Motion planning for robotic manipulators using robust constrained control

Andrea Maria Zanchettin*, Paolo Rocco

Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Piazza Leonardo Da Vinci 32, Milano, Italy

Since their first appearance in the 1970's, industrial robotic manipulators have considerably extended their application fields, allowing end-users to adopt this technology in previously unexplored scenarios. Correspondingly, the way robot motion can be specified has become more and more complex, requiring new capabilities to the robot, such as reactivity and adaptability. For an even enhanced and widespread use of industrial manipulators, including the newly introduced collaborative robots, it is necessary to simplify robot programming, thus allowing this activity to be handled by non-expert users. Next generation robot controllers should intelligently and autonomously interpret production constraints, specified by an application expert, and transform them into motion commands only at a lower and real-time level, where updated sensor information or other kind of events can be handled consistently with the higher level specifications. The availability of several execution strategies could be then effectively exploited in order to further enhance the flexibility of the resulting robot motion, especially during collaboration with humans.

This paper presents a novel methodology for motion specification and robust reactive execution. Traditional trajectory generation techniques and optimisation-based control strategies are merged into a unified framework for simultaneous motion planning and control. An experimental case study demonstrates the effectiveness and the robustness of this approach, as applied to an image-guided grasping task.

Keywords: Robotic manipulators, Trajectory planning, Constraints, Robustness, Uncertainty, Industrial robots

1. Introduction

Robot motions are typically programmed by means of Cartesian and angular position and velocity profiles of the end-effector along a given path. Programming a specific task, which happens in today's motion generation algorithms, results in solving the motion planning problem by actually over-constraining the space of solutions in order to select a particular end-effector motion among others. Even more advanced and commercially available trajectory planning strategies prevent the low-level controller from adapting or modifying the generated trajectory based on real-time events or sensor readings, or need a lot of handling logics to be pre-programmed, rarely guaranteeing hard real-time capabilities or reduced reaction times.

Constraint-based programming of robot motions represents the most natural solution to the problem of indirectly planning a robot trajectory based on process requirements, which are specified by means of constraints. The constraint-based approach, originally introduced in Siciliano and Slotine (1991) based on the task-function formalism, Samson, Espiau, and Borgne (1991), has been recently and intensively exploited within the so-called iTaSC (instantaneous Task Specification using Constraints) framework, see De Schutter et al. (). While originally developed to cope only with instantaneous constraints (i.e., those corresponding to the current time instant), recent extensions towards a more comprehensive constraints representation are reported, see e.g. Decre, Bruyninckx, and De Schutter (2013). A very similar approach has been presented within the Stack of Tasks framework, see Mansard, Khatib, and Kheddar (2009), Escande, Mansard, and Wieber (2014). Similar to the Stack of Tasks, the iCAT (inequality Control objectives, Activations and Transitions) task priority framework was introduced in Simetti and Casalino (2016) to deal with smooth transitions during task activation/deactivation. Yet another similar prioritised framework has been recently proposed in Tazaki and Suzuki (2014). In Kermorgant and Chaumette (2014), another control method dealing with constraints is presented: constraints (such as joint limits and field-of-view constraints) are conveniently defined as additional costs to be optimised and activated by suitable thresholds.

A constraint-based approach to accommodate joint position limits, as well as velocity, acceleration or even torque ones has been proposed in Antonelli, Chiaverini, and Fusco (2003), Flacco and De Luca (2015) with reactive capabilities based on time-based trajectory scaling. These methods, however, only rely on pure kinematic or dynamic rescaling of a predefined trajectory, which limits the reaction capabilities of the robot. In fact, a simple scaling technique not always guarantees the existence of a feasible motion. Moreover, such approaches only deal

Received 10 May 2016;
Received in revised form 8 September 2016;
Accepted 16 November 2016
Available online 06 December 2016

* Corresponding author.
E-mail addresses: andreamaria.zanchettin@polimi.it (A.M. Zanchettin), paolo.rocco@polimi.it (P. Rocco).

Published Journal Article available at http://dx.doi.org/10.1016/j.conengprac.2016.11.010

with decoupled constraints in the joint space like the avoidance of joint position, velocity or acceleration limits.

In the framework of constraint-based trajectory planning, it is important to mention the works in Macfarlane and Croft (2003), Biagiotti and Melchiorri (2008), Kroeger and Wahl (2010) which present various approaches for online generation of smooth and time-optimal trajectories for multi-axes machines and robots.

For a better and reliable task execution, the problem of trajectory planning and the subsequent (constraint-based) control should be merged in a unified framework with capabilities of both motion planning and reactive control/execution. Beside the constraint-based specification, the first idea of connecting planning and reactive/adaptation capabilities was developed in Quinlan and Khatib (1993), within the well-known Elastic Strips framework, and later refined within the Elastic Bands framework, Brock and Khatib (2002). Other examples of real-time reaction planners can be found in, e.g., Haddadin et al. (2010), where virtual and physical contact are merged to form the desired reactive behaviour, or in Khansari-Zadeh and Billard (2012), where the reactive behaviour of the robot is obtained by the imposition of a proper attractor dynamics. Finally (Hauser, 2012) presents a re-planning strategy to be executed when the nominal plan fails, due to unpredictable obstacles movements.

This work presents a method to combine a trajectory generation algorithm with a constrained optimisation problem, which relies on a continuously updated reference trajectory and a reactive control strategy. Further details on the features of the proposed algorithm and on how the method stands with respect to available results are discussed in the next Section. The remainder of this paper is organised as follows. Section 2 describes the motivations underlying this work and discusses the original contribution with respect to the existing literature. In Section 3, the main contribution is detailed and an algorithm for constraint-based reactive trajectory generation is presented. Section 4 briefly discusses two extensions to cope with redundant robots and to enforce a compliant behaviour of the robot, respectively. Section 5 complements the specification of the algorithm with best practice to guarantee robustness of the overall system with respect to measurement uncertainties. Finally, an experimental case study, describing how a selected application may benefit from the proposed approach, is reported in Section 6.

2. Motivations and problem setting

In this Section, we first motivate our work and then discuss the main features of the approach as compared with the state of the art.

2.1. Motivations and key concepts

Today's robots are able to execute tasks that are way more complex than those they were originally designed for. Is worth noticing that the VAL scripting language, developed by Unimation Robotics in the mid 1970's, consisted in no more than 30 instructions. Nowadays, advanced scripting languages for modern industrial robots, like ABB's RAPID or KUKA KRL, include hundreds of different instructions. Therefore, as the adoption of robots is becoming more and more pervasive, novel applications require more and more advanced functionalities, hence more involved programming primitives, especially to promptly react or adapt to unpredicted situations.

The main idea behind this work is to develop a control architecture that endows the robot with advanced flexibility during task execution. Specifically, we propose to use control tools to move from an *imperative programming* paradigm (i.e. specifying the robot how to perform a task), in which process requirements are semantically and uniquely mapped by the robot programmer into a suitable end-effector velocity profile, towards a *declarative programming* paradigm (i.e. specifying requirements for task execution, and leave to the robot the autonomy to execute it properly). With this paradigm, process requirement and constraints are turned by the controller into motion commands only at run-time,

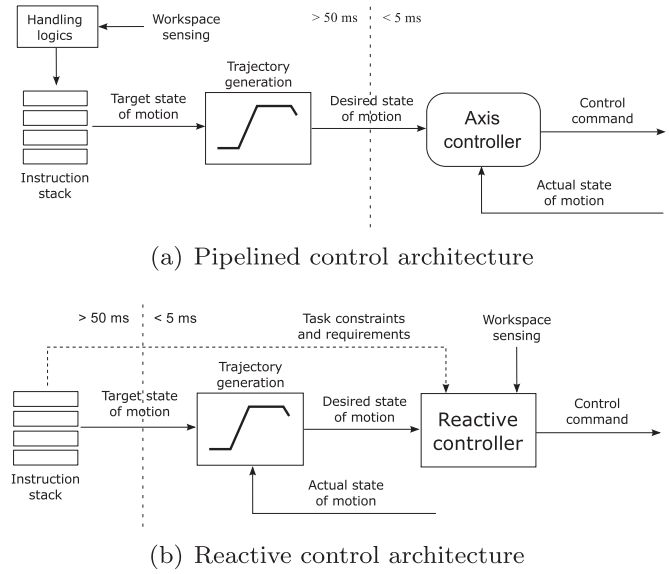


Fig. 1. Proposed control architecture (bottom) as compared to a traditional solution (top).

hence embedding the capability of handling real-time events, with reduced pre-programmed control logics. This shift of paradigm represents a major improvement for the control of new generation robots working in unstructured environments, and thus allows a more widespread use of robotic technology, especially in SMEs.

A pictorial view of the idea is sketched in Fig. 1, as compared to the traditional pipelined approach. In traditional robot programming, see Fig. 1(a), the trajectory is offline computed and possibly optimised, whilst being only online evaluated, hence leading to a completely pipelined and non reactive execution. This architecture may introduce inefficient handling strategy of sensed events, as the event has to traverse the whole architecture before ultimately influencing the control commands. Moreover, the programmer should specify how to handle each single event or deviation from the nominal task execution when a corresponding trajectory has been already generated. This setting has clearly two major drawbacks: (a) the user of the robotic application should be simultaneously an expert of both the process and the typical robotic programming structures, and (b) a significant lag between an event and the corresponding handle triggering may be thus introduced.

Within this work, we propose a method, see Fig. 1(b), to continuously re-generate, and hence evaluate, the reference trajectory towards a given goal depending on the current state of the robot. The reactive controller generates the next control command based on the current desired state of motion and of the workspace sensing, with the aim of tracking the desired motion to the extent allowed by the constraints specified by the programmer. If this entails a deviation from the planned trajectory, a new trajectory from the current state of motion to the target is generated. Furthermore the robot is endowed with reactive capabilities within a millisecond time scale, something difficult, if not impossible, to achieve with a traditional pipelined architecture. This way the robotic programmer is free to focus only on process requirements (which are then suitably turned into constraints) without caring of *what-if* handling strategies to define each possible non-nominal behaviour to cope with at execution time. The main task of the programmer is then to specify all the common features of each possible nominal behaviour, whilst a particular one will be eventually selected during execution, based on the actual context (i.e., presence of human co-workers, moving obstacles, etc).

2.2. Contributions and comparisons

Most of the available results in the literature, like the iTaSC (De

Schutter et al.,), the Stack of Tasks (Escande et al., 2014) or others (Flacco & De Luca, 2015), are focused on the control aspects of the constraint resolution, i.e. they propose methods to solve constrained optimisation problems in real-time. Also, the usually address on a regulation problem, rather than a trajectory tracking one.

In Ceriani, Zanchettin, Rocco, Stolt, and Robertsson (2015), as a compromise between offline planning and real-time execution, we have proposed a method to adapt the execution of a certain motion based on perception. The method was able to (i) temporarily suspend the generation of the nominal reference trajectory, (ii) react to some event, (iii) plan a recovery trajectory towards the point where it was interrupted, and eventually (iv) resume the nominal behaviour to reach the target state of motion. The control architecture selected in that work was motivated by the reduced accessibility of the proprietary trajectory planner of the robot controller. It turned out, however, that in many circumstances, the recovery strategy (iii) and the corresponding handover to the nominal execution (iv) were not strictly necessary, a re-planning towards the target state being more effective. The point is then how to endow the robot controller with re-planning capabilities in the most effective, reliable, and easy to implement way.

Differently from the available re-planning strategies that are triggered whenever the nominal plan fails, see also Hauser (2012), the present paper addresses the problem of combining trajectory generation and constraint-based control from a different and unifying perspective. In particular, the trajectory generation algorithm is moved *within the feedback loop*, so as to be continuously adapted to all the sensed information. The approach itself of continuously adapting the reference trajectory is very similar to the Elastic Strips/Bands framework (Brock & Khatib, 2002). Similar adaptation approaches can be found in Haddadin et al. (2010), Khansari-Zadeh and Billard (2012), where proper attractor dynamics are defined and updated to steer the robot towards the desired position.

In this work, however, the robot trajectory is modified online by means of constraints, rather than based on virtual forces, potential fields or attractors, thus guaranteeing satisfaction of hard constraints (like e.g. safety, or joint limits).

Moving from the above considerations, the present paper contributes in: (i) presenting a framework to evolve from a purely pipelined planning and execution control architecture, towards a more flexible one which guarantees lower reaction time and reduced handling logics; (ii) proposing an algorithm that continuously adapt the trajectory of the robot, by solving a constrained optimisation problem; (iii) addressing the problem of robust satisfaction of constraints to deal with measurement uncertainties. The idea behind this work has been originally presented in Zanchettin and Rocco (2013) and verified in simulations, only, while a specific application involving redundant degrees of freedom has been preliminarily presented in Zanchettin and Rocco (2015). This paper further contributes with a more comprehensive definition of the programming and control framework, including the dynamic properties of the robot and analysing the effect of the uncertainties on the performance of the control strategy, also based on an experimental verification.

3. Proposed method

In this Section we formulate the problem to be solved. Further, we present the reactive control architecture and detail the constraints the robot must enforce during its motion. In particular, we split the discussion into two parts concerning the joint space and the task space, respectively.

3.1. Symbols and notation

The symbols and corresponding notation adopted in the paper are listed here:

$\boldsymbol{q}$	vector of joint variables
$\dot{\boldsymbol{q}}$	time derivative of $\boldsymbol{q}$
$\boldsymbol{x}$	vector of task variables
${}_i q$	i-th element of vector $\boldsymbol{q}$
${}_i \boldsymbol{A}$	i-th row of matrix $\boldsymbol{A}$
$\boldsymbol{A}_k$	matrix $\boldsymbol{A}$ evaluated at discrete time instant k
$\boldsymbol{b}_k$	vector $\boldsymbol{b}$ evaluated at discrete time instant k

3.2. Robot behaviour in the joint space

Consider the dynamic model of a generic n-dof robot in the joint space as a nonlinear discrete-time dynamical system

$$\boldsymbol{\tau}_k = \boldsymbol{B}(\boldsymbol{q}_k)\boldsymbol{u}_k + \boldsymbol{h}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) + \boldsymbol{\tau}_f(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + T_s\dot{\boldsymbol{q}}_k + 0.5T_s^2\boldsymbol{u}_k\dot{\boldsymbol{q}}_{k+1} = \dot{\boldsymbol{q}}_k + T_s\boldsymbol{u}_k \tag{1}$$

where T_s is the sampling time of the control law, $\boldsymbol{B}$ is the positive definite inertia matrix, $\boldsymbol{h}$ represents centrifugal/Coriolis, and gravitational torques, $\boldsymbol{\tau}$ stands for the motor torque input signal, whilst $\boldsymbol{\tau}_f$ is an additional friction torque. Here, we consider a second order kinematic control strategy, where the joint acceleration $\ddot{\boldsymbol{q}}_k = \boldsymbol{u}_k$ represents the control input. The behaviour of system (1) is subject to (hard) constraints, such as joint position, velocity and torque limits, like

$${}_i q^{min} \leq {}_i q \leq {}_i q^{max} \quad |{}_i \dot{q}| \leq {}_i \dot{q}^{max} \quad |{}_i \tau| \leq {}_i \tau^{max} \tag{2}$$

Let ${}_i\mathcal{Q}_\infty$ represent the largest region in plane ${}_i q - {}_i \dot{q}$ such that any point $({}_i q, {}_i \dot{q})$ admits at least one feasible input acceleration ${}_i u$ allowing the system to remain within the region itself. For further details on invariant sets for linear systems we refer the reader to Blanchini (1999). This eventually corresponds to a pair of configuration and velocity dependent values for the maximum and minimum input accelerations, i.e.

$${}_i u_{inf} \leq {}_i u \leq {}_i u^{sup} \tag{3}$$

3.3. Trajectory generation and task constraints

Let vector $\boldsymbol{x}$ identify a set of m task variables (e.g., Cartesian or cylindrical coordinates, Euler angles, etc.), Samson et al. (1991). The relationship between the robot joint space and the corresponding task space is well-known and is just recalled here

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}) \quad \dot{\boldsymbol{x}} = \boldsymbol{J}(\boldsymbol{q})\dot{\boldsymbol{q}} \quad \ddot{\boldsymbol{x}} = \dot{\boldsymbol{J}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{J}(\boldsymbol{q})\boldsymbol{u} \tag{4}$$

where $\boldsymbol{J}$ is the m-by-n task Jacobian matrix, which locally (i.e., *instantaneously*) relates the joint space to the task space. Notice that, since the system (1) together with the output equation $\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q})$ has a relative degree of two, second order differentiation is needed to make the dependency of the task variable $\boldsymbol{x}$ on the control input $\boldsymbol{u}$ explicit.

For task variables $\boldsymbol{x}$ and their time derivatives, dynamics similar to (1) are assumed. By taking into account the kinematic relations between the task space and the joint space in (6), the following relations hold

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s\boldsymbol{J}_k\dot{\boldsymbol{q}}_k + 0.5T_s^2(\dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k + \boldsymbol{J}_k\boldsymbol{u}_k)\dot{\boldsymbol{x}}_{k+1} = \boldsymbol{J}_k\dot{\boldsymbol{q}}_k + T_s(\dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k + \boldsymbol{J}_k\boldsymbol{u}_k) \tag{5}$$

where $\boldsymbol{J}_k = \boldsymbol{J}(\boldsymbol{q}_k)$ and, similarly, $\dot{\boldsymbol{J}}_k = \dot{\boldsymbol{J}}(\boldsymbol{q}_k)$. Moreover, we assume the following bounds to apply on task velocities and accelerations

$$|{}_i \dot{x}| \leq {}_i \dot{x}^{max} \quad |{}_i \ddot{x}| \leq {}_i \ddot{x}^{max} \tag{6}$$

Vectors $\dot{\boldsymbol{x}}^{max}$ and $\ddot{\boldsymbol{x}}^{max}$ denote maximum task velocities and accelerations, respectively.

A variety of methods to generate a trajectory from a given state of motion $(\boldsymbol{x}_k, \dot{\boldsymbol{x}}_k)$ to a target one $(\boldsymbol{x}^{trg}, \dot{\boldsymbol{x}}^{trg})$ exist, see e.g. Biagiotti and

Melchiorri (2008), Kroeger and Wahl (2010). Their output can be made consistent with bound constraints like those in (6). We here assume that an algorithm for trajectory generation is available and returns the planned trajectory, in terms of desired task position and velocity, at time instant $k + 1$.

3.4. Development of the method

We here aim at developing a kinematic control strategy that handles different kinds of constraints, both in the task and in the joint space, possibly known only at execution time, as it happens in case of sensor-related constraints. Without any loss of generality, we assume that the robot motion has to be consistent with the following constraints[1]

$$\boldsymbol{E}_k \boldsymbol{u}_k \leq \boldsymbol{f}_k \tag{7}$$

Such constraints are introduced to cope with sensor-related events occurring at time instant k to be handled, at time $k + 1$, with a proper selection of the input acceleration $\boldsymbol{u}_k$. The case study in Section 6 provides an example of constraints expressed in such form.

A block diagram describing the overall algorithm has been already reported in Fig. 1(b). It comprises two main modules: *Trajectory generation* and *Reactive controller*. The block named *Trajectory generation* is responsible for generating a reference trajectory for each of the task variables, by implementing any of the well-known algorithms described in Biagiotti and Melchiorri (2008). It has notion of all task constraints, i.e., the target state of motion $(\boldsymbol{x}^{trg}, \dot{\boldsymbol{x}}^{trg})$, and of the velocity and acceleration bounds $\dot{\boldsymbol{x}}^{max}$, $\ddot{\boldsymbol{x}}^{max}$. On the other hand, this module completely ignores all constraints at joint level as well as all sensor-dependent constraints. The output of this module is the next state of motion in the task space $(\boldsymbol{x}_{k+1}^{ref}, \dot{\boldsymbol{x}}_{k+1}^{ref})$ to be possibly reached by the robot within the next discrete-time interval.

This information is propagated to the *Reactive controller*, whose role is to track the reference signal while accommodating the constraints in (7). Its output consists in the next state of motion $(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1})$ to be commanded to the lower level (axes) controllers. Because of the constraints $\boldsymbol{E}_k \boldsymbol{u}_k \leq \boldsymbol{f}_k$ to be handled, the reference calculated by the *Trajectory generation* module might not be feasible at the current time instant. Therefore the block *Direct kinematics* evaluates (4) to compute the state of motion in the task space $(\boldsymbol{x}_k, \dot{\boldsymbol{x}}_k)$ which is then used by the Trajectory generation module within the next control cycle, as a starting point for the computation of an updated trajectory.

Algorithm 1.

Input: $\boldsymbol{q}_k$, $\dot{\boldsymbol{q}}_k$, $\boldsymbol{x}^{trg}$, $\dot{\boldsymbol{x}}^{trg}$, $\dot{\boldsymbol{x}}^{max}$, $\ddot{\boldsymbol{x}}^{max}$, $\boldsymbol{E}_k$, $\boldsymbol{f}_k$
Output: $\boldsymbol{q}_{k+1}$, $\dot{\boldsymbol{q}}_{k+1}$, $\boldsymbol{\tau}_k$

1: compute relevant kinematic quantities (Jacobians, task variables, etc.)
2: generate a trajectory, compliant with(2), to connect $\boldsymbol{x}_k$, $\dot{\boldsymbol{x}}_k$ with $\boldsymbol{x}^{trg}$, $\dot{\boldsymbol{x}}^{trg}$
3: evaluate the next desired state of motion $\boldsymbol{x}_{k+1}^{ref}$, $\dot{\boldsymbol{x}}_{k+1}^{ref}$
4: evaluate acceleration bounds $\boldsymbol{u}_{inf}$, $\boldsymbol{u}^{sup}$ as in(3)
5: solve the following QP problem

$$\min_{\boldsymbol{u}_k} \mathcal{L}(\dot{\boldsymbol{x}}_{k+1} - \dot{\boldsymbol{x}}_{k+1}^{ref}, \boldsymbol{x}_{k+1} - \boldsymbol{x}_{k+1}^{ref}) \tag{8a}$$

$$\text{subject to} \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + T_s \boldsymbol{J}_k \dot{\boldsymbol{q}}_k + 0.5 T_s^2 (\dot{\boldsymbol{J}}_k \dot{\boldsymbol{q}}_k + \boldsymbol{J}_k \boldsymbol{u}_k) \tag{8b}$$

$$\dot{\boldsymbol{x}}_{k+1} = \boldsymbol{J}_k \dot{\boldsymbol{q}}_k + T_s (\dot{\boldsymbol{J}}_k \dot{\boldsymbol{q}}_k + \boldsymbol{J}_k \boldsymbol{u}_k) \tag{8c}$$

$$\boldsymbol{\tau}_k = \boldsymbol{B}(\boldsymbol{q}_k)\boldsymbol{u}_k + \boldsymbol{h}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) + \boldsymbol{\tau}_f(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) \tag{8d}$$

$$\boldsymbol{u}_{inf} \leq \boldsymbol{u}_k \leq \boldsymbol{u}^{sup} \tag{8e}$$

$$-\boldsymbol{\tau}^{max} \leq \boldsymbol{\tau}_k \leq \boldsymbol{\tau}^{max} \tag{8f}$$

$$-\dot{\boldsymbol{x}}^{max} \leq \dot{\boldsymbol{x}}_{k+1} \leq \dot{\boldsymbol{x}}^{max} \tag{8g}$$

$$-\ddot{\boldsymbol{x}}^{max} \leq \dot{\boldsymbol{J}}_k \dot{\boldsymbol{q}}_k + \boldsymbol{J}_k \boldsymbol{u}_k \leq \ddot{\boldsymbol{x}}^{max} \tag{8h}$$

$$\boldsymbol{E}_k \boldsymbol{u}_k \leq \boldsymbol{f}_k \tag{8i}$$

6: update the state of motion as in(1)

$$\boldsymbol{\tau}_k = \boldsymbol{B}(\boldsymbol{q}_k)\boldsymbol{u}_k + \boldsymbol{h}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) + \boldsymbol{\tau}_f(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + T_s \dot{\boldsymbol{q}}_k + 0.5 T_s^2 \boldsymbol{u}_k \dot{\boldsymbol{q}}_{k+1} = \dot{\boldsymbol{q}}_k + T_s \boldsymbol{u}_k \tag{9}$$

In the following, we further detail the algorithm implementing the *Reactive controller* in order to (i) transform task velocity into corresponding joint commands (inverse kinematics), (ii) handle sensor-related constraints and (iii) track the reference signal.

A commonly adopted methodology to transform task velocities into corresponding joint commands relies on the so called Closed-Loop Inverse Kinematics (CLIK) algorithm. Different solutions exist to cope with either first-order, Siciliano (1990), or second-order differential kinematics, Caccavale, Chiaverini, and Siciliano (1997). While the CLIK algorithm is well-suited to accurately track a given task space trajectory, it is however hardly applicable to reactive control strategies, especially in case the robot motion is subject to constraints, apart from those in (4). Moreover, in some situations, the tracking accuracy of the reference has lower priority with respect to other constraints in terms of task completion. For this reason, we introduce Algorithm 1, to simultaneously handle the different types of constraints as well as the inverse kinematics problem. The quadratic cost function in (8a) has been introduced to weigh the difference between the next state of motion $(\boldsymbol{x}_{k+1}, \dot{\boldsymbol{x}}_{k+1})$ and its reference values $(\boldsymbol{x}_{k+1}^{ref}, \dot{\boldsymbol{x}}_{k+1}^{ref})$ obtained as an output of the trajectory generation algorithm. A generic, yet quadratic, cost function can be used to this purpose, i.e.

$$\mathcal{L}(\boldsymbol{e}, \dot{\boldsymbol{e}}) = 0.5 \boldsymbol{e}^T \boldsymbol{Q}_p \boldsymbol{e} + 0.5 \dot{\boldsymbol{e}}^T \boldsymbol{Q}_v \dot{\boldsymbol{e}} + \boldsymbol{e}^T \boldsymbol{Q}_{p,v} \dot{\boldsymbol{e}} + + \boldsymbol{g}_p^T \boldsymbol{e} + \boldsymbol{g}_v^T \dot{\boldsymbol{e}} \tag{10}$$

where $\boldsymbol{Q}_p$, $\boldsymbol{Q}_v$ are positive definite matrices, while $\boldsymbol{Q}_{p,v}$, $\boldsymbol{g}_p$ and $\boldsymbol{g}_v$ are a matrix and two vectors of suitable dimensions.

Constraints (8b) and (8c) are required to map joint space velocities and accelerations into their task space counterparts. Constraints in (8f) represent the maximum motor actuator torques, while (8d) specifies the dynamics of the robotic manipulator. Constraints (8e) are introduced to maintain the robot's next state of motion $(\boldsymbol{q}_{k+1}, \dot{\boldsymbol{q}}_{k+1})$ within the maximum invariant set $\mathcal{Q}_\infty$. Constraints in (8g) and (8h) relate to velocity and acceleration bounds on the task variables, respectively. Finally, (8i) represents all instantaneous and sensor-related constraints, as already discussed.

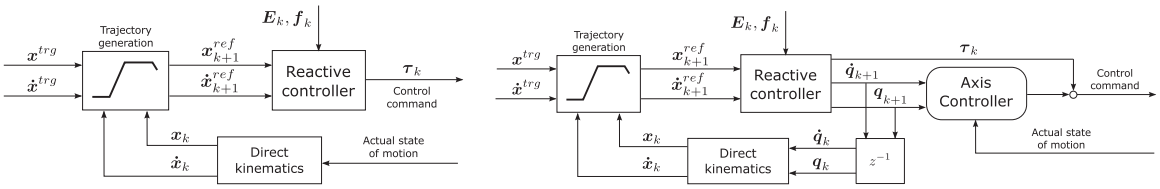
The output of the QP problem consists in the desired joint acceleration $\boldsymbol{u}_k$ from which the corresponding desired position $\boldsymbol{q}_{k+1}$ and velocity $\dot{\boldsymbol{q}}_{k+1}$ can be computed using (1).

Remark. A constrained QP formulation has been introduced to solve the control problem, which guarantees stability of the resulting motion thanks to the introduction of the invariant set in ${}_i\mathcal{Q}_\infty$, see Scokaert, Mayne, and Rawlings (1999). Using a state of the art QP solver (Ferreau, Kirches, Potschka, Bock, & Diehl, 2014), a solution can be obtained within a millisecond time scale, thus allowing a real-time implementation. The existence of a solution clearly depends on the selection of additional, and typically sensor-dependent, constraints in (7). In Section 5, constraints are formulated so as to guarantee existence of a solution, even in case of uncertainties.

3.5. Possible implementations

In describing the control architecture for simultaneous trajectory

[1] See Zanchettin and Rocco (2015) for an example of such constraints.



(a) Implementation as an actual feedback robot controller.

(b) Implementation as a trajectory generation algorithm on top of an existing axis controller.

Fig. 2. Possible implementations of the developed methodology.

generation and control, no assumption has been made on the existence of a pre-existing (industrial) controller. In particular, the output of the algorithm, as sketched in Fig. 1(b), consists in position, velocity and torque signals. The torque signal $\boldsymbol{\tau}_k$ can be used directly as a motor torque input to be commanded, while $\boldsymbol{q}_k$, $\dot{\boldsymbol{q}}_k$ represent the actual state of the robot. This way, the described algorithm can be used directly as a feedback controller. This situation is sketched in Fig. 2(a). It usually happens in practical situations, however, that the high-level motion planning algorithm must be interfaced with a lower-level axis controller. In this case, the output signals of the algorithm in Fig. 1(b) can be used as references (i.e. feedforward actions) for a pre-existing lower level axis controller. This, more common, situation is sketched in Fig. 2(b).

4. Kinematic redundancy and compliant behaviour

4.1. Redundant degrees of freedom

In case of redundant degrees of freedom or in case of task redundancy, the cost function in (10) is no longer strictly positive definite, although $\boldsymbol{Q}_p$ and $\boldsymbol{Q}_v$ are designed to be positive definite. It follows that the QP problem in (8) might have infinite, and equally optimal, solutions. Let $\boldsymbol{u}_k^0$ be any of those optimal solutions. All solutions must then satisfy the following constraint

$$\boldsymbol{J}_k(\boldsymbol{u}_k - \boldsymbol{u}_k^0) = \boldsymbol{0} \tag{11}$$

meaning that all alternative solutions might differ from the candidate one $\boldsymbol{u}_k^0$ only in the null space of the task Jacobian, see e.g. Kanoun, Lamiraux, and Wieber (2011), Escande et al. (2014). In this case, it is convenient, and to some extent mandatory, to exploit this redundancy to achieve additional and secondary requirements, having lower priority with respect to the main task. Within an optimisation-based framework this is usually handled by introducing a second optimisation stage, acting only in the task null space. We then introduce the following optimisation problem, which inherits all the constraints from the original one in (8), together with the corresponding optimality criterion in (11) (thus preserving the optimality of the higher priority task)

$$\min_{\boldsymbol{u}_k} \frac{1}{2}\boldsymbol{u}_k^T \boldsymbol{Q}_u \boldsymbol{u}_k + \boldsymbol{g}_u^T \boldsymbol{u}_k \tag{12a}$$

$$\boldsymbol{J}_k \boldsymbol{u}_k = \boldsymbol{J}_k \boldsymbol{u}_k^0 \tag{12b}$$

$$\boldsymbol{u}_{inf} \leq \boldsymbol{u}_k \leq \boldsymbol{u}^{sup} \tag{12c}$$

$$\boldsymbol{E}_k \boldsymbol{u}_k \leq \boldsymbol{f}_k \tag{12d}$$

where $\boldsymbol{Q}_u$ and $\boldsymbol{g}_u$ are design parameters of suitable dimensions, whilst $\boldsymbol{Q}_u$ is now a positive definite matrix in order to ensure the uniqueness of the solution in terms of commanded joint accelerations $\boldsymbol{u}_k$.

4.2. Compliant motion

In some applications, the motion of the robot should be made compliant to external forces: this is the case, for example, of safe human-robot interaction. This means that the trajectory followed by the robot is modified by an external force or moment. Assume a force $\boldsymbol{\mu}$ is applied to the robot TCP (Tool Centre Point), then the following admittance filter can be adopted to compute the variation of the desired trajectory with respect to the nominal, i.e. planned, one

$$\Delta\dot{\boldsymbol{x}}_{k+1} = \Delta\dot{\boldsymbol{x}}_k + T_s\boldsymbol{M}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k)\Delta\boldsymbol{x}_{k+1} = T_s\Delta\dot{\boldsymbol{x}}_k + 0.5T_s^2\boldsymbol{M}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{D}\Delta\dot{\boldsymbol{x}}_k) \tag{13}$$

where $\Delta\boldsymbol{x}$ and $\Delta\dot{\boldsymbol{x}}$ represent the position and velocity modifications, respectively, while $\boldsymbol{M}$ and $\boldsymbol{D}$ are positive definite design matrices composing a mass-damper low-pass filter. Eq. (13) represents the state update and the output equation of a set of, possibly independent, first-order systems, having $\Delta\dot{\boldsymbol{x}}$ as state variable. The output equation simply computes the positional displacement $\Delta\boldsymbol{x}$. In order to accommodate external forces, the computed trajectory reference is modified as follows

$$\tilde{\boldsymbol{x}}_{k+1}^{ref} = \boldsymbol{x}_{k+1}^{ref} + \Delta\boldsymbol{x}_{k+1} \tilde{\dot{\boldsymbol{x}}}_{k+1}^{ref} = \dot{\boldsymbol{x}}_{k+1}^{ref} + \Delta\dot{\boldsymbol{x}}_{k+1} \tag{14}$$

Finally, the updated reference has to be forwarded to the actual reactive controller, described in the previous Section. Notice that, in case of saturation or activation of other constraints, the robot TCP might not be able to follow the modified trajectory. For this reason, the output of the direct kinematics is sent back to the admittance filter, to prevent wind-up phenomena to arise. The state of the admittance filter is then updated as $\Delta\dot{\boldsymbol{x}}_{k+1} = \tilde{\dot{\boldsymbol{x}}}_{k+1}^{ref} - \dot{\boldsymbol{x}}_{k+1}$.

5. Robust constraints specification

Despite the very general form of the constraints in (7), within previous implementations of our and similar approaches, only small attention has been paid to possible uncertainties in the description of the system kinematics/dynamics or in sensor measurement. The presence of uncertainties or disturbances (of different nature) might easily compromise the satisfaction of a given constraint. Only few attempts to solve this problem are actually reported in the literature, see e.g. Del Prete and Mansard (2015). The main approach therein is to rely on the so called chance constraints, Schwarm and Nikolaou (1999), hence entailing an underlying probabilistic model of all possible uncertainties. In the following, inspired by Blanchini, Mayne, Seron, and Rakovi (2005), we first describe a systematic way to handle uncertainties within the formulation of constraints of relative degree two, i.e. position-dependent. Then, we discuss how the same strategy can be adopted for other constraints with lower relative degree.

5.1. Robust bounds on configuration dependent variables

Consider a generic scalar task function $p = p(\boldsymbol{q})$ and its time derivatives

$$\dot{p} = \boldsymbol{J}_p(\boldsymbol{q})\dot{\boldsymbol{q}} \quad \ddot{p} = \dot{\boldsymbol{J}}_p(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{J}_p(\boldsymbol{q})\boldsymbol{u} \tag{15}$$

and assume, that during the motion of the robot, the controller must ensure an upper bound[2] for variable p, i.e. $p_k \leq p^{max}$, $\forall\, k$.

[2] Similar arguments can be derived for lower bounds.

Define the corresponding state of motion as $\boldsymbol{\pi} = [p \ \ \dot{p}]^T$. Then, the behaviour of variable p in time, can be described by the following discrete time system

$$\boldsymbol{\pi}_{k+1} = \boldsymbol{A}\boldsymbol{\pi}_k + \boldsymbol{B}\ddot{p}_k \tag{16}$$

where

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix} \boldsymbol{B} = \begin{bmatrix} 0.5T_s^2 \\ T_s \end{bmatrix}$$

and T_s is the sampling time. In order to achieve such a requirement, the authors of Scheint et al. (2008) analytically derived the biggest controllable invariant set, corresponding to the given constraint. In particular, any control law satisfying the invariance must ensure, at any time, the following constraint

$$\phi(p, \dot{p}) \leq 0 \tag{17}$$

where

$$\phi(p, \dot{p}) = \begin{cases} p + \frac{\dot{p}^2}{2\ddot{p}^{max}} - p^{max}, & \text{if } \dot{p} > 0 \\ p - p^{max}, & \text{otherwise} \end{cases} \tag{18}$$

where $\ddot{p}^{max}$ represents the maximum acceleration of p, i.e. $\ddot{p}_k \leq \ddot{p}^{max}$, $\forall\, k$. Let us denote with $\mathbb{I}$, the region of the $\boldsymbol{\pi}$-space in which condition (17) holds, i.e.

$$\mathbb{I} = \{\boldsymbol{\pi}\colon \phi(p, \dot{p}) \leq 0\} \tag{19}$$

Unfortunately, Eq. (16) only accounts for the nominal behaviour of variable p in time, neglecting possible uncertainties and measurement errors. A better description of the evolution of p in time is given by:

$$\boldsymbol{\pi}_{k+1} = \boldsymbol{A}(\boldsymbol{\pi}_k + \boldsymbol{\Delta}_k) + \boldsymbol{B}\ddot{p}_k + \boldsymbol{\nu}_k \tag{20}$$

where $\boldsymbol{\Delta}$ and $\boldsymbol{\nu}$ represent the measurement uncertainty on the state vector and a generic, non structured, modelling error, respectively. We here assume $\boldsymbol{\Delta}_k$ and $\boldsymbol{\nu}_k$ to be bounded and such that $\forall\, k$, $\boldsymbol{\Delta}_k \in \mathbb{D}$ and $\forall\, k$, $\boldsymbol{\nu}_k \in \mathbb{V}$, where $\mathbb{D}$ and $\mathbb{V}$ are generic polytopes. By accounting for uncertainty, Eq. (20) introduces a more realistic representation of a system that the one in (16).

The values of p and $\dot{p}$ at discrete time instant $k+1$ can be then written in terms of the following set inclusion

$$\boldsymbol{\pi}_{k+1} \in \boldsymbol{A}\boldsymbol{\pi}_k + \boldsymbol{B}\ddot{\mathbb{P}} \oplus \boldsymbol{A}\mathbb{D} \oplus \mathbb{V} \tag{21}$$

where $\oplus$ represents the Minkowski sum, while $\ddot{\mathbb{P}} = \{\ddot{p}\colon \ddot{p} \leq \ddot{p}^{max}\}$. In order to make the invariant set $\mathbb{I}$ robust with respect to the given bounded uncertainties, the controller must select a value $\ddot{p}_k \in \ddot{\mathbb{P}}$ such that $\boldsymbol{\pi}_{k+1} \in \mathbb{I}$ even in case of worst-case uncertainties. Eq. (21) represents all possible values of $\boldsymbol{\pi}_{k+1}$ given a certain initial state $\boldsymbol{\pi}_k$ and for all possible uncertainties. The key idea is to find the maximum and the minimum admissible control inputs $\ddot{p}^{sup}$ and $\ddot{p}_{inf}$ such that $\boldsymbol{\pi}_{k+1} \in \mathbb{I}$. As the right hand side of Eq. (21) suggests, the reachable set of $\boldsymbol{\pi}_{k+1}$ results from the sum of different sets. The relationship between these different sets, accounting for the two kinds of possible uncertainties, is depicted in Fig. 3. In order to allow the controller to select an admissible value for $\ddot{p}_k$, the following optimisation problem can be introduced

$$\ddot{p}_k^{sup} = \max_{\ddot{p}_k \in \ddot{\mathbb{P}}} \ddot{p}_k \ \text{ such that } \boldsymbol{A}\boldsymbol{\pi}_k + \boldsymbol{B}\ddot{p}_k \oplus \boldsymbol{A}\mathbb{D} \oplus \mathbb{V} \subseteq \mathbb{I}. \tag{22}$$

Its solution gives the maximum allowed acceleration $\ddot{p}_k^{sup}$ such that the (robust) invariance property is guaranteed.

Remark. The solution of problem (22) provides the maximum value for $\ddot{p}_k$ such that the constraint $p \leq p^{max}$ is made robustly invariant, meaning that it will remain satisfied at time instant $k+1$ and on, for any possible value of the uncertainties within the prescribed bounds, i.e. for all possible $\boldsymbol{\Delta}_k \in \mathbb{D}$ and $\boldsymbol{\nu}_k \in \mathbb{V}$.So far, the input $\ddot{p}_k$ has been considered ideal, i.e. not affected by uncertainties. It is worth noticing that should the signal $\ddot{p}_k$ be uncertain itself, problem (22) can be written in the following way

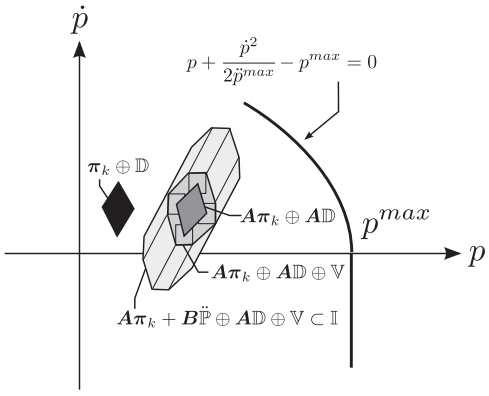


Fig. 3. Geometric interpretation of condition (21).

$$\ddot{p}_k^{sup} = \max_{\ddot{p}_k + \Delta\ddot{p}^{max} \in \ddot{\mathbb{P}}} \ddot{p}_k \ \text{ such that } \boldsymbol{A}\boldsymbol{\pi}_k + \boldsymbol{B}(\ddot{p}_k + \Delta\ddot{p}^{max}) \oplus \boldsymbol{A}\mathbb{D} \oplus \mathbb{V} \subseteq \mathbb{I} \tag{23}$$

where $\Delta\ddot{p}^{max}$ represents the maximum actuation uncertainty. Notice that, problem (23) is equivalent to (22) once the condition $\ddot{p}_k \leq \ddot{p}_k^{max}$ is replaced with $\ddot{p}_k \leq \ddot{p}^{max} - \Delta\ddot{p}^{max}$ in the definition of $\ddot{\mathbb{P}}$. In the current setting, i.e. polytopic sets and piece-wise linear and quadratic boundaries, the mentioned problem can be solved in closed form, i.e. without adopting a numerical algorithm, by solving suitably defined second order polynomial equations.

Finally, the constraint that the given control variable $\boldsymbol{u}$ must satisfy at time k can be written as follows

$$\boldsymbol{J}_p(\boldsymbol{q}_k)\boldsymbol{u}_k \leq \ddot{p}_k^{sup} - \dot{\boldsymbol{J}}_p(\boldsymbol{q}_k)\dot{\boldsymbol{q}}_k \tag{24}$$

hence consistently with the general form in (7).

5.2. Robust bounds on velocity dependent variables and actuation torques

Consider now a constraint of the form $\dot{p} = \dot{p}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \leq \dot{p}^{max}$ and assume that $\partial\dot{p}/\partial\dot{\boldsymbol{q}}$ is always full rank. The discrete time behaviour of variable $\dot{p}$ can be now introduced by the following (nominal) first order discrete time system

$$\dot{p}_{k+1} = \dot{p}_k + T_s\ddot{p}_k \tag{25}$$

whilst the corresponding uncertain system is as follows

$$\dot{p}_{k+1} = \dot{p}_k + \Delta_k + T_s\ddot{p}_k + v_k \tag{26}$$

where we assume $\Delta_k \in \mathbb{D} = \{\Delta\colon \Delta_{min} \leq \Delta_k \leq \Delta^{max}\}$, $\forall\, k$ and similarly for the input noise $v_k \in \mathbb{V} = \{v\colon v_{min} \leq v_k \leq v^{max}\}$, $\forall\, k$. Differently from the previous case, in case of constraints of unitary relative degree (i.e. velocity-dependent), the biggest controllable invariant set is already represented by the constraint itself, i.e. $\mathbb{I} = \{\dot{p}\colon \dot{p} \leq \dot{p}^{max}\}$.

$$\ddot{p}_k^{sup} = \max_{\ddot{p}_k \in \ddot{\mathbb{P}}} \ddot{p}_k \ \text{ such that } \dot{p}_k + T_s\ddot{p}_k \oplus \mathbb{D} \oplus \mathbb{V} \subseteq \mathbb{I} \tag{27}$$

which can be simply translated into the general form (7) in the same way as explained in the previous case.

Finally, since within the presented strategy we adopt a full dynamic model of the manipulator, it is important to notice that at least the real joint friction torque is uncertain with respect to the modelled one $\boldsymbol{\tau}_f(\boldsymbol{q}, \dot{\boldsymbol{q}})$ in (1). It follows that the joint torque bounds in (2) should be rewritten in order to account for the uncertain representation of the friction torque. To this end, it is convenient to explicitly account for the uncertainty on the model of friction torque by introducing the following range of variability

$$\boldsymbol{\tau}_{f,min}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \leq \boldsymbol{\tau}_f(\boldsymbol{q}, \dot{\boldsymbol{q}}) \leq \boldsymbol{\tau}_f^{max}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \tag{28}$$

Hence, the specification of torque actuation bounds can be made robust by simply accounting for the variability of the friction torques as follows

$$\boldsymbol{\tau}_{inf} \leq \boldsymbol{B}(\boldsymbol{q}_k)\boldsymbol{u}_k + \boldsymbol{h}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k) \leq \boldsymbol{\tau}^{sup} \tag{29}$$

where $\boldsymbol{\tau}_{inf} = -\boldsymbol{\tau}^{max} + \boldsymbol{\tau}_{f,min}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)$ and $\boldsymbol{\tau}^{sup} = \boldsymbol{\tau}^{max} - \boldsymbol{\tau}_f^{max}(\boldsymbol{q}_k, \dot{\boldsymbol{q}}_k)$.

6. Experimental case study

This Section describes an experimental case study, that shows most of the features of the developed control architecture: robust definition of constraints, compliant motion and redundant degree(s) of freedom. As an experimental verification, we consider an image-based grasping task, performed by a redundant robot. The experimental setup consists of an ABB dual-arm lightweight prototype robot of which only the 7-DOF right arm will be used, with an eye-in-hand camera (web-cam), see Fig. 4. The robot has to approach the orange sphere, while being compliant with respect to external forces applied by a human operator. As the robot is endowed with the capability of accommodating external forces, it is also asked to constantly maintain the visibility of the object within the camera field of view. Finally, a vertical virtual wall is implemented to prevent both the TCP and the elbow to enter a forbidden area on the right of the robot itself.

6.1. Task constraints and redundancy resolution

Typical frames describing the grasping operation are shown in Fig. 4: $\mathcal{W} \in SE(3)$ is the world frame, $\mathcal{R} \in SE(3)$ is the robot TCP frame, $\mathcal{C} \in SE(3)$ is the camera frame related to $\mathcal{R}$ by means of the so-called extrinsic parameters and $\mathcal{O} \in SE(3)$ is the object frame. The object frame is rigidly attached to the object itself. In order to accomplish its task, the robot has to align frame $\mathcal{R}$ to the object frame $\mathcal{O}$. Vector of task variables $\boldsymbol{x} = [x\ y\ z\ \rho\ \theta\ \phi]^T$ is introduced to represent the origin of frame $\mathcal{R}$ in frame $\mathcal{W}$ and the corresponding XYZ Euler angles.

First, the robot has to estimate the position of the object frame $\mathcal{O}$ with respect to the world frame $\mathcal{W}$; this is accomplished by taking a picture of the scene. The position of the object is then computed with off-the-shelf computer vision algorithms.

Then, the robot starts its grasping motion. Additional constraints are here introduced to maintain as much as possible the object centred in the image, or at least in the camera field of view. This is obtained by limiting the camera velocity in some directions of motion whenever some relevant features approach the border of the image. This requirement will be further detailed in the following, with special emphasis to its robust characterisation.

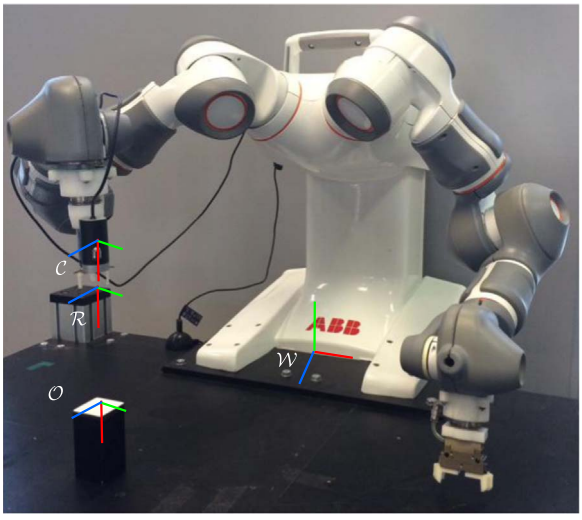


Fig. 4. Setup for the experimental verification comprising: a redundant robot with eye-in-hand camera, and the object to be grasped.

For a point feature, the adoption of the perspective camera model results in the following relationship between point coordinates and corresponding position $\boldsymbol{\xi} = (e, f)$ (in pixels) in the image frame

$$e = -\lambda\frac{x_C}{z_C} + 0.5e^{max} \quad f = -\lambda\frac{y_C}{z_C} + 0.5f^{max}$$

where x_C, y_C and z_C are the point coordinates in the camera frame C, λ is the focal length (in pixels), while $\boldsymbol{\xi}^{max} = (e^{max}, f^{max})$ defines image width and height.

Consider the following relationship between the time derivatives of image features and the velocity of the camera frame:

$$\dot{\boldsymbol{\xi}} = \begin{bmatrix} \dot{e} \\ \dot{f} \end{bmatrix} = \boldsymbol{L}(\boldsymbol{q}) \begin{bmatrix} \dot{\boldsymbol{p}}_O \\ \boldsymbol{\omega}_O \end{bmatrix} = \boldsymbol{L}(\boldsymbol{q})\boldsymbol{J}_C(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{P}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{30}$$

where $\boldsymbol{L}$ is the so-called *interaction matrix*, see Espiau, Chaumette, and Rives (1992), $\dot{\boldsymbol{p}}_O$ and $\boldsymbol{\omega}_O$ are the linear and angular velocities of the camera frame (in local frame), while $\boldsymbol{J}_C$ is the Jacobian of the camera frame and $\boldsymbol{P} = \boldsymbol{L}\boldsymbol{J}_c$. The relationship containing the control input $\boldsymbol{u}$ is obtained by further differentiating (30), i.e.

$$\ddot{\boldsymbol{\xi}} = \dot{\boldsymbol{P}}(\boldsymbol{q})\dot{\boldsymbol{q}} + \boldsymbol{P}(\boldsymbol{q})\boldsymbol{u} \tag{31}$$

The evolution of variables $\boldsymbol{\xi}$ and $\dot{\boldsymbol{\xi}}$ can be written in terms of the following (nominal) discrete time system, see (16)

$$\boldsymbol{\xi}_{k+1} = \boldsymbol{\xi}_k + T_s\dot{\boldsymbol{\xi}}_k + 0.5T_s^2\ddot{\boldsymbol{\xi}}_k\dot{\boldsymbol{\xi}}_{k+1} = \dot{\boldsymbol{\xi}}_k + T_s\ddot{\boldsymbol{\xi}}_k \tag{32}$$

As both the feature coordinates $\boldsymbol{\xi}$ as well as their time derivatives $\dot{\boldsymbol{\xi}}$ might be affected by uncertainties (e.g. pixel discretisation and/or approximation of the interaction matrix $\boldsymbol{L}$ in (30), the approach described in Section 5 might be beneficial to the purpose of maintaining such feature within the camera field of view. Once uncertainty bounds are specified, using the approach described in Section 5, visibility constraints can be written as follows

$$\ddot{\boldsymbol{\xi}}_k^{inf} \leq \boldsymbol{P}_k\boldsymbol{u}_k + \dot{\boldsymbol{P}}_k\dot{\boldsymbol{q}}_k \leq \ddot{\boldsymbol{\xi}}_k^{sup} \tag{33}$$

As for the virtual wall, for the TCP position the requirement can be simply formulated as

$$y \geq y_{min}$$

and made robust with respect to possible uncertainties

$$\ddot{y}_k^{inf} \leq {}_2\boldsymbol{J}_k\boldsymbol{u}_k + {}_2\dot{\boldsymbol{J}}_k\dot{\boldsymbol{q}}_k \leq \ddot{y}_k^{sup}$$

where $\ddot{y}_k^{inf}$ and $\ddot{y}_k^{max}$ are computed as described in Section 5, while ${}_2\boldsymbol{J}$ stands for the 2nd row of the Jacobian matrix, corresponding to the y-coordinate. A similar discussion can be made to implement the same constraint for the elbow.

All the listed constraints can be ultimately translated into the general form (7).

As the given manipulator is redundant, it is necessary to implement an additional optimisation layer to resolve this kinematic redundancy. One possible way to exploit the redundant degree of freedom of the manipulator is to make its motion similar to the human arm performing the same task, using the method explained in Zanchettin, Bascetta, & Rocco (2013b). This redundancy resolution technique has also proved to facilitate the perceived safety by humans collaborating with robots, Zanchettin, Bascetta, & Rocco (2013a).

Since the particular selection of a redundancy resolution scheme is outside the scope of this paper, in the experiments the redundancy is exploited in order to select the minimum velocity in the joint space.

6.2. Implementation and experiments

The robot is position controlled and the axis controller, receiving joint position and velocity references at 250 Hz ($T_s = 4$ ms), communicates with an external LINUX real-time PC through a research inter-

face. The same PC computes the overall algorithm and implements the computer vision processing code developed with the OpenCV[3] libraries running at around 30 fps. A simple extrapolation strategy has been implemented to increase the update rate of the video processing algorithm. The qpOASES[4] solver has been used to implement the control algorithm. As for the trajectory generation module in Fig. 1(b), the Reflexxes Motion Library[5] has been used. The robot is not equipped with joint torque sensors. Therefore, a model-based sensor-less reconstruction of interaction forces is introduced. The inertial, centrifugal and gravitational torques have been made available from the robot manufacturer, whilst the friction torques and their variability, see (28), have been experimentally identified offline, using standard Least Squares (LS) techniques. The sensor-less torque estimation is based on the method in De Luca and Mattone (2005). The following values have been assigned within the cost function in (8a)

$$\boldsymbol{Q} = {}_{p}\boldsymbol{I} \quad \boldsymbol{Q} = {}_{v}0.1\boldsymbol{Q} \quad {}_{p}$$

while all the other weights in (10) are set to zero. As for the redundancy resolution criterion, the cost function in (12a) has been tuned as follows

$$\boldsymbol{Q}_u = \mathrm{diag}(({}_{i}\dot{q}^{max})^{-2}) \quad \boldsymbol{g}_u = T_s^{-1}\dot{\boldsymbol{q}}_k$$

. Finally, the values for the mass-damper admittance filter have been selected as $\boldsymbol{M} = \boldsymbol{I}, \boldsymbol{D} = 20\boldsymbol{I}$, suitable to obtain high reactivity of the robot with respect to external forces.

Two experiments have been run. During the experiments, the QP problem (8) in Algorithm 1, consisting in 7 optimisation variables and 43 constraints, has been solved within an average time of 360 μs (less than 700 μs in worst case), while the second QP problem set to handle the redundant degree of freedom has been solved in closed-form, hence in negligible time. Statistics are shown in Fig. 5 with reference to a 2,5 GHz INTEL Core i5 (only one core is used by the QP solver).

6.2.1. Experiment without robustness

The first experiment has been run without accounting for the robust characterisation of the constraints, i.e. setting all uncertainties to zero.

Fig. 6 reports a bird's eye view of the workspace showing the traversed path of both the end-effector and of the elbow. In point ① a force is applied to the robot end-effector. The robot accommodates the external force, and approaches the virtual wall with both the end-effector and the elbow. As one can see, the virtual wall constraints are satisfied, since they are not affected by any uncertainty.

Fig. 7 reports the position of the object within the camera frame. During the experiment, due to external forces, the object approaches the limit of the camera frame. At a certain point, the object cannot be maintained within the field of view and the experiment terminates with a failure state.

6.2.2. Experiment with robustness

The second experiment has been performed similarly to the first one, except that the robust characterisation of the constraints explained in this paper in Section 5 has been enforced. In particular, for each feature coordinate and its time derivative, the maximum measurement uncertainty, i.e. $|\boldsymbol{\Delta}_k| \leq \boldsymbol{\Delta}^{max}$, has been set to 8 pixels (mainly to account for pixel quantisation and uncertainty in features extraction) and 4 pixels/second (to account for uncertainty on the focal length λ as well as on depth estimation within the interaction matrix in (30)), respectively. As for possible modelling errors $\boldsymbol{v}_k$, the same worst-case values, i.e. such that $|\boldsymbol{v}_k| \leq \boldsymbol{v}^{max}$, have been considered to possibly account for camera distortion effects, and similar, yet generic, non idealities. Concerning the virtual wall constraint, 1 mm and 1 mm/s

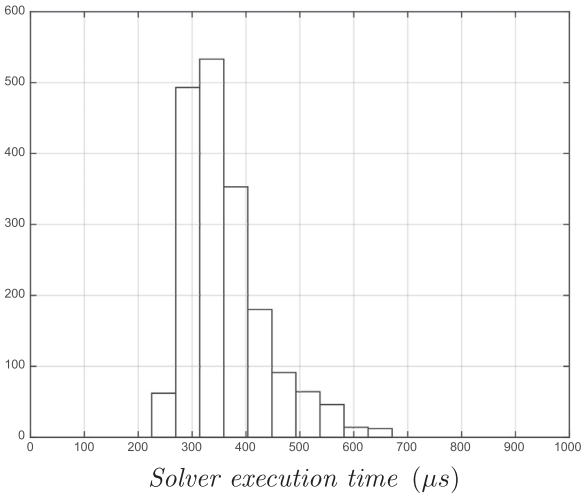


Fig. 5. Execution time of the first QP problem.

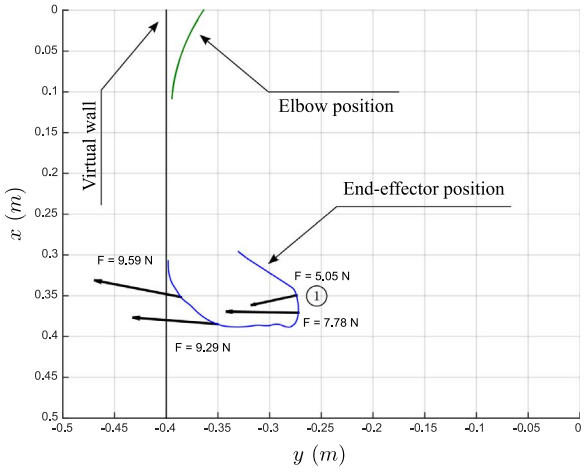


Fig. 6. End-effector position (blue) and elbow position (green) – bird's eye view (without robust constraint characterisation) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.).

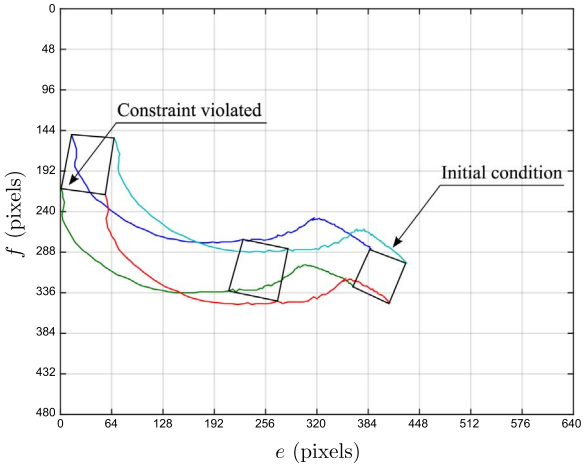


Fig. 7. Object position in the camera frame (without robust constraint characterisation).

have been selected to describe possible uncertainties, which are mainly due to the discrete time implementation of the control law.

Fig. 8 reports the position of the end-effector and of the elbow in the Cartesian space. Similarly to the previous experiment, while the robot is accommodating external forces ①, both the end-effector and the elbow approach the virtual wall. Also in this case, the correspond-

[3] http://www.opencv.org/.
[4] https://projects.coin-or.org/qpOASES/.
[5] http://www.reflexxes.ws.

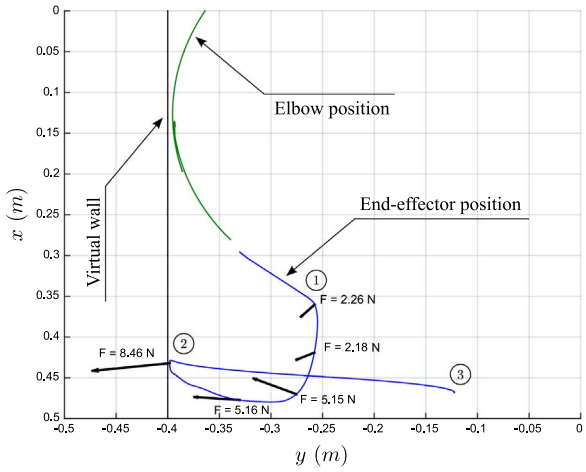


Fig. 8. End-effector position (blue) and elbow position (green) – bird's eye view. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

ing constraints are satisfied as they do not entail any uncertainty. Once the external force vanishes, ②, the robot is able to reach its final destination ③.

Fig. 9 reports the object position in the camera frame. During this experiment, as the algorithm is able to account for possible measurement uncertainties, the object can be maintained within the camera field of view all along the execution of the task.

Finally, Fig. 10 reports the 3D trajectory of the end-effector during the experiment and also reports the corresponding planned trajectory, i.e. the output of the *Trajectory generation* of Fig. 1(b), which is evaluated at each control cycle and represents the end-effector trajectory that would be executed in case no further events occur.

6.3. Discussion

With reference to a possible novel programming language, using the proposed method, this kind of task could be simply specified by means of a single instruction like

```
MoveTo object with
    \compliant-behaviour
    \visibility
    \y-wall>=-0.4;
```

without the need to further define handling strategies, e.g. when the object approaches the borders of the image or the virtual wall.

As discussed within the introduction, the possibility to specify the

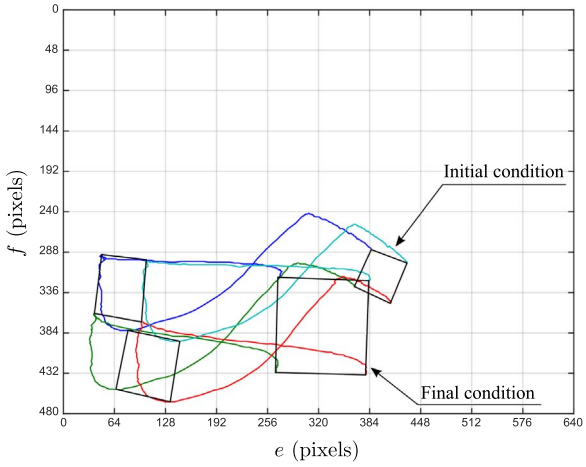


Fig. 9. Object position in the camera frame.

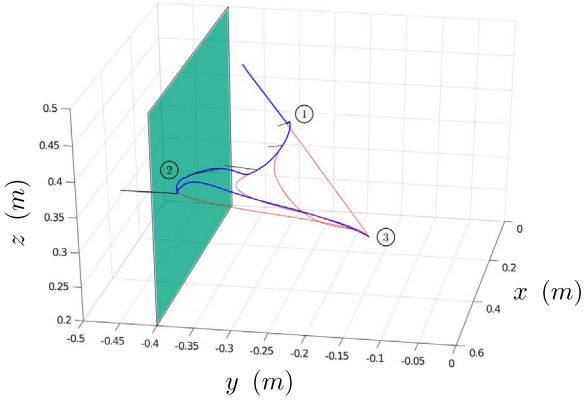


Fig. 10. Resulting trajectory of the end-effector (black) and planned trajectories during the experiment (red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

task of the robot without focusing on a specific execution strategy will allow enhanced reactivity, which is not possible with today's motion controller algorithm. Moreover, the novel paradigm described in this paper will free the robotic programmer from the need of specifying complex handling logics, which unnecessarily complicate the readability and the maintenance of the program. As an example, within the reported case study, it is not necessary to (i) specify a complete robot trajectory at programming time, (ii) implement *what-if* handling logics to modify the planned trajectory to obtain a compliant behaviour and, above all, (iii) there is no need at all to specify a priori how the robot should behave when one of the object features approaches the boundaries of the camera field of view or when the robot itself approaches the virtual wall. Using today's motion specification paradigm, this can be done by specifying all possible reactions to all possible situations to be handled.

Another important aspect of the present work is that there is no need for the robot to recover the *nominal* execution, as the method itself does not entail any nominal execution, since any possible execution satisfying the given constraints is eligible to be used by the robot. With reference to the second experiment, for example, when the effect of external forces vanishes ②, the robot can directly proceed to its target position ③ without adopting any recovery strategy, i.e. without returning to point ① where the external forces were accommodated by the robot.

Finally, as compared with similar strategies within the literature, robustness with respect to measurement as well as to generic uncertainties can be efficiently handled in real-time, thus mitigating the effect of noise, which typically affects sensed information.

From the above considerations, we believe that the work reported in this paper might stimulate discussion on whether the current implementation of robot programming and control algorithms is suitable to handle future challenges in robotics. Robotic technology will be adopted in increasingly demanding applications, requiring advanced reactive capabilities, especially when robots need to interact with human workers. The method presented in this paper is beneficial for next generation robots as it tends to simultaneously simplify robot programming and to endow the robot controller with robustness and adaptation capabilities. Nowadays, reactive and adaptive behaviours are only obtained with complicated *what-if* logics to be specified at programming time, which seriously limit the possibility for a more flexible adoption of robotic manipulators.

7. Conclusions

In this work we presented a control method to endow a robot with reaction capabilities with respect to sensed events. The method has been shown to be effective in a practical application and allows to

achieve possibly complex behaviours by composing basic functionalities that are specified in terms of task constraints. The effectiveness of the presented methodology has been experimentally verified focusing on an image-guided grasping task, during which the robot was able to exhibit a compliant behaviour with respect to external forces, while robustly maintaining the visibility of the object to be grasped.

Appendix A. Supplementary data

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.conengprac.2016.11.010.

References

Antonelli, G., Chiaverini, S., & Fusco, G. (2003). A new on-line algorithm for inverse kinematics of robot manipulators ensuring path tracking capability under joint limits. *IEEE Transactions on Robotics and Automation*, *19*(*1*), 162–167.

Biagiotti, L., & Melchiorri, C. (2008). *Trajectory planning for automatic machines and robots*. Springer.

Blanchini, F. Constrained control for uncertain linear systems, *Journal of Optimization Theory and Applications 71* (3) 465–484

Blanchini, F. (1999). Set invariance in control. *Automatica*, *35*(*11*), 1747–1767. Brock, O., & Khatib, O. (2002). Elastic strips: a framework for motion generation in human environments. *The International Journal of Robotics Research*, *21*(*12*), 1031–1052.

Caccavale, F., Chiaverini, S., & Siciliano, B. (1997). Second-order kinematic control of robot manipulators with jacobian damped least-squares inverse: theory and experiments. *IEEE/ASME Transactions on Mechatronics*, *2*(*3*), 188–194.

Ceriani, N. M., Zanchettin, A. M., Rocco, P., Stolt, A., & Robertsson, A. (2015). Reactive task adaptation based on hierarchical constraints classification for safe industrial robots. *IEEE/ASME Transactions on Mechatronics*, *20*(*6*), 2935–2949.

De Luca, A., & Mattone, R., (2005). Sensorless robot collision detection and hybrid force/motion control. *In IEEE International Conference on Robotics and Automation*.

De Schutter, J., De Laet, T., Rutgeerts, J., Decre, W., Smits, R., Aertbelien, E., Claes, K., & Bruyninckx, H. Constraint-based task specification and estimation for sensor-based robot systems in the presence of geometric uncertainty, *The International Journal of Robotics Research 26* (5).

Decre, W., Bruyninckx, H., & De Schutter, J. (2013). Extending the iTaSC constraint-based robot task specification framework to time-independent trajectories and user-configurable task horizons. *In IEEE International Conference on Robotics and Automation*, ICRA.

Del Prete, A., & Mansard, N. (2015). Addressing constraint robustness to torque errors in task-space inverse dynamics. *In Robotics, Sciences and Systems*.

Escande, A., Mansard, N., & Wieber, P.-B. (2014). Hierarchical quadratic programming: fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, *33*(*7*), 1006–1028.

Espiau, B., Chaumette, F., & Rives, P. (1992). A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, *8*(*3*), 313–326.

Ferreau, H., Kirches, C., Potschka, A., Bock, H., & Diehl, M. (2014). qpOASES: a parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, *6*(*4*), 327–363.

Flacco, F., & De Luca, A. (2015). Discrete-time redundancy resolution at the velocity level with acceleration/torque optimization properties. *Robotics & Autonomous Systems*, *70*, 191–201.

Haddadin, S., Urbanek, H., Parusel, S., Burschka, D., Roßmann, J., Albu-Schaffer, A., & Hirzinger, G., (2010). Real-time reactive motion generation based on variable attractor dynamics and shaped velocities. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS.

Hauser, K. (2012). On responsiveness, safety, and completeness in real-time motion planning. *Autonomous Robots*, *32*(*1*), 35–48.

Kanoun, O., Lamiraux, F., & Wieber, P. B. (2011). Kinematic control of redundant manipulators: generalizing the task-priority framework to inequality task. *IEEE Transactions on Robotics*, *27*(*4*), 785–792.

Kermorgant, O., & Chaumette, F. (2014). Dealing with constraints in sensor-based robot control. *IEEE Transactions on Robotics*, *30*(*1*), 244–257.

Khansari-Zadeh, S. M., & Billard, A. (2012). A dynamical system approach to realtime obstacle avoidance. *Autonomous Robots*, *32*(*4*), 433–454.

Kroeger, T., & Wahl, F. (2010). Online trajectory generation: basic concepts for instantaneous reactions to unforeseen events. *IEEE Transactions on Robotics*, *26*, 94–111.

Macfarlane, S., & Croft, E. (2003). Jerk-bounded manipulator trajectory planning: design for real-time applications. *IEEE Transactions on Robotics and Automation*, *19*(*1*), 42–52.

Mansard, N., Khatib, O., & Kheddar, A. (2009). A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Transactions on Robotics*, *25*(*3*), 670–685.

Mayne, D., Seron, M., & Rakovi, S. (2005). Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica*, *41*(*2*), 219–224.

Quinlan, S., & Khatib, O. (1993). Elastic bands: connecting path planning and control. *In IEEE International Conference on Robotics and Automation*, ICRA.

Samson, C., Espiau, B., & Borgne, M. L. (1991). *Robot Control: The Task Function Approach*. Oxford University Press.

Scheint, M., Wolff, J., & Buss, M. (2008). Invariance control in robotic applications: Trajectory supervision and haptic rendering. *In American Control Conference*, 2008.

Schwarm, A. T., & Nikolaou, M. (1999). Chance-constrained model predictive control. *AIChE Journal*, *45*(*8*), 1743–1752.

Scokaert, P. O., Mayne, D. Q., & Rawlings, J. B. (1999). Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, *44*(*3*), 648–654.

Siciliano, B., & Slotine, J.J.E. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. *In Advanced Robotics, 1991. Robots in Unstructured Environments*, 91 ICAR., In Proceedings of the Fifth International Conference on, pp. 1211–1216 *vol. 2*.

Siciliano, B. (1990). A closed-loop inverse kinematic scheme for on-line joint-based robot control. *Robotica*, *8*(*03*), 231–243.

Simetti, E., & Casalino, G. (2016). A novel practical technique to integrate inequality control objectives and task transitions in priority based control. *Journal of Intelligent & Robotic Systems*, 1–26.

Tazaki, Y., & Suzuki, T. (2014). Constraint-based prioritized trajectory planning for multibody systems. *IEEE Transactions on Robotics*, *30*(*5*), 1227–1234.

Zanchettin, A.M., & Rocco, P., (2013). Near time-optimal and sensor-based motion planning for robotic manipulators. *In IEEE Conference on Decision and Control*, CDC.

Zanchettin, A.M., & Rocco, P., (2015). Reactive motion planning and control for compliant and constraint-based task execution. *In IEEE International Conference on Robotics and Automation*, ICRA.

Zanchettin, A. M., Bascetta, L., & Rocco, P. (2013). Acceptability of robotic manipulators in shared working environments through human-like redundancy resolution. *Applied Ergonomics*, *44*(*6*), 982–989.

Zanchettin, A. M., Bascetta, L., & Rocco, P. (2013). Achieving humanlike motion: resolving redundancy for anthropomorphic industrial manipulators. *IEEE Robotics and Automation Magazine*, *20*(*4*), 131–138.