, « Online estimation of feature depth for image-based visual servoing schemes », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 2823–2828.

[ECR92] B. Espiau, F. Chaumette, and Patrick Rives, « A new approach to visual servoing in robotics », *in*: *IEEE Transactions on Robotics and Automation* 8.*3* (1992), pp. 313–326.

[EMW14] Adrien Escande, Nicolas Mansard, and Pierre Brice Wieber, « Hierarchical quadratic programming: Fast online humanoid-robot motion generation », *in*: *International Journal of Robotics Research* 33.*7* (2014), pp. 1006–1028.

[FA02] Rolf Findeisen and Frank Allgöwer, « An introduction to nonlinear model predictive control », *in*: *21st Benelux meeting on systems and control*, vol. 11, Technische Universiteit Eindhoven Veldhoven Eindhoven, The Netherlands, 2002, pp. 119–141.

[FCM00] Grégory Flandin, François Chaumette, and Eric Marchand, « Eye-in-hand/eye-to-hand cooperation for visual servoing », *in*: *Proceedings-IEEE International Conference on Robotics and Automation* 3 (2000), pp. 2741–2746.

[FKM19] Franco Fusco, Olivier Kermorgant, and Philippe Martinet, « A Comparison of Visual Servoing from Features Velocity and Acceleration Interaction Models », *in*: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, 2019, pp. 4447–4452.

[FKM20] Franco Fusco, Olivier Kermorgant, and Philippe Martinet, « Integrating Features Acceleration in Visual Predictive Control », *in*: *IEEE Robotics and Automation Letters* 5.*4* (2020), pp. 5197–5204.

[GD03] Jacques A. Gangloff and Michel F. De Mathelin, « High-speed visual servoing of a 6-d.o.f. manipulator using multivariable predictive control », *in*: *Advanced Robotics* 17.*10* (2003), pp. 993–1021.

[GDA98] Jacques A. Gangloff, Michel De Mathelin, and Gabriel Abba, « 6 DOF high speed dynamic visual servoing using GPC controllers », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 3, IEEE, 1998, pp. 2008–2013.

[Gif+18] Markus Giftthaler, Michael Neunert, Markus Stauble, Jonas Buchli, and Moritz Diehl, « A Family of Iterative Gauss-Newton Shooting Methods for Nonlinear Optimal Control », *in*: *IEEE International Conference on Intelligent Robots and Systems*, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 6903–6910.

[Gin+05] Romuald Ginhoux, Jacques Gangloff, Michel de Mathelin, Luc Soler, Mara M. Arenas Sanchez, and Jacques Marescaux, « Active filtering of physiological motion in robotized surgery using predictive control », *in*: *IEEE Transactions on Robotics* 21.*1* (2005), pp. 67–79.

[GJO10] Gaël Guennebaud, Benoît Jacob, and Others, *Eigen v3*, http://eigen.tuxfamily.org, 2010.

[GM00] J.A. Gangloff and M.F. De Mathelin, « High speed visual servoing of a 6 DOF manipulator using MIMO predictive control », *in*: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)* 4 (2000), pp. 3751–3756.

[Han77] S. P. Han, « A globally convergent method for nonlinear programming », *in*: *Journal of Optimization Theory and Applications* 22.*3* (1977), pp. 297–309.

[HO99] 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* 1.*May* (1999), pp. 618–623.

[HZ03] Richard Hartley and Andrew Zisserman, *Multiple view geometry in computer vision*, Cambridge university press, 2003.

[IO05] Masami Iwatsuki and Norimitsu Okiyama, « A new formulation of visual servoing based on cylindrical coordinate system », *in*: *IEEE Transactions on Robotics* 21.*2* (2005), pp. 266–273.

[Joh] Steven G. Johnson, *The NLopt nonlinear-optimization package*.

[KC14] Olivier Kermorgant and François Chaumette, « Dealing with constraints in sensor-based robot control », *in*: *IEEE Transactions on Robotics* 30.*1* (2014), pp. 244–257.

[KD04] Wisama Khalil and Etienne Dombre, *Modeling, identification and control of robots*, Butterworth-Heinemann, 2004.

[Kel+00] Rafaël Kelly, Ricardo Carelli, Oscar Nasisi, Benjamin Kuchen, and Fernando Reyes, « Stable visual servoing of camera-in-hand robotic systems », *in*: *IEEE/ASME Transactions on Mechatronics* 5.*1* (2000), pp. 39–48.

[KLW11] Oussama Kanoun, Florent Lamiraux, and Pierre Brice Wieber, « Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task », *in*: *IEEE Transactions on Robotics* 27.*4* (2011), pp. 785–792.

[Kra88] Dieter Kraft, « A software package for sequential quadratic programming », *in*: *DFVLR-FB 88-28* (1988).

[Kra94] Dieter Kraft, « Algorithm 733: TOMP–Fortran modules for optimal control calculations », *in*: *ACM Transactions on Mathematical Software (TOMS)* 20.*3* (1994), pp. 262–281.

[KX12] Mohammad Keshmiri and Wen Fang Xie, « Augmented Imaged Based Visual Servoing controller for a 6 DOF manipulator using acceleration command », *in*: *Proceedings of the IEEE Conference on Decision and Control*, IEEE, 2012, pp. 556–561.

[KXM14] Mohammad Keshmiri, Wen Fang Xie, and Abolfazl Mohebbi, « Augmented image-based visual servoing of a manipulator using acceleration command », *in*: *IEEE Transactions on Industrial Electronics* 61.*10* (2014), pp. 5444–5452.

[LB08] Corneliu Lazar and Adrian Burlacu, « Modeling of visual servo open-loop for robot manipulators », *in*: *IEEE International Symposium on Industrial Electronics*, IEEE, 2008, pp. 1154–1159.

[LB09] C. Lazar and A. Burlacu, « Visual servoing of robot manipulators using model-based predictive control », *in*: *IEEE International Conference on Industrial Informatics (INDIN)*, IEEE, 2009, pp. 690–695.

[LBC11] C Lazar, A Burlacu, and C Copot, « Predictive control architecture for visual servoing of robot manipulators », *in*: *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 44, 1 PART 1, Elsevier, 2011, pp. 9464–9469.

[Lea19] Joao Rui Leal, *CppADCodeGen: C++ Algorithmic Differentiation with Source Code Generation*, 2019.

[LM04] J. T. Lapresté and Y. Mezouar, « A hessian approach to visual servoing », *in*: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 1, 2004, pp. 998–1003.

[Low99] David G. Lowe, « Object recognition from local scale-invariant features », *in*: *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, IEEE, 1999, pp. 1150–1157.

[LSV05] Vincenzo Lippiello, Bruno Siciliano, and Luigi Villani, « Eye-in-hand/eye-to-hand multi-camera visual servoing », *in*: *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, vol. 2005, 2005, pp. 5354–5359.

[LVS12] Frank L Lewis, Draguna Vrabie, and Vassilis L Syrmos, *Optimal control*, John Wiley & Sons, 2012.

[Mal04] E. Malis, « Improving vision-based control using efficient second-order minimization techniques », *in*: *IEEE International Conference on Robotics and Automation*, 2004, 1843–1848 Vol.2.

[Man12] N. Mansard, « A dedicated solver for fast operational-space inverse dynamics », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 4943–4949.

[MBG96] P. Martinet, F. Berry, and J. Gallice, « Use of first derivative of geometric features in visual servoing », *in*: *Proceedings of the 1996 13th IEEE International Conference on Robotics and Automation. Part 1 (of 4)*, vol. 4, IEEE, 1996, pp. 3413–3419.

[MC02a] Ezio Malis and François Chaumette, « Theoretical improvements in the stability analysis of a new class of model-free visual servoing methods », *in*: *IEEE Transactions on Robotics and Automation* 18.*2* (2002), pp. 176–186.

[MC02b] Éric Marchand and François Chaumette, « Virtual visual servoing: A framework for real-time augmented reality », *in*: *Computer Graphics Forum*, vol. 21, 3, Wiley/Blackwell (10.1111), 2002, pp. 289–297.

[MC07] Nicolas Mansard and François Chaumette, « Task sequencing for high-level sensor-based control », *in*: *IEEE Transactions on Robotics* 23.*1* (2007), pp. 60–72.

[MC08] Mohammed Marey and Francois Chaumette, « Analysis of classical and new visual servoing control laws », *in*: *2008 IEEE International Conference on Robotics and Automation* (2008), pp. 3244–3249.

[MC10] Mohammed Marey and François Chaumette, « A new large projection operator for the redundancy framework », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, pp. 3727–3732.

[MCB99] Ezio Malis, François Chaumette, and Sylvie Boudet, « 2-1/2-D visual servoing », *in*: *IEEE Transactions on Robotics and Automation* 15.*2* (1999), pp. 238–250.

[MD16] Michael Muehlebach and Raffaello D'Andrea, « Parametrized infinite-horizon model predictive control for linear time-invariant systems with input and state constraints », *in*: *Proceedings of the American Control Conference*, vol. 2016-July, Institute of Electrical and Electronics Engineers Inc., 2016, pp. 2669–2674.

[MKF08] Toshiyuki Murao, Hiroyuki Kawai, and Masayuki Fujita, « Predictive Visual Feedback Control with Eye-in/to-Hand Configuration via Stabilizing Receding Horizon Approach », *in*: *IFAC Proceedings Volumes* 41.*2* (2008), pp. 5341–5346.

[MKK09] Nicolas Mansard, Oussama Khatib, and Abderrahmane Kheddar, « A unified approach to integrate unilateral constraints in the stack of tasks », *in*: *IEEE Transactions on Robotics* 25.*3* (2009), pp. 670–685.

[MKX14] Abolfazl Mohebbi, Mohammad Keshmiri, and Wen Fang Xie, « An acceleration command approach to robotic stereo image-based visual servoing », *in*: *IFAC Proceedings Volumes (IFAC-PapersOnline)*, vol. 19, 3, Elsevier, 2014, pp. 7239–7245.

[MR93] Henri Michel and Patrick Rives, « Singularities in the determination of the situation of a robot effector from the perspective view of 3 points », *in*: (1993).

[MRC09] Nicolas Mansard, Anthony Remazeilles, and François Chaumette, « Continuity of varying-feature-set control laws », *in*: *IEEE Transactions on Automatic Control* 54.*11* (2009), pp. 2493–2505.

[MSC05] Eric Marchand, Fabien Spindler, and Francois Chaumette, « ViSP for visual servoing: a generic software platform with a wide class of robot control skills », *in*: *IEEE Robotics & Automation Magazine* 12.*December* (2005), pp. 40–52.

[Ozg+13] Eroi Ozgur, Redwan Dahmouche, Nicolas Andreff, and Philippe Martinet, « High speed parallel kinematic manipulator state estimation from legs observation », *in*: *IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 424–429.

[PM+18] David Perez-Morales, Olivier Kermorgant, Salvador Dominguez-Quijada, and Philippe Martinet, « Automatic Perpendicular and Diagonal Unparking Using a Multi-Sensor-Based Control Approach », *in*: *2018 15th International Conference on Control, Automation, Robotics and Vision, ICARCV 2018*, Institute of Electrical and Electronics Engineers Inc., 2018, pp. 783–788.

[Pow78] M. J. D. Powell, « A fast algorithm for nonlinearly constrained optimization calculations », *in*: Springer, Berlin, Heidelberg, 1978, pp. 144–157.

[PPT14] Jorge Pomares, Iván Perea, and Fernando Torres, « Dynamic visual servoing with chaos control for redundant robots », *in*: *IEEE/ASME Transactions on Mechatronics* 19.*2* (2014), pp. 423–431.

[RMC06] Anthony Remazeilles, Nicolas Mansard, and François Chaumette, « A qualitative visual servoing to ensure the visibility constraint », *in*: *IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2006, pp. 4297–4303.

[RY05] Thomas Philip Runarsson and Xin Yao, « Search biases in constrained evolutionary optimization », *in*: *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews* 35.*2* (2005), pp. 233–243.

[Sau+06] M. Sauvee, P. Poignet, E. Dombre, and E. Courtial, « Image Based Visual Servoing through Nonlinear Model Predictive Control », *in*: *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego (CA), 2006, pp. 1776–1781.

[SE91] C. Samson and B. Espiau, « Application of the task-function approach to sensor-based control of robot manipulators », *in*: *IFAC Symposia Series - Proceedings of a Triennial World Congress*, vol. 5, 8, Publ by Pergamon Press Inc, 1991, pp. 269–274.

[SEB91] Claude Samson, Bernard Espiau, and Michel Le Borgne, *Robot control: the task function approach*, Oxford University Press, Inc., 1991.

[SG13] Riccardo Spica and Paolo Robuffo Giordano, « A framework for active estimation: Application to structure from motion », *in*: *Proceedings of the IEEE Conference on Decision and Control*, Institute of Electrical and Electronics Engineers Inc., 2013, pp. 7647–7653.

[SGC15] Riccardo Spica, Paolo Robuffo Giordano, and Francois Chaumette, « Plane estimation by active vision from point features and image moments », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2015-June, June, IEEE, 2015, pp. 6003–6010.

[Sha+13] Navid Shahriari, Silvia Fantasia, Fabrizio Flacco, and Giuseppe Oriolo, « Robotic visual servoing of moving targets », *in*: *IEEE International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 77–82.

[Sic+10] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo, *Robotics: modelling, planning and control*, Springer Science & Business Media, 2010.

[SS91] B. Siciliano and J J E Slotine, « A general framework for managing multiple tasks in highly redundant robotic systems », *in*: *Advanced Robotics 1991 Robots in Unstructured Environments 91 ICAR Fifth International Conference on* (1991), 1211–1216 vol.2.

[TC05] Omar Tahri and François Chaumette, « Point-based and region-based image moments for visual servoing of planar objects », *in*: *IEEE Transactions on Robotics* 21.*6* (2005), pp. 1116–1127.

[Ted+16] David Tedaldi, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza, « Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS) », *in*: *2016 2nd International Conference on Event-Based Control, Communication, and Signal Processing, EBCCSP 2016 - Proceedings*, IEEE, 2016, pp. 1–7.

[Tho+14] Justin Thomas, Giuseppe Loianno, Koushil Sreenath, and Vijay Kumar, « Toward image based visual servoing for aerial grasping and perching », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, IEEE, 2014, pp. 2113–2118.

[Thu+02] Benoit Thuilot, Philippe Martinet, Lionel Cordesses, and Jean Gallice, « Position based visual servoing: Keeping the object in the field of vision », *in*: *Proceedings - IEEE International Conference on Robotics and Automation* 2 (2002), pp. 1624–1629.

[TM08] Omar Tahri and Youcef Mezouar, « On the efficient second order minimization and image-based visual servoing », *in*: *Proceedings - IEEE International Conference on Robotics and Automation*, 2008, pp. 3213–3218.

[TM10] Omar Tahri and Youcef Mezouar, « On visual servoing based on efficient second order minimization », *in*: *Robotics and Autonomous Systems* 58.*5* (2010), pp. 712–719.

[Van+16] Sylvain Vandernotte, Abdelhamid Chriette, Philippe Martinet, and Adolfo Suarez Roos, « Dynamic sensor-based control », *in*: *14th International Conference on Control, Automation, Robotics and Vision*, Phuket (TH), 2016, pp. 1–6.

[Van16] Sylvain Vandernotte, « Manipulation référencée multi-capteurs d'objets manufacturés », PhD Thesis, Ecole Centrale de Nantes, 2016, p. 165.

[WHB96] William J. Wilson, Carol C.Williams Hulls, and Graham S. Bell, « Relative end-effector control using cartesian position based visual servoing », *in*: *IEEE Transactions on Robotics and Automation* 12.*5* (1996), pp. 684–696.

[De +16] Jose Oniram A. De Limaverde Filho, Tiago S. Lourenco, Eugenio Fortaleza, Andre Murilo, and Renato V. Lopes, « Trajectory tracking for a quadrotor system: A flatness-based nonlinear predictive control approach », *in*: *2016*

IEEE Conference on Control Applications, CCA 2016, Institute of Electrical and Electronics Engineers Inc., 2016, pp. 1380–1385.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

CENTRALE NANTES

Titre : Asservissement Visuel Dynamique pour Bras Manipulateurs Rapides

Mot clés : Asservissement Visuel, Commande par Modèle Prédictif, Manipulateurs Redondants

Résumé : Cette thèse traite de l'augmentation de la productivité des robots manufacturiers, lors de l'exécution de tâches référencées capteurs. De telles tâches peuvent provenir de la cible non positionnée de manière absolue ou d'un environnement mal connu.

Les commandes par asservissement visuel sont bien connues pour leur robustesse et leur précision, mais nécessitent généralement de longs temps d'exécution en raison de différents facteurs. La commande est généralement formulée uniquement à un niveau cinématique et caractérisée par des vitesses décroissantes de façon exponentielle. De plus, l'application non linéaire de l'espace opérationnel à l'espace des capteurs peut conduire à des chemins sous-optimaux et plus longs.

Pour augmenter les performances de commande et réduire le temps nécessaire à la réalisation d'une tâche, cette thèse étudie l'utilisation de modèles d'interaction de second ordre. Leur utilisation dans une commande au niveau dynamique est étudiée et comparée aux approches classiques. Ils sont ensuite utilisés dans des schémas de commande par modèle prédictif, permettant d'obtenir des vitesses plus élevées tout en générant de meilleures trajectoires. Cependant, un inconvénient des techniques prédictives est leur charge de calcul. Afin d'obtenir de pallier ce défaut, un nouveau type de commande prédictive est étudié. Il conduit à une réduction du nombre de variables impliquées dans les problèmes d'optimisation grâce à l'utilisation d'un paramétrage des séquences d'entrée.

Title: Dynamic Visual Servoing for Fast Robotic Arms

Keywords: Visual Servoing, Model Predictive Control, Redundant Manipulators

Abstract: This thesis deals with increasing the productivity in manufacturing robots, when performing sensor-based tasks. Such tasks may be coming from the target not being absolutely positioned.

Visual servoing control schemes are well known for their robustness and precision, but generally require long execution times due to different factors. Control laws are generally formulated only at a kinematic level and characterized by exponentially decreasing velocities. Moreover, the nonlinear map from the operational space to the sensor space can lead to sub-optimal and longer paths.

To increase control performances and reduce the time required to complete a task, this thesis investigates the use of second-order interaction models. Their use in dynamic feedback control laws is investigated and compared to classical controllers. They are then employed in Model Predictive Control (MPC) schemes, allowing to obtain higher velocities and better sensor trajectories. However, a drawback of MPC techniques is their computational load. In order to obtain even better results, a new type of predictive control is thus investigated, leading to a reduced number of variables involved in MPC optimization problems thanks to the use of a parameterization of the control input sequences.