Whole-Body Control With (Self) Collision Avoidance Using Vector Field Inequalities

Juan José Quiroz-Omaña and Bruno Vilhena Adorno, *Senior Member, IEEE*

***Abstract*—This letter uses vector field inequalities (VFI) to prevent robot self-collisions and collisions with the workspace. Differently from previous approaches, the method is suitable for both velocity and torque-actuated robots. We propose a new distance function and its corresponding Jacobian in order to generate a VFI to limit the angle between two Plücker lines. This new VFI is used to prevent both undesired end-effector orientations and violation of joints limits. The proposed method is evaluated in a realistic simulation and on a real humanoid robot, showing that all constraints are respected while the robot performs a manipulation task.**

***Index Terms*—Collision avoidance, humanoid robots, quadratic programming, robot control.**

I. Introduction

COLLISION avoidance has been addressed either in off-line or on-line approaches. The former is based on motion planning, where probabilistic methods have been widely used [1], [2]. These strategies are usually applied in the configuration space and are computationally expensive, free of local minima and used in structured scenarios [3]. The latter is based on reactive methods and usually require less computation time; therefore, they can be used in real-time feedback control and are suitable for applications within unstructured workspaces.

Reactive methods are, in general, based on minimization problems [4], which exploit the robot redundancy by selecting admissible control inputs based on a specified criterion. When the robot is commanded by joint velocity inputs and operates under relatively low velocities and accelerations, its behavior is appropriately described by the kinematic model and the minimization can be performed in the joint velocities. In that case, since the control law is based entirely on the kinematic model, it is not affected by uncertainties in the inertial parameters (e.g., mass and moment of inertia). On the other hand, if the robot is commanded by torque inputs, the minimization is performed in the joint torques, which usually requires the robot dynamic model. Both methods are widely used to perform whole-body control with reactive behavior.

Whole-body control strategies with collision avoidance usually have been handled by using the task priority framework, where the overall task is divided into subtasks with different priorities. For instance, distance functions with continuous gradients between convex hulls (or between simple geometrical primitives such as spheres and cylinders) that represent the body parts are used and the lower-priority collision-avoidance tasks are satisfied in the null space of higher-priority ones [5], [6]. Those secondary tasks are fulfilled as long as they are not in conflict with the higher-priority ones. Therefore, they do not prevent collisions when in conflict with the primary task. One way to circumvent this problem is to place the collision avoidance as the higher priority task [7], at the expense of not guaranteeing the fulfillment of the main task, such as reaching targets with the end-effector. This can be addressed by using a dynamic task prioritization [8], where the control law is blended continuously between the collision-avoidance task and the end-effector pose control task as a function of the collision distance [9]. Dietrich *et al.* [10] propose a torque-based self-collision avoidance also using dynamic prioritization, where the transitions are designed to be continuous and comply with the robot's physical constraints, such as the limits on joint torque derivatives. However, changing priorities to enforce inequality constraints has an exponential cost in the number of inequalities [11]. Alternatively, non-hierarchical formulations based on weighted least-square solutions are used to solve the problem of constrained closed-loop kinematics in the context of self-collision avoidance [12]. However, the non-hiearchical approach requires an appropriate weighting matrix that results in a collision-free motion,which may not work in general, specially for high-speed motions [13]. That can be solved by combining the weighting matrix with the virtual surface method, which redirects the collision points along a virtual surface surrounding the robot links at the expense of potentially disturbing the trajectory tracking when the distance between collidable parts is smaller than a critical value.

Other strategies address the collision-free motion generation problem explicitly by using mathematical programming, which allows dealing with inequality constraints directly in the optimization formulation, providing an efficient and elegant

Manuscript received February 24, 2019; accepted June 28, 2019. Date of publication July 15, 2019; date of current version August 8, 2019. This letter was recommended for publication by Associate Editor G. Palli and Editor P. Rocco upon evaluation of the reviewers' comments. This work was supported in part by the Brazilian agencies CAPES, CNPq under Grants 424011/2016-6 and 303901/2018-7, in part by FAPEMIG, and in part by the INCT (National Institute of Science and Technology) under Grant CNPq (Brazilian National Research Council) 465755/2014-3. *(Corresponding author: Bruno Vilhena Adorno.)*

J. J. Quiroz-Omaña is with the Graduate Program in Electrical Engineering, Federal University of Minas Gerais, 31270-901 Belo Horizonte-MG, Brazil (e-mail: juanjqo@ufmg.br).

B. V. Adorno is with the Department of Electrical Engineering, Federal University of Minas Gerais, 31270-901 Belo Horizonte-MG, Brazil (e-mail: adorno@ufmg.br).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors. The video shows the effectiveness of the proposed control laws based on first and second-order vector field inequalities. Contact: adorno@ufmg.br for further questions about this letter.

Digital Object Identifier 10.1109/LRA.2019.2928783

solution, where all constraints are clearly separated from the main task. Analytical solutions, however, usually do not exist and numeric solvers must be used [14]–[16].

Marinho *et al.* [17] propose active constraints based on VFI to deal with collision avoidance in surgical applications. Both robot and obstacles are modeled by using geometric primitives—such as points, planes, and Plücker lines—, and distance functions with their respective Jacobian matrices are computed from these geometric primitives.

Our work extends the VFI method, which was first proposed using first order kinematics [17], to use second order kinematics. This enables applications that use the robot dynamics by means of the relationship between joint torques and joint accelerations in the Euler-Lagrange equations. Furthermore, we propose a new distance function related to the angle between two Plücker lines and the corresponding Jacobian matrix to prevent violations of joint limits and avoid undesired end-effector orientations. Differently from other approaches, such as [18], our approach has a formal proof showing that the second-order inequality constraints for collision avoidance always prevent collisions between any pair of primitives.

II. Task-Space Control Using Quadratic Programming

A classic strategy used in differential inverse kinematic problems consists in solving an optimization problem that minimizes the joint velocities, $\dot{\boldsymbol{q}} \in \mathbb{R}^n$, in the l_2-norm sense. Given a desired task $\boldsymbol{x}_d \in \mathbb{R}^m$, where $\dot{\boldsymbol{x}}_d = \boldsymbol{0}$, $\forall t$, and the task error $\tilde{\boldsymbol{x}} \triangleq \boldsymbol{x} - \boldsymbol{x}_d$, the control input $\boldsymbol{u}$ is obtained as

$$\begin{aligned} \boldsymbol{u} \in \underset{\dot{\boldsymbol{q}}}{\arg\min} \quad & \left\| \boldsymbol{J}\dot{\boldsymbol{q}} + \eta\tilde{\boldsymbol{x}} \right\|_2^2 + \lambda^2 \left\| \dot{\boldsymbol{q}} \right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{W}\dot{\boldsymbol{q}} \leq \boldsymbol{w}, \end{aligned} \tag{1}$$

where $\boldsymbol{J} \in \mathbb{R}^{m \times n}$ is the task Jacobian, $\lambda \in [0, \infty)$ is a damping factor, and $\boldsymbol{W} \in \mathbb{R}^{l \times n}$ and $\boldsymbol{w} \in \mathbb{R}^l$ are used to impose linear constraints in the control inputs.

An analogous scheme can be used to perform the minimization at the joint acceleration level. Given the desired error dynamics $\ddot{\tilde{\boldsymbol{x}}} + k_d\dot{\tilde{\boldsymbol{x}}} + k_p\tilde{\boldsymbol{x}} = \boldsymbol{0}$, with $k_d, k_p \in (0, \infty)$ such that $k_d^2 - 4k_p > 0$ to obtain a non-oscillatory exponential error decay, the control input $\boldsymbol{u}_a$ is obtained as

$$\begin{aligned} \boldsymbol{u}_a \in \underset{\ddot{\boldsymbol{q}}}{\arg\min} \quad & \left\| \boldsymbol{J}\ddot{\boldsymbol{q}} + \boldsymbol{\beta} \right\|_2^2 + \lambda^2 \left\| \ddot{\boldsymbol{q}} \right\|_2^2 \\ \text{subject to} \quad & \boldsymbol{\Lambda}\ddot{\boldsymbol{q}} \leq \boldsymbol{\zeta}, \end{aligned} \tag{2}$$

where $\boldsymbol{\beta} = (k_d\boldsymbol{J} + \dot{\boldsymbol{J}})\dot{\boldsymbol{q}} + k_p\tilde{\boldsymbol{x}}$, $\boldsymbol{\Lambda} \triangleq \begin{bmatrix} \boldsymbol{I} & -\boldsymbol{I} & \boldsymbol{W}^T \end{bmatrix}^T$ and $\boldsymbol{\zeta} \triangleq \begin{bmatrix} \boldsymbol{\gamma}_u^T, -\boldsymbol{\gamma}_l^T, \boldsymbol{w}^T \end{bmatrix}^T$, with $\boldsymbol{\Lambda} \in \mathbb{R}^{(2n+l)\times n}$ and $\boldsymbol{\zeta} \in \mathbb{R}^{2n+l}$. Analogously to (1), $\boldsymbol{W}$ and $\boldsymbol{w}$ are used to impose arbitrary linear constraints in the acceleration inputs, and two additional constraints, that is $\boldsymbol{\gamma}_l \leq \ddot{\boldsymbol{q}} \leq \boldsymbol{\gamma}_u$, are imposed to minimize the joints velocities by limiting the joints accelerations. More specifically, we define the acceleration lower bound $\boldsymbol{\gamma}_l$ and the acceleration upper bound $\boldsymbol{\gamma}_u$, respectively:

$$\boldsymbol{\gamma}_l \triangleq k\left(-\boldsymbol{1}_n g\left(\dot{\tilde{\boldsymbol{x}}}\right) - \dot{\boldsymbol{q}}\right), \quad \boldsymbol{\gamma}_u \triangleq k\left(\boldsymbol{1}_n g\left(\dot{\tilde{\boldsymbol{x}}}\right) - \dot{\boldsymbol{q}}\right), \tag{3}$$

where $k \in [0, \infty)$ is used to scale the feasible region, $g : \mathbb{R} \to [0, \infty)$ is a positive definite nondecreasing function (e.g., $g\left(\dot{\tilde{\boldsymbol{x}}}\right) \triangleq \left\|\dot{\tilde{\boldsymbol{x}}}\right\|_2$) and $\boldsymbol{1}_n$ is a n-dimensional column vector composed of ones. As the bounds (3) depend on the error velocity $\dot{\tilde{\boldsymbol{x}}}$, then $\boldsymbol{\gamma}_l \to \boldsymbol{\gamma}_u$ when $\dot{\tilde{\boldsymbol{x}}} \to \boldsymbol{0}$. Therefore, $\boldsymbol{\gamma}_l = \boldsymbol{\gamma}_u = \boldsymbol{\gamma}$ and $\boldsymbol{\gamma} \leq \ddot{\boldsymbol{q}} \leq \boldsymbol{\gamma}$ becomes $\ddot{\boldsymbol{q}} = \boldsymbol{\gamma} = -k\dot{\boldsymbol{q}}$, whose solution is given by $\dot{\boldsymbol{q}}(t) = \dot{\boldsymbol{q}}(0)\exp(-kt)$; that is, as the task velocity goes to zero, the robot stops accordingly.[1]

If the robot is commanded by means of torque inputs (i.e., $\boldsymbol{\tau} \triangleq \boldsymbol{u}_\tau$), we use (2) to compute $\boldsymbol{u}_a$ and the Euler-Langrange equation $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{n} = \boldsymbol{\tau}$, where $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is the inertia matrix and $\boldsymbol{n} \in \mathbb{R}^n$ denotes the nonlinear terms including Coriolis and gravity forces, to compute the control input

$$\boldsymbol{u}_\tau = \boldsymbol{n} + \boldsymbol{M}\boldsymbol{u}_a. \tag{4}$$

III. Vector Field Inequalities

The VFI framework is composed of differential inequalities that are used to avoid collisions between pairs of geometrical primitives [17], [19]. It requires a signed distance function $d(t) \in \mathbb{R}$ between two collidable objects and the Jacobian matrix $\boldsymbol{J}_d$ relating the robot joint velocities with the time derivative of the distance; that is, $\dot{d}(t) = \boldsymbol{J}_d\dot{\boldsymbol{q}}$.

In order to keep the robot outside a collision zone, the error distance is defined as $\tilde{d}(t) \triangleq d(t) - d_{\text{safe}}$, where $d_{\text{safe}} \in [0, \infty)$ is an arbitrary constant safe distance, and the following inequalities must hold for all t [17]:

$$\dot{\tilde{d}}(t) \geq -\eta_d\tilde{d}(t) \iff -\boldsymbol{J}_d\dot{\boldsymbol{q}} \leq \eta_d\tilde{d}(t), \tag{5}$$

where $\eta_d \in [0, \infty)$ is used to adjust the approach velocity. The lower is η_d, the lower is the approach velocity.

Alternatively, if we consider acceleration inputs, the VFI can be extended by means of a second-order differential inequality [20]

$$\ddot{\tilde{d}}(t) \geq -\eta_1\dot{\tilde{d}}(t) - \eta_2\tilde{d}(t) \iff -\boldsymbol{J}_d\ddot{\boldsymbol{q}} \leq \beta_d, \tag{6}$$

where $\beta_d = (\eta_1\boldsymbol{J}_d + \dot{\boldsymbol{J}}_d)\dot{\boldsymbol{q}} + \eta_2\tilde{d}(t)$, and $\eta_1, \eta_2 \in [0, \infty)$ are used to adjust the approach acceleration. We enforce the condition $\eta_1^2 - 4\eta_2 > 0$ to obtain, in the worst case, a non-oscillatory exponential approach.

Analogously, if it is desired to keep the robot inside a safe region (i.e., $\tilde{d}(t) \leq 0$), the constraints (5) and (6) are rewritten, respectively, as

$$\boldsymbol{J}_d\dot{\boldsymbol{q}} \leq -\eta_d\tilde{d}(t), \qquad \boldsymbol{J}_d\ddot{\boldsymbol{q}} \leq -\beta_d. \tag{7}$$

In order to show that the inequality in (6) prevents collisions with static obstacles—i.e., $\tilde{d}(t) \geq 0$, $\forall t \in [0, \infty)$—in robot commanded by means of acceleration inputs,[2] we propose the following lemma.

Lemma 1: Consider the inequality constraint (6), with $\eta_1, \eta_2 \in (0, \infty)$, $\eta_1^2 - 4\eta_2 > 0$, $\tilde{d}(0) \triangleq \tilde{d}_0 \in (0, \infty)$,

[1] Those bounds are necessary because (2) minimizes the joints *accelerations*. Without them, the objective function can be minimized even if the joints velocities are not null.

[2] Actual robots are usually commanded by means of velocity or torque inputs. In the latter case, acceleration inputs are transformed into torque inputs by using (4). Furthermore, we assume that the actuators do not work in saturation in order to ensure feasibility of (6) and (7).

$\dot{\tilde{d}}(0) \triangleq \dot{\tilde{d}}_0 \in (-\infty, \infty)$ and $\eta_1 \geq 2\left|\dot{\tilde{d}}_0\right| / \tilde{d}_0$. Let $\tilde{d}(t)$ be a smooth function for all $t \in [0, \infty)$, then $\tilde{d}(t) > 0, \forall t \in [0, \infty)$.

Proof: First consider $\ddot{\tilde{d}}(t) = -\eta_1 \dot{\tilde{d}}(t) - \eta_2 \tilde{d}(t)$, whose solution is given by $\tilde{d}(t) = (r_1 - r_2)^{-1}(c_1 \exp(r_1 t) + c_2 \exp(r_2 t))$, where $c_1 = \dot{\tilde{d}}_0 - \tilde{d}_0 r_2$, $c_2 = \tilde{d}_0 r_1 - \dot{\tilde{d}}_0$ and $r_1, r_2 \in (-\infty, 0)$ are the roots of the characteristic equation. Using the Comparison Lemma[3] [21], it can be shown that the solution of (6) $\forall t \in [0, \infty)$ is $d(t) \geq (r_1 - r_2)^{-1}(c_1 \exp(r_1 t) + c_2 \exp(r_2 t))$.

Both $f_1(t) \triangleq \exp(r_1 t)$ and $f_2 \triangleq \exp(r_2 t)$ are decreasing monotonic functions where $f_2(t)$ decreases at a higher rate than $f_1(t)$ because $r_1, r_2 \in (-\infty, 0)$ and $r_2 < r_1$. Therefore, by inspection, $\tilde{d}(t) \geq 0, \quad \forall t \in [0, \infty)$ if $c_1 \geq 0$ and $c_1 \geq |c_2|$, which is satisfied when $\dot{\tilde{d}}_0 \geq \tilde{d}_0 r_2$ and $\dot{\tilde{d}}_0 \geq \tilde{d}_0 (r_1 + r_2)/2$. Because $\dot{\tilde{d}}_0 \geq \tilde{d}_0 (r_1 + r_2)/2 \geq \tilde{d}_0 r_2$ and $\eta_1 = -(r_1 + r_2)$, then $\eta_1 \geq -2\dot{\tilde{d}}_0 / \tilde{d}_0$. Since $2|\dot{\tilde{d}}_0|/\tilde{d}_0 \geq -2\dot{\tilde{d}}_0/\tilde{d}_0$ then we choose $\eta_1 \geq 2|\dot{\tilde{d}}_0|/\tilde{d}_0$, which concludes the proof. ■

Since we consider arbitrary values for $\dot{\tilde{d}}_0$, additional bounds on the accelerations (i.e., $\ddot{\boldsymbol{q}}_{\min} \leq \ddot{\boldsymbol{q}} \leq \ddot{\boldsymbol{q}}_{\max}$) may result in the unfeasibility of constraints (6) and (7). Therefore, existing techniques [22] may be adapted to determine the maximum bounds on the accelerations such that the VFI can still be satisfied.

IV. Distance Functions and Jacobians

Each VFI requires a distance function between two geometric primitives and the Jacobian that relates the robot joint velocities to the time-derivative of the distance function. Marinho *et al.* [17] presented some useful distance functions and their corresponding Jacobians based on pair of geometric primitives composed of Plücker lines, planes, and points.

In some applications, it is advantageous to define target regions, instead of one specific position and/or orientation, in order to relax the task and, therefore, release some of the robot degrees of freedom (DOF) for additional tasks. For instance, if the task consists in carrying a cup of water from one place to another, one way to perform it is by defining a time-varying trajectory that determines the robot end-effector pose at all times, which requires six DOF. An alternative is to define a target position to where the cup of water should be moved (which requires three DOF) while ensuring that the cup is not too tilted to prevent spilling, which require one more DOF (i.e., the angle between a vertical line and the line passing through the center of the cup, as shown in Fig. 1). The task can be further relaxed by using only the distance to the final position, which requires only one DOF, plus the angle constraint, which requires, when activated, one more DOF.

The idea of using a conic constraint, which is equivalent to constraining the angle between two intersecting lines, was first proposed by Gienger *et al.* [23]. However, their description is singular when the angle equals $k\pi$, for all $k \in \mathbb{Z}$. Therefore, we propose a singularity-free conic constraint based on VFI. Since this constraint requires a new Jacobian, namely the line-static-line angle Jacobian, we briefly review the line Jacobian [17].

[3]Let $\dot{u} = f(t, u), \dot{v} \leq f(t, v)$, where f is continuous in t and in the functions $u(t)$ and $v(t)$, and $v_0 \leq u_0$; then $v(t) \leq u(t)\ \forall t \in [0, T)$. Because it is possible to reduce any n^{th} order linear differential equation to n first-order linear ones, the Comparison Lemma, which is defined to first order systems, can be extended to higher order linear systems in a straightforward way.

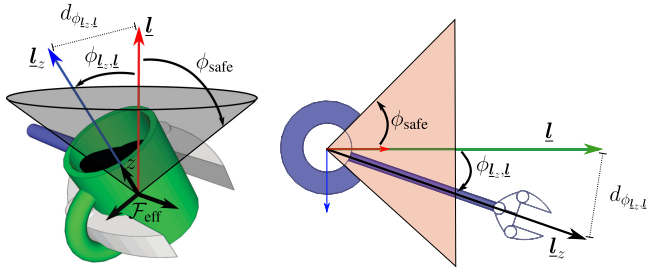

Fig. 1. Applications of the line-static-line-angle Jacobian. On the *left*, the constraint is used to keep a bounded inclination of the end-effector with respect to a vertical line passing through the origin of the end-effector frame. On the *right*, the constraint is used to impose joint limits. The angle $\phi_{\underline{l}_z,\underline{l}}$ between two Plücker lines is related to the distance $d_{\phi_{\underline{l}_z,\underline{l}}} = f(\phi_{\underline{l}_z,\underline{l}})^{1/2}$ by means of the law of cosines.

A. Line Jacobian

Given a frame $\mathcal{F}_z$ attached to the robot kinematic chain, let $\underline{\boldsymbol{l}}_z$ be a dynamic Plücker line collinear to the z-axis of $\mathcal{F}_z$. The line $\underline{\boldsymbol{l}}_z$ and its time derivative $\dot{\underline{\boldsymbol{l}}}_z$ are described with respect to the inertial frame $\mathcal{F}$ as [17]

$$\underline{\boldsymbol{l}}_z = \begin{bmatrix} \boldsymbol{l}_z \\ \boldsymbol{m}_z \end{bmatrix} \in \mathbb{R}^6, \qquad \dot{\underline{\boldsymbol{l}}}_z = \begin{bmatrix} \boldsymbol{J}_{l_z} \\ \boldsymbol{J}_{m_z} \end{bmatrix} \dot{\boldsymbol{q}} \triangleq \boldsymbol{J}_{\underline{l}_z} \dot{\boldsymbol{q}}. \tag{8}$$

where $\boldsymbol{l}_z \in \mathbb{R}^3$, with $\|\boldsymbol{l}_z\|_2 = 1$, is the line direction, and $\boldsymbol{p}_z \times \boldsymbol{l}_z = \boldsymbol{m}_z \in \mathbb{R}^3$ is the line moment, with $\boldsymbol{p}_z \in \mathbb{R}^3$ being an arbitrary point on the line.

B. Line-Static-Line-Angle Jacobian, $\boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}$

Since the angle between $\underline{\boldsymbol{l}}_z$ and the static Plücker line $\underline{\boldsymbol{l}} = \begin{bmatrix} \boldsymbol{l}^T & \boldsymbol{m}^T \end{bmatrix}^T$ (i.e., $\dot{\underline{\boldsymbol{l}}} = \boldsymbol{0}, \ \forall t$), which is given by $\phi_{\underline{l}_z,\underline{l}} = \arccos(\langle \boldsymbol{l}_z, \boldsymbol{l} \rangle)$, has a singular time derivative when $\langle \boldsymbol{l}_z, \boldsymbol{l} \rangle = \pm 1$, the function $f : [0, \pi] \to [0, 4]$ is proposed instead:

$$f(\phi_{\underline{l}_z,\underline{l}}) \triangleq \|\boldsymbol{l}_z - \boldsymbol{l}\|_2^2 = 2 - 2\cos\phi_{\underline{l}_z,\underline{l}} = (\boldsymbol{l}_z - \boldsymbol{l})^T (\boldsymbol{l}_z - \boldsymbol{l}). \tag{9}$$

As $f(\phi_{\underline{l}_z,\underline{l}})$ is a continuous bijective function, controlling the distance function (9) is equivalent to controlling the angle $\phi_{\underline{l}_z,\underline{l}} \in [0, \pi]$.

The time derivative of (9) is given by

$$\frac{d}{dt} f(\phi_{\underline{l}_z,\underline{l}}) = 2(\boldsymbol{l}_z - \boldsymbol{l})^T \frac{d}{dt}(\boldsymbol{l}_z - \boldsymbol{l}) = \underbrace{2(\boldsymbol{l}_z - \boldsymbol{l})^T \boldsymbol{J}_{l_z}}_{\boldsymbol{J}_{\phi_{\underline{l}_z,\underline{l}}}} \dot{\boldsymbol{q}}. \tag{10}$$

V. (Self)-Collision Avoidance Constraints

As shown in Fig. 1, the line-static-line-angle constraint is used to prevent violation of joint limits (*right*) and also to avoid undesired end-effector orientations (*left*). In order to prevent violation of joint limits, for each joint we place a line, $\underline{\boldsymbol{l}}$, perpendicular to the joint rotation axis and in the middle of its angular

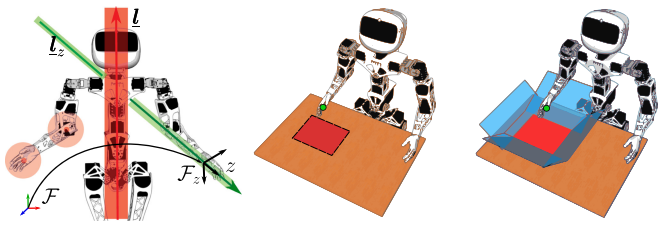

Fig. 2. On the *left*, the tobot description using geometric primitives. The torso and the forearms are modeled with infinite cylinders and spheres. On the *middle*, a target region is defined to the right hand. On the *right*, planes are used to constrain the right hand into an admissible space towards the target region.

displacement range. We also place a line, $\underline{\boldsymbol{l}}_z$, along the link attached to the joint; therefore, the angle distance between them is calculated by using (9). Since the angle distance is constrained between $[0, \pi]$ in any direction (the other one is equivalent to $[-\pi, 0]$), the maximum joint range limit can be defined in the interval $[0, 2\pi]$. This strategy requires one constraint per joint instead of two, as in other approaches [24], [25], because those approaches need to define constraints for both upper and lower joint limits.

Likewise, this strategy allows defining a maximum end-effector angle to avoid undesired orientations. In this case, however, the angle is obtained between a vertical line, $\underline{\boldsymbol{l}}$, passing through the origin of the end-effector frame and a line collinear with the z-axis of the end-effector frame, $\underline{\boldsymbol{l}}_z$. Given a maximum angle ϕ_{safe}, the constraint $\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}} \leq \phi_{\text{safe}}$ defines a cone whose centerline is given by $\underline{\boldsymbol{l}}$.

The corresponding first-order and second-order VFIs used to constrain the angle inside a safe cone are given, respectively, by

$$\boldsymbol{J}_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}}\dot{\boldsymbol{q}} \leq -\eta \tilde{f}\left(\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}\right), \tag{11}$$

$$\boldsymbol{J}_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}}\ddot{\boldsymbol{q}} \leq -\beta_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}}, \tag{12}$$

where $\tilde{f}(\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}) \triangleq f(\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}) - f(\phi_{\text{safe}})$ and $\beta_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}} = (\eta_1 \boldsymbol{J}_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}} + \dot{\boldsymbol{J}}_{\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}})\dot{\boldsymbol{q}} + \eta_2 \tilde{f}(\phi_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}})$.

In order to prevent self-collisions, we modeled the robot with spheres and cylinders, as shown in Fig. 2.

The line-static-line constraint [17] is used to prevent collisions between the arm and the torso, where the line $\underline{\boldsymbol{l}}_z$ is located along the torso and the line $\underline{\boldsymbol{l}}$ is placed along the forearm. In case that the distance $d_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}$ between the lines is zero but there is no collision, which could happen because lines are infinite, the constraint is disabled and a point-static-line constraint is used instead. In this case, the point located at the hand or the one located at the elbow is used, depending which one is closest to the line. The torso-arm constraint is written as

$$-\boldsymbol{J}_{\text{tarm}}\dot{\boldsymbol{q}} \leq \eta \tilde{d}_{\text{tarm}}, \tag{13}$$

where $\boldsymbol{J}_{\text{tarm}} = \boldsymbol{J}_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}$ and $\tilde{d}_{\text{tarm}} = d_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}} - d_{\text{safe}}$, if $d_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}} \neq 0$, or $\boldsymbol{J}_{\text{tarm}} = \boldsymbol{J}_{p,\underline{\boldsymbol{l}}}$ and $\tilde{d}_{\text{tarm}} = d_{p,\underline{\boldsymbol{l}}} - d_{\text{safe}}$ otherwise, where $d_{p,\underline{\boldsymbol{l}}}$ is the point-static-line distance, $\boldsymbol{J}_{p,\underline{\boldsymbol{l}}}$ and $\boldsymbol{J}_{\underline{\boldsymbol{l}}_z,\underline{\boldsymbol{l}}}$ are the point-static-line and the line-static-line Jacobians, respectively [17].

The inequality constraints in terms of the joints accelerations are written in similar way, as shown in (12), and therefore they will be omitted for the sake of conciseness.

To prevent collisions with obstacles, which are modeled with spheres, cylinders or planes, we use additional constraints. Fig. 2 shows an application based on the plane constraint, where four lateral planes are used to constrain the right hand of the robot inside a desired region. This way, it is impossible for the robot hand to reach the target region from below because the hand cannot trespass any of the four lateral planes. These four additional constraints are written as

$$-\boldsymbol{J}_{p,n_{\underline{\boldsymbol{\pi}}_i}}\dot{\boldsymbol{q}} \leq \eta \tilde{d}_{p,n_{\underline{\boldsymbol{\pi}}_i}}, \tag{14}$$

where $i = \{1, 2, 3, 4\}$ and $\tilde{d}_{p,n_{\underline{\boldsymbol{\pi}}_i}} = d_{p,n_{\underline{\boldsymbol{\pi}}_i}} - d_{\pi,\text{safe}}$, with $d_{\pi,\text{safe}}$ being the safe distance to each plane and $d_{p,n_{\underline{\boldsymbol{\pi}}_i}}$ and $\boldsymbol{J}_{p,n_{\underline{\boldsymbol{\pi}}_i}}$ are the point-static-plane distance and its Jacobian, respectively [17].

VI. Simulations and Experimental Results

To validate the proposed approach, we performed simulations on V-REP[4] and Matlab[5], and real experiments using ROS[6] and Python. We used DQ Robotics[7] for both robot modeling and the description of the geometric primitives.

We implemented two controllers based on quadratic programming, namely QP and CQP, where QP does not take into account constraints and CQP denotes the constrained case, and validated them in both simulation and in a real-time implementation on a real robot.

A. Simulation Setup

The goal is to control the left-hand of a humanoid robot to put a cup on a table while keeping the cup tilting below threshold (constraint (11)), avoiding reaching joint limits (one constraint (11) for each joint) and preventing self-collisions (constraint (13)) and collisions with the workspace (constraint (14)), as shown in Fig. 6. To accomplish this goal, the control law minimizes the distance between the left-hand and the table, which is modeled as a plane, and the control inputs take into account the kinematic chain composed of the 5-DOF torso and the 4-DOF left arm, which has a total of nine DOF. The safe angle, $\phi_{\text{safe}} = 0.1$ rad, denotes the cup maximum tilting angle, as shown in Fig. 1. We defined four additional planes, as shown in Fig. 2, to prevent reaching the table from below, where $d_{\pi,\text{safe}} = 0.01$ for all four planes. Furthermore, the safe distance between the arm and the torso is $d_{\text{safe}} \triangleq d_{\text{tarm}} = 0.07$. The same setup was used for the simulation using velocity inputs and the one using torque inputs.

B. Task Space Control: Joint Velocity Inputs

In this simulation, for both QP and CQP, the objective function is defined as in (1), where $\tilde{\boldsymbol{x}} \triangleq d_{p,n_{\underline{\boldsymbol{\pi}}}}$ is the distance between the

[4]http://www.coppeliarobotics.com/
[5]https://www.mathworks.com/
[6]https://www.ros.org/
[7]https://dqrobotics.github.io/

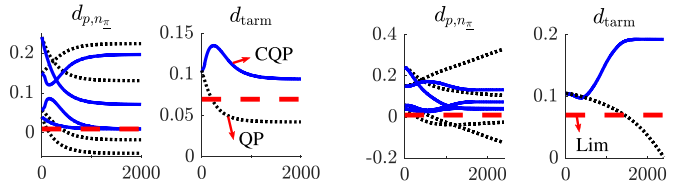

Fig. 3. Control of the left-hand in simulation. On the *left*, the robot kinematics is used with first-order VFI. On the *right*, the robot dynamics is used with second-order VFI.

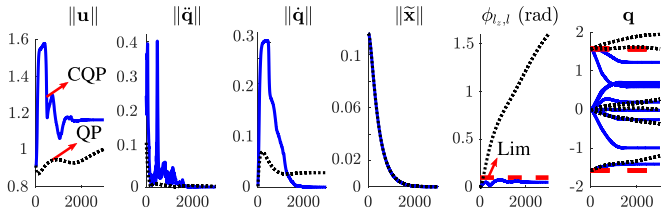

Fig. 4. Control of the left-hand using the robot dynamics and second-order VFI in simulation. The *dashed-black* line corresponds to QP and the *solid-blue line* denotes the CQP. From *left* to *right*: norm of the control inputs (torques); norm of the joint accelerations; norm of the joint velocities; norm of the task error; angle $\phi_{\underline{l}_z,\underline{l}}$; and the joints positions of both torso and left arm. The *red-dashed line* in the last two graphs correspond to the maximum allowable tilting angle ϕ_{safe} and the joints limits, respectively. The horizontal axis corresponds to iterations.

end-effector and the table, and $\boldsymbol{J} \triangleq \boldsymbol{J}_{p,n_{\underline{\pi}}}$ is the corresponding point-static-plane Jacobian matrix. Furthermore, the convergence parameter ($\eta = 0.36$) and the damping factor ($\lambda = 0.01$) were defined empirically in order to provide a smooth and fast enough convergence rate.

In the constrained case (CQP), the control inputs $\boldsymbol{u} \triangleq \dot{\boldsymbol{q}}$ are computed using (1) with the aforementiond task-Jacobian and error function, where $\boldsymbol{W} = [\boldsymbol{J}^T_{\phi_{\underline{l}_z,\underline{l}}} - \boldsymbol{J}^T_{p,n_{\underline{\pi}_\mathcal{I}}} \boldsymbol{J}^T_{\phi_{\underline{l}_{z\mathcal{I}},\underline{l}_\mathcal{I}}} - \boldsymbol{J}^T_{\text{tarm}}]^T$ with

$$\boldsymbol{J}_{p,n_{\underline{\pi}_\mathcal{I}}} = \left[\boldsymbol{J}^T_{p,n_{\underline{\pi}_1}} \cdots \boldsymbol{J}^T_{p,n_{\underline{\pi}_4}}\right]^T, \tag{15}$$

$$\boldsymbol{J}_{\phi_{\underline{l}_{z\mathcal{I}},\underline{l}_\mathcal{I}}} = \left[\boldsymbol{J}^T_{\phi_{\underline{l}_{z_1},\underline{l}_1}} \cdots \boldsymbol{J}^T_{\phi_{\underline{l}_{z_9},\underline{l}_9}}\right]^T, \tag{16}$$

and $\boldsymbol{w} = \eta[-\tilde{f}\left(\phi_{\underline{l}_z,\underline{l}}\right) \tilde{\boldsymbol{d}}^T_{p,n_{\underline{\pi}_\mathcal{I}}} - \tilde{\boldsymbol{F}}^T \tilde{d}_{\text{tarm}}]^T$ with

$$\tilde{\boldsymbol{d}}_{p,n_{\underline{\pi}_\mathcal{I}}} = \left[\tilde{d}_{p,n_{\underline{\pi}_1}} \cdots \tilde{d}_{p,n_{\underline{\pi}_4}}\right]^T, \tag{17}$$

$$\tilde{\boldsymbol{F}} = \left[\tilde{f}\left(\phi_{\underline{l}_{z_1},\underline{l}_1}\right) \cdots \tilde{f}\left(\phi_{\underline{l}_{z_9},\underline{l}_9}\right)\right]^T. \tag{18}$$

Eq. (15) and (17) are used to enforce the four plane constraints (recall (14)), whereas (16) and (18) are used to avoid joint limits in all nine joints.

Figure 6*A* shows the simulation snapshots. Although the task is fulfilled for both controllers, only CQP prevents undesired cup orientations and respects all other constraints, as shown in Fig. 3.

C. Task Space Control: Joint Torque Inputs

We performed a second simulation using the robot dynamic model with second-order VFI. V-REP was used only as a visualization and collision-checking tool. We assume perfect knowledge of the inertial parameters, which are obtained directly from V-REP. The minimization is performed in the joint accelerations using formulation (2), where $\boldsymbol{J}$ and $\tilde{\boldsymbol{x}}$, which are needed to obtain $\boldsymbol{\beta}$, are the same as in Section VI-B. In addition, all constraints are rewritten as in (6) and (7) to obtain $\boldsymbol{W}$ and $\boldsymbol{w}$, where $\boldsymbol{W}$ is exactly the same as in the first-order case and $\boldsymbol{w} = [-\beta_{\phi_{\underline{l}_z,\underline{l}}} \boldsymbol{\beta}^T_{p,n_{\underline{\pi}_\mathcal{I}}} - \boldsymbol{\beta}^T_{\text{joints}} \beta_{\text{tarm}}]^T$, with $\boldsymbol{\beta}_{p,n_{\underline{\pi}_\mathcal{I}}} = [\beta_{p,n_{\underline{\pi}_1}} \cdots \beta_{p,n_{\underline{\pi}_4}}]^T$ and $\boldsymbol{\beta}_{\text{joints}} = [\beta_{\phi_{\underline{l}_{z_1},\underline{l}_1}} \cdots \beta_{\phi_{\underline{l}_{z_9},\underline{l}_9}}]^T$. Furthermore, $k_d = \eta_1 = 1.5$, $k_p = \eta_2 = 0.3$, and $g\left(\dot{\tilde{\boldsymbol{x}}}\right) \triangleq \|\dot{\tilde{\boldsymbol{x}}}\|_2$. Finally, the control inputs $\boldsymbol{u} \triangleq \boldsymbol{\tau}$ are computed using (4).

Fig. 4 shows that both QP and CQP minimize the task error and the joint accelerations, as expected. The joint accelerations in CQP, however, are more abrupt because the collision-avoidance constraints enforce abrupt changes in the robot velocities to prevent collisions. Because of that, CQP required higher control effort, as shown by larger values of $\|\boldsymbol{u}\|$ for CQP in Fig. 4.

Since CQP enforces the constraints $\boldsymbol{\gamma}_l \leq \ddot{\boldsymbol{q}} \leq \boldsymbol{\gamma}_u$, where $\boldsymbol{\gamma}_l$ and $\boldsymbol{\gamma}_u$ are calculated according to (3), the joint velocities go to zero as the end-effector stops. As QP does not enforce those constraints, the robot joints continue to move after the end-effector stops, as shown in Fig. 4.

Finally, the cup maximum tilting angle is respected, as well as the other constraints for collision avoidance and joint limit avoidance, only when CQP is used, as shown in Fig. 3.

D. Experimental Results

We performed a real experiment using the platform composed of the upper body of the Poppy humanoid robot.[8] We used Python and ROS Melodic in a computer running Ubuntu 18.04 64 bits, equipped with a Intel i7 4712HQ with 16 GB RAM, in addition to quadprog[9] to solve (1). The solver required about $191\ \mu\text{s} \pm 47\ \mu\text{s}$ to generate the control inputs in the constrained case. Furthermore, it was required about $5.2\ \text{ms} \pm 1\ \text{ms}$ to compute 15 constraint Jacobians (one angle constraint, four plane constraints, one torso-left-arm constraint, and nine joint limit constraints). However, we set the loop rate control to 50 Hz due to low-level drivers limitations.

We used the same task specification and parameters defined in Sections VI-A and VI-B. All collision-avoidance constraints were activated, for both QP and CQP, to ensure the robot safety. This way, the only difference between them is that QP does not have the cup tilting angle constraint whereas CQP has. Fig. 5 shows the task error decay and the angle $\phi_{\underline{l}_z,\underline{l}}$ for both controllers. In both cases, the task is fulfilled with the same convergence rate, which is determined by η. However, only CQP respects the angle constraint and the limit joints constraints, as expected.

Figure 6*B* shows that, in both cases, the end-effector reached the target zone, but only CQP does not violate the cup tilting angle constraint, as expected.

[8]https://www.poppy-project.org/en/robots/poppy-humanoid
[9]https://pypi.org/project/quadprog/

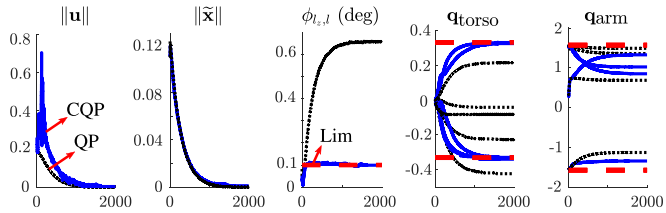

Fig. 5. Control of the left-hand using the robot kinematics and first-order VFI using the real-time implementation on the real robot. From *left* to *right*: norm of the control inputs (joint velocities); norm of the task error; the angle $\phi_{\underline{l}_z,\underline{l}}$ between the cup centerline and a vertical line; joints configurations of the torso; and the joints configurations of the left arm. The *dashed red line* denotes the maximum allowable tilting angle and the joint limits. The horizontal axis corresponds to iterations.

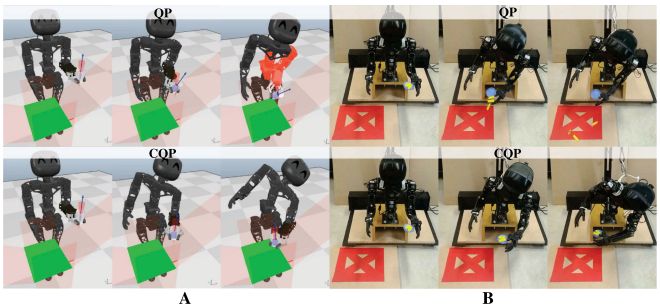

Fig. 6. Control of the left-hand using the robot kinematics and first-order VFI. ***A*** (simulation): on *top,* only the plane constraints were used, whereas at the *bottom* all constraints were used. The *blue axis* denotes the cup orientation and the *red cone* represents the maximum allowable tilting for the *blue axis*. *Red-shaded* body parts indicate a collision. ***B*** (real-time implementation on the real robot): on the *top* three snapshots, the cup tilting angle constraint is disabled, whereas on the three snapshots at the bottom this constraint is enabled. All other collision-avoidance constraints are activated to ensure the robot safety. The red square denotes the target region.

VII. Conclusions

This work extended the VFI framework, which was initially proposed using first-order kinematics, to second-order kinematics. This allows its use in applications that requires the robot dynamics by means of the relationship between the joint torques and joint accelerations in the Euler-Lagrange equations. Furthermore, we proposed a novel singularity-free conic constraint to limit the angle between two Plücker lines. This new constraint is used to prevent the violation of joint limits and/or to avoid undesired end-effector orientations.

We evaluated the proposed method, using kinematic and dynamic formulations, both in simulation and on a real humanoid robot. The results showed that the robot always avoids collisions with static obstacles, self-collisions, and undesired orientations while performing manipulation tasks. Our approach requires an environment modeled with sufficient geometric primitives and is not free of local minima, but it does provide reactive behaviors. Future works will be focused on the implementation of the VFI framework on a full humanoid to enforce balance constraints and will also consider dynamic obstacles.

References

[1] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1478–1483.

[2] F. Burget, M. Bennewitz, and W. Burgard, "BI 2 RRT*: An efficient sampling-based path planning framework for task-constrained mobile manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2016, pp. 3714–3721.

[3] M. Moll, I. A. Sucan, and L. E. Kavraki, "Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization," *IEEE Robot. Autom. Mag.*, vol. 22, no. 3, pp. 96–102, Sep. 2015.

[4] J.-P. Laumond, N. Mansard, and J. B. Lasserre, "Optimization as motion selection principle in robot action," *Commun. ACM*, vol. 58, no. 5, pp. 64–74, Apr. 2015.

[5] O. Stasse, A. Escande, N. Mansard, S. Miossec, P. Evrard, and A. Kheddar, "Real-time (self)-collision avoidance task on a HRP-2 humanoid robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, vol. 25, no. 3, pp. 3200–3205.

[6] M. Schwienbacher, T. Buschmann, S. Lohmeier, V. Favot, and H. Ulbrich, "Self-collision avoidance and angular momentum compensation for a biped humanoid robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 581–586.

[7] L. Sentis and O. Khatib, "Task-oriented control of humanoid robots through prioritization," in *Proc. IEEE Int. Conf. Humanoid Robots*, 2004, pp. 1–16.

[8] N. Mansard, O. Khatib, and A. Kheddar, "A unified approach to integrate unilateral constraints in the stack of tasks," *IEEE Trans. Robot.*, vol. 25, no. 3, pp. 670–685, Jun. 2009.

[9] H. Sugiura, M. Gienger, H. Janssen, and C. Goerick, "Real-time collision avoidance with whole body motion control for humanoid robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2007, pp. 2053–2058.

[10] A. Dietrich, T. Wimbock, A. Albu-Schaffer, and G. Hirzinger, "Integration of reactive, torque-based self-collision avoidance into a task hierarchy," *IEEE Trans. Robot.*, vol. 28, no. 6, pp. 1278–1293, Dec. 2012.

[11] O. Kanoun, F. Lamiraux, and P.-B. Wieber, "Kinematic control of redundant manipulators: Generalizing the task-priority framework to inequality task," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 785–792, Aug. 2011.

[12] R. V. Patel, F. Shadpey, F. Ranjbaran, and J. Angeles, "A collision-avoidance scheme for redundant manipulators: Theory and experiments," *J. Robot. Syst.*, vol. 22, no. 12, pp. 737–757, Dec. 2005.

[13] B. Dariush, Y. Zhu, A. Arumbakkam, and K. Fujimura, "Constrained closed loop inverse kinematics," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 2499–2506.

[14] I. Kim and J.-H. Oh, "Inverse kinematic control of humanoids under joint constraints," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 1, pp. 1–12, Jan. 2013.

[15] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *Int. J. Robot. Res.*, vol. 33, no. 7, pp. 1006–1028, Jun. 2014.

[16] V. M. Goncalves, P. Fraisse, A. Crosnier, and B. V. Adorno, "Parsimonious kinematic control of highly redundant robots," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 65–72, Jan. 2016.

[17] M. M. Marinho, B. V. Adorno, K. Harada, and M. Mitsuishi, "Active constraints using vector field inequalities for surgical robots," in *Proc. IEEE Int. Conf. Robot. Autom.,* May 2018, pp. 5364–5371.

[18] B. Dariush, G. Bin Hammam, and D. Orin, "Constrained resolved acceleration control for humanoids," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2010, pp. 710–717.

[19] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Proc. IEEE Int. Conf. Robot. Autom.*, 1987, pp. 1152–1159.

[20] K. Bouyarmane, S. Caron, A. Escande, and A. Kheddar, "Multi-contact motion planning and control," in *Humanoid Robotics: A Reference*. Dordrecht, The Netherlands: Springer, 2017, pp. 1–42.

[21] H. K. Khalil, *Nonlinear Systems*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1996.

[22] A. Del Prete, "Joint position and velocity bounds in discrete-time acceleration/torque control of robot manipulators," *IEEE Robot. Autom. Lett.*, vol. 3, no. 1, pp. 281–288, Jan. 2018.

[23] M. Gienger, H. Janben, and C. Goerick, "Exploiting task intervals for whole body robot control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 2484–2490.

[24] F.-T. Cheng, T.-H. Chen, and Y.-Y. Sun, "Resolving manipulator redundancy under inequality constraints," *IEEE Trans. Robot. Autom.*, vol. 10, no. 1, pp. 65–71, Feb. 1994.

[25] D. Guo and Y. Zhang, "Acceleration-level inequality-based MAN scheme for obstacle avoidance of redundant robot manipulators," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 6903–6914, Dec. 2014.