frontiers in Robotics and AI

ORIGINAL RESEARCH
published: 18 April 2016
doi: 10.3389/frobt.2016.00016

Set-Based Tasks within the Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework: General Formulation, Stability Analysis, and Experimental Results

Signe Moe[1]*, *Gianluca Antonelli*[2], *Andrew R. Teel*[3], *Kristin Y. Pettersen*[1] and *Johannes Schrimpf*[4]

[1] Center for Autonomous Marine Operations and Systems (AMOS), Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway, [2] Department of Electrical and Information Engineering, University of Cassino and Southern Lazio, Cassino, Italy, [3] Department of Electrical Engineering, University of California Santa Barbara, Santa Barbara, CA, USA, [4] Department of Engineering Cybernetics, Norwegian University of Science and Technology, Trondheim, Norway

OPEN ACCESS

Edited by:
Zoe Doulgeri, Aristotle University of Thessaloniki, Greece

Reviewed by:
Dongbin Lee, Oregon Institute of Technology, USA
Charalampos P. Bechlioulis, National Technical University of Athens, Greece

***Correspondence:**
Signe Moe
signe.moe@itk.ntnu.no

Specialty section:
This article was submitted to Robotic Control Systems, a section of the journal Frontiers in Robotics and AI

Received: *07 December 2015*
Accepted: *22 March 2016*
Published: *18 April 2016*

Citation:
Moe S, Antonelli G, Teel AR, Pettersen KY and Schrimpf J (2016) Set-Based Tasks within the Singularity-Robust Multiple Task-Priority Inverse Kinematics Framework: General Formulation, Stability Analysis, and Experimental Results. Front. Robot. AI 3:16. doi: 10.3389/frobt.2016.00016

Inverse kinematics algorithms are commonly used in robotic systems to transform tasks to joint references, and several methods exist to ensure the achievement of several tasks simultaneously. The multiple task-priority inverse kinematics framework allows tasks to be considered in a prioritized order by projecting task velocities through the null spaces of higher-priority tasks. This paper extends this framework to handle set-based tasks, i.e., tasks with a range of valid values, in addition to equality tasks, which have a specific desired value. Examples of set-based tasks are joint limit and obstacle avoidance. The proposed method is proven to ensure asymptotic convergence of the equality task errors and the satisfaction of all high-priority set-based tasks. The practical implementation of the proposed algorithm is discussed, and experimental results are presented where a number of both set-based and equality tasks have been implemented on a 6 degree of freedom UR5, which is an industrial robotic arm from Universal Robots. The experiments validate the theoretical results and confirm the effectiveness of the proposed approach.

Keywords: robotics, kinematic control, set-based control, task-priority, redundant systems

1. INTRODUCTION

The desired trajectory of robotic systems is typically given in the task space. In particular, this trajectory often describes the desired position trajectory of the end effector, which is then given in the Cartesian space. For control of robotic systems, the desired trajectory needs to be mapped from this task space to a reference trajectory in the joint space in which the actuators provide their input. In general, this topic falls within the *inverse kinematics* control problem. In most cases, the complexity of the problem requires numerical methods to solve the mapping (Siciliano et al., 2009).

Robotic systems with a large number of degrees of freedom (DOFs) are commonly used for industrial purposes (Siciliano et al., 2009) and are becoming increasingly important within a variety of domains, including humanoid robots (Escande et al., 2014) and unmanned vehicles, such as underwater (Simetti et al., 2013; Antonelli, 2014) and aerial systems (Baizid et al., 2015). Having developed beyond the structured and predictable industrial environment, robotic control algorithms are required to handle real-time trajectory generation. This requirement imposes the use of differential approaches for practical purposes, both at the kinematic and dynamic levels. In Whitney (1969), the use of the pseudoinverse is first recognized as a promising tool to solve the inverse kinematics problem at the kinematic level in robotic applications.

A robotic system is defined as *redundant* when it possesses more DOFs than those strictly required to solve its task, in which case-specific approaches need to be used (Chiaverini et al., 2008). Additional control possibilities then arise in terms of utilizing the "excess" DOFs by adding other tasks to be controlled simultaneously. This is an important possibility in unstructured or non-repetitive environments, as these require the algorithms to handle additional control objectives, such as staying within the mechanical joint limits, avoidance of obstacles, control of the orientation of directional sensors, and maximizing arm manipulability, in addition to the main control objectives. Note that joint limit tasks may be considered as a special case where the task space and the configuration space are identical. Hence, frameworks that map from configuration space to task space may still be utilized for these tasks.

In a task-priority framework, the different control objectives are embedded with a priority relative to their respective importance. The importance of prioritizing between the control objectives can arise from several reasons. Safety, for example, may be considered to be of supreme importance if the environment is shared with human operators. Thus, tracking of the end-effector trajectory may actually be assigned a lower priority in the rank of importance. These considerations lead to solutions as in Liégeois (1977), where the null-space projector is considered in the solution to achieve secondary control objectives afforded by a gradient-based approach. Secondary objectives are defined and handled in a task-priority approach in Maciejewski and Klein (1985) and Nakamura et al. (1987). The work (Siciliano and Slotine, 1991) extends this approach to multiple tasks. In Chiaverini (1997), a different approach is introduced, which guarantees singularity robustness with respect to algorithmic singularities. This approach is further extended to multiple tasks in priority and analyzed in Mansard and Chaumette (2007) and Antonelli (2009), and it is referred to as the singularity-robust multiple task-priority inverse kinematics framework.

Notice that the term *hierarchy* and *priority* are often used as synonyms in the literature. In this paper, the latter will be used.

The singularity-robust multiple task-priority inverse kinematics framework has been developed for *equality tasks*, which are tasks that assign an exact desired value to the controlled variable (e.g., the end-effector configuration). However, for a general robotic system, several goals may be described as *set-based tasks*, i.e., tasks with a desired interval [*area of satisfaction* (Escande et al., 2014)]. Two classic, yet vital examples are joint limits and obstacle avoidance. Additional examples are manipulability, dexterity, and field of view (for directional sensors, such as a camera mounted on the end effector). Set-based tasks are often referred to as *inequality* or *unilateral* constraints in the literature.

As recognized in Kanoun et al. (2011), inclusion of inequality constraints is still missing for the singularity-robust multiple task-priority inverse kinematics framework. This paper aims to fill that gap. In particular, in this paper, we will present a method that allows a general number of scalar set-based tasks to be handled with a given priority within a number of equality tasks. A preliminary description of the idea is given in Antonelli et al. (2015), and Moe et al. (2015a) present some experimental results of the proposed algorithm. It is worth noticing that, with respect to similar approaches presented in the literature, this is the very first one with a formal stability analysis, presented in Moe et al. (2015b). The contribution of this paper is to streamline and present the previous conference contributions in a unified manner and to address how set-based tasks that are given lower priority in the task hierarchy should be handled. Furthermore, new experimental results are presented.

This paper is organized as follows. Section 2 presents relevant literature and alternative approaches to handle set-based tasks. Section 3 gives a brief introduction to the singularity-robust multiple task-priority inverse kinematics framework that is the basis for the presented method, and a closer definition of set-based tasks. The proposed method is described in Section 4, both for high-priority and low-priority set-based tasks and a combination of the two, and it is analyzed with respect to stability in Section 5. Finally, Section 6 considers the practical implementation of the suggested algorithm, and Section 7 presents experimental results performed on a UR5 manipulator. Conclusion and future work are given in Section 8.

2. RELEVANT LITERATURE

In Kanoun et al. (2011) and Simetti et al. (2013), it is pointed out that in order to handle set-based tasks, these are often transformed into equality constraints or un-proper potentials, which leads to over-constraining the original control problem.

One possible way to systematically handle tasks arranged in priority with set-based and equality objectives is to transform the inverse kinematics problem into a quadratic programing (QP) or a similar optimization problem (Ioannou and Sun, 1996), thereby preventing the tasks from being directly included in the multiple task-priority inverse kinematics algorithm unlike the approach suggested in this paper. One of the first solutions using this approach is given by Faverjon and Tournassoud (1987), generalized by Kanoun et al. (2011), and further improved by de Lasa et al. (2010) and Escande et al. (2014) in terms of convergence time. In de Lasa et al. (2010), however, set-based tasks can be considered only as high priority tasks. In Azimian et al. (2014), the inequality constrains are transformed to an equality constraint by the use of proper slack variables. The optimization problem is then modified in order to minimize the defined slack variables together with the task errors in a task-priority architecture.

Set-based tasks may also be embedded in the control problem by assigning virtual forces that push the robot away from the set boundaries. This idea was first proposed in Khatib (1986) and then used in several following approaches. However, when resorting to virtual forces/potentials, satisfaction of the boundaries can not be analytically guaranteed and often the overall control architecture falls within the paradigm of *cooperative behavioral control*, which is possibly affected with drawbacks described in Antonelli et al. (2008). In Simetti et al. (2013), there is a systematic method for handling set-based tasks within the framework of potential fields. In particular, the key aspect is to resort to finite-support smooth potential fields for representing set-based objectives and activation functions. Experimental results with an underwater vehicle-manipulator system are provided. During the activation of the smoothing functions, however, strict priority among the tasks is lost, which is a common drawback when smoothing functions are introduced to avoid discontinuities also in alternative approaches.

In Mansard et al. (2009), set-based tasks are handled within a framework of transitions between solutions. In particular, the transition is necessary to handle the unilateral constraint and avoid discontinuities. Using the damped least-square inverse of the Jacobian, the proposed algorithm does not respect priorities during transitions. In practice, the proposed solution is a way to handle the algorithm given in Siciliano and Slotine (1991) by resorting to homotopy.

In adaptive control, projection methods may be used to ensure that the parameter estimates are bounded by using projection methods (Ioannou and Sun, 1996). The method proposed in this paper may be considered as a type of projection method in the sense that we suggest a discontinuous solution for the system velocities based on projecting the solutions. We suggest a method to directly embed the set-based tasks into the singularity-robust multiple task-priority inverse kinematics framework, thereby making it unnecessary to rewrite the problem into an optimization problem or resort to potential fields. By including the set-based tasks directly into this framework, a stability analysis is feasible, and the paper presents a formal stability analysis with explicit assumptions. Furthermore, the strict priority of the tasks is fulfilled at all times.

3. BACKGROUND

3.1. Singularity-Robust Multiple Task-Priority Inverse Kinematics

A general robotic system has n DOFs. Its configuration is given by the joint values $\boldsymbol{q} = [q_1, q_2, \ldots, q_n]^T$. It is then possible to express tasks and task velocities in the operational space through forward kinematics and the task Jacobian matrix. Let us define $\boldsymbol{\sigma}(t) \in \mathbb{R}^m$ as the task variable to be controlled:

$$\boldsymbol{\sigma}(t) = \boldsymbol{f}(\boldsymbol{q}(t)) \tag{1}$$

with the corresponding differential relationship:

$$\dot{\boldsymbol{\sigma}}(t) = \frac{\partial \boldsymbol{f}(\boldsymbol{q}(t))}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}}(t) = \boldsymbol{J}(\boldsymbol{q}(t))\dot{\boldsymbol{q}}(t)\,, \tag{2}$$

where $\boldsymbol{J}(\boldsymbol{q}(t)) \in \mathbb{R}^{m\times n}$ is the configuration-dependent analytical task Jacobian matrix and $\dot{\boldsymbol{q}}(t) \in \mathbb{R}^n$ is the system velocity. For compactness, the argument $\boldsymbol{q}$ of tasks and Jacobians are omitted from the equations in this paper.

Let us first consider a single m-dimensional task to be followed, with a defined desired trajectory $\boldsymbol{\sigma}_{\text{des}}(t) \in \mathbb{R}^m$. The corresponding joint references $\boldsymbol{q}_{\text{des}}(t) \in \mathbb{R}^n$ for the robotic system may be computed by integrating the locally inverse mapping of equation (2) achieved by imposing minimum-norm velocity (Siciliano, 1990). The following least-squares solution is given:

$$\dot{\boldsymbol{q}}_{\text{des}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{\sigma}}_{\text{des}} = \boldsymbol{J}^{\text{T}}\left(\boldsymbol{J}\boldsymbol{J}^{\text{T}}\right)^{-1}\dot{\boldsymbol{\sigma}}_{\text{des}}\,, \tag{3}$$

where $\boldsymbol{J}^{\dagger}$, implicitly defined in the above equation for full row rank matrices, is the right pseudoinverse of $\boldsymbol{J}$. In the general case, the pseudoinverse is the matrix that satisfies the four Moore–Penrose conditions [equations (4)–(7)] (Golub and Van Loan, 1996), and it is defined for systems that are not square ($m \neq n$) nor have full rank (Buss, 2009):

$$\boldsymbol{J}\boldsymbol{J}^{\dagger}\boldsymbol{J} = \boldsymbol{J}, \tag{4}$$

$$\boldsymbol{J}^{\dagger}\boldsymbol{J}\boldsymbol{J}^{\dagger} = \boldsymbol{J}^{\dagger}, \tag{5}$$

$$(\boldsymbol{J}\boldsymbol{J}^{\dagger})^{\star} = \boldsymbol{J}\boldsymbol{J}^{\dagger}, \tag{6}$$

$$(\boldsymbol{J}^{\dagger}\boldsymbol{J})^{\star} = \boldsymbol{J}^{\dagger}\boldsymbol{J}. \tag{7}$$

Here, $\boldsymbol{J}^{\star}$ denotes the complex conjugate of $\boldsymbol{J}$.

The vector $\boldsymbol{q}_{\text{des}}$ achieved by taking the time integral of equation (3) is prone to drifting. To handle this, a closed loop inverse kinematics (CLIK) version of the algorithm is usually implemented (Chiaverini, 1997), namely,

$$\dot{\boldsymbol{q}}_{\text{des}} = \boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_{\text{des}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}) = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{\sigma}}_{\text{ref}}\,, \tag{8}$$

where $\tilde{\boldsymbol{\sigma}} \in \mathbb{R}^m$ is the task error defined as

$$\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_{\text{des}} - \boldsymbol{\sigma} \tag{9}$$

and $\boldsymbol{\Lambda} \in \mathbb{R}^{m\times m}$ is a positive-definite matrix of gains. This feedback approach reduces the error dynamics to

$$\begin{aligned}\dot{\tilde{\boldsymbol{\sigma}}} &= \dot{\boldsymbol{\sigma}}_{\text{des}} - \dot{\boldsymbol{\sigma}} = \dot{\boldsymbol{\sigma}}_{\text{des}} - \boldsymbol{J}\dot{\boldsymbol{q}} \\ &= \dot{\boldsymbol{\sigma}}_{\text{des}} - \boldsymbol{J}\boldsymbol{J}^{\dagger}(\dot{\boldsymbol{\sigma}}_{\text{des}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}) \\ &= -\boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}},\end{aligned} \tag{10}$$

if $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{des}}$ and $\boldsymbol{J}$ has full rank, implying that $\boldsymbol{J}\boldsymbol{J}^{\dagger} = \boldsymbol{I}$. Equation (10) describes a linear system with a globally exponentially stable equilibrium point at the equilibrium $\tilde{\boldsymbol{\sigma}} = \boldsymbol{0}$. It is worth noticing that the assumption $\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}_{\text{des}}$ is common to all inverse kinematics algorithms (Antonelli, 2008). For practical applications, it requires that the low-level dynamic control loop is faster than the kinematic one.

In case of system redundancy, i.e., if $n > m$, the classic general solution contains a null projector operator (Liégeois, 1977):

$$\dot{\boldsymbol{q}}_{\text{des}} = \boldsymbol{J}^{\dagger}\dot{\boldsymbol{\sigma}}_{\text{ref}} + \left(\boldsymbol{I}_n - \boldsymbol{J}^{\dagger}\boldsymbol{J}\right)\dot{\boldsymbol{q}}_{\text{null}}, \tag{11}$$

where $\boldsymbol{I}_n$ is the $(n \times n)$ identity matrix and the vector $\dot{\boldsymbol{q}}_{\text{null}} \in \mathbb{R}^n$ is an arbitrary system velocity vector. It can be recognized that the operator $(\boldsymbol{I}_n - \boldsymbol{J}^{\dagger}\boldsymbol{J})$ projects $\dot{\boldsymbol{q}}_{\text{null}}$ in the null space of the Jacobian matrix. This corresponds to generating a motion of the robotic system that does not affect that of the given task.

For highly redundant systems, multiple tasks can be arranged in priority. Let us consider three tasks that will be denoted with the subscripts 1, 2 and 3, respectively:

$$\boldsymbol{\sigma}_1 = f_1(q) \in \mathbb{R}^{m_1} \tag{12}$$

$$\boldsymbol{\sigma}_2 = f_2(q) \in \mathbb{R}^{m_2} \tag{13}$$

$$\boldsymbol{\sigma}_3 = f_3(q) \in \mathbb{R}^{m_3}. \tag{14}$$

For each of the tasks, a corresponding Jacobian matrix can be defined, denoted $J_1 \in \mathbb{R}^{m_1 \times n}$, $J_2 \in \mathbb{R}^{m_2 \times n}$, and $J_3 \in \mathbb{R}^{m_3 \times n}$, respectively. Let us further define the corresponding null-space projectors for the first two tasks as

$$N_1 = \left(I_n - J_1^\dagger J_1\right) \tag{15}$$

$$N_2 = \left(I_n - J_2^\dagger J_2\right). \tag{16}$$

The *augmented Jacobian* of tasks 1 and 2 is given by stacking the two independent task Jacobians:

$$J_{12}^{\mathrm{A}} = \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \tag{17}$$

The common null-space for tasks 1 and 2 is then defined as

$$N_{12}^{\mathrm{A}} = \left(I_n - J_{12}^{\mathrm{A}\,\dagger} J_{12}^{\mathrm{A}}\right), \tag{18}$$

where $J_{12}^{\mathrm{A}\,\dagger}$ is the pseudoinverse of J_{12}^{A} that satisfies the four Moore–Penrose conditions [equations (4)–(7)]. By expanding the expression for J_{12}^{A}, we see that

$$\begin{aligned} \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} N_{12}^{\mathrm{A}} &= \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \left(I_n - \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}^\dagger \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}\right) \\ &= \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} - \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \begin{bmatrix} J_1 \\ J_2 \end{bmatrix}^\dagger \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} \\ &= \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} - \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \end{aligned} \tag{19}$$

so in general,

$$J_i N_{12..k}^{\mathrm{A}} = \mathbf{0} \text{ for } i \in \{1, \ldots, k\}. \tag{20}$$

The following equation then defines the desired joint velocities:

$$\dot{q}_{\mathrm{des}} = \underbrace{J_1^\dagger \dot{\boldsymbol{\sigma}}_{1,\mathrm{ref}}}_{\dot{q}_{1,\mathrm{des}}} + N_1 \underbrace{J_2^\dagger \dot{\boldsymbol{\sigma}}_{2,\mathrm{ref}}}_{\dot{q}_{2,\mathrm{des}}} + N_{12}^{\mathrm{A}} \underbrace{J_3^\dagger \dot{\boldsymbol{\sigma}}_{3,\mathrm{ref}}}_{\dot{q}_{3,\mathrm{des}}} \tag{21}$$

where the definition of $\dot{\boldsymbol{\sigma}}_{x,\mathrm{ref}}$ can be easily extrapolated from equation (8) for each task with the corresponding positive-definite matrix $\boldsymbol{\Lambda}_x \in \mathbb{R}^{m_x \times m_x}$. The priority of the tasks follows the numerical order, with $\boldsymbol{\sigma}_1$ being the highest-priority task. Equation (21) also implicitly defines the joint velocities $\dot{q}_{x,\mathrm{des}} \in \mathbb{R}^n$ that represent the desired joint velocity corresponding to task $\boldsymbol{\sigma}_x$ if this was the sole task.

The generalization to k tasks is straightforward: equation (21) can be expanded as:

$$\dot{q}_{\mathrm{des}} = J_1^\dagger \dot{\boldsymbol{\sigma}}_{1,\mathrm{ref}} + N_1 J_2^\dagger \dot{\boldsymbol{\sigma}}_{2,\mathrm{ref}} + \cdots + N_{12\ldots(k-1)}^{\mathrm{A}} J_k^\dagger \dot{\boldsymbol{\sigma}}_{k,\mathrm{ref}} \tag{22}$$

where $N_{12\ldots(k-1)}^{\mathrm{A}}$ is the null space of the augmented Jacobian matrix

$$J_{12\ldots(k-1)}^{\mathrm{A}} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_{k-1} \end{bmatrix}. \tag{23}$$

3.2. Set-Based Definitions

The previous section introduced the concept of multiple task-priority inverse kinematic control for a robotic system, as a method to generate reference trajectories for the system configuration that, if satisfied, will result in the successful achievement of several tasks. However, this framework has been developed for equality tasks that have a specific desired value $\boldsymbol{\sigma}_{\mathrm{des}}(t)$, e.g., the desired end effector position. This paper proposes a method to extend the existing framework to handle set-based tasks, such as the avoidance of joint limits and obstacles, and field of view.

A set-based task is still expressed through forward kinematics equation (1), but the objective is to keep the task in a defined set D rather than controlling it to a desired value. Mathematically, this can be expressed as $\boldsymbol{\sigma}(t) \in D\ \forall t$ rather than $\boldsymbol{\sigma}(t) = \boldsymbol{\sigma}_{\mathrm{des}}(t)$. Thus, set-based tasks cannot be directly inserted into the singularity-robust multiple task-priority inverse kinematics framework [equation (22)]. In this paper, we will present a method that allows a general number of scalar set-based tasks to be handled with a given priority within a number of equality tasks.

From here on, equality tasks are denoted with number subscripts and set-based tasks with letters. Furthermore, while equality tasks in general can be multidimensional and are thus described as vectors, set-based tasks are scalar and are therefore not expressed in bold-face, e.g., $\boldsymbol{\sigma}_1$ and σ_{a}. Finally, only regulation equality tasks are considered, that is equality tasks to guide the system to a stationary value ($\dot{\boldsymbol{\sigma}}_{\mathrm{des}} \equiv 0$). Finally, it is assumed that the desired joint velocities $\dot{q}_{\mathrm{des}}$ are tracked perfectly by the system, so that $\dot{q} = \dot{q}_{\mathrm{des}}$.

Consider **Figure 1**. A set-based task σ is defined as *satisfied* when it is contained in its valid set, i.e., $\sigma \in D = [\sigma_{\min}, \sigma_{\max}]$. On the boundary of D, the task is still satisfied, but it might be necessary to actively handle the task to prevent it from becoming violated. This is elaborated on in Section 4.

4. SET-BASED SINGULARITY-ROBUST TASK-PRIORITY INVERSE KINEMATICS

This section presents the proposed method for incorporating set-based tasks in the singularity-robust multiple task-priority inverse kinematics framework. *High-priority set-based tasks* are defined as all set-based tasks with a higher priority than the highest-priority equality task, whereas *low-priority set-based tasks* have priority after at least one equality task. Section 4.1 describes how high-priority set-based tasks are handled. For simplicity, a system with a single set-based task is considered first, before the general case of j set-based tasks is presented. Low-priority set-based tasks offer some additional challenges and are further described in Section 4.2, also for 1 and j set-based tasks, respectively. Finally, in

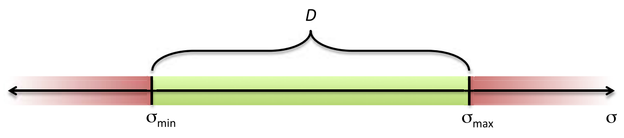

FIGURE 1 | Illustration of valid set *D*. The set-based task σ is satisfied in D and violated outside of D.

Section 4.3, the framework is presented for the completely general case of a combination of high- and low-priority set-based tasks. Note that concrete examples are given in Section 7.

When a set-based task is in the interior of D (**Figure 1**), it should not affect the behavior of the system. All the system DOFs can then be used to fulfill the equality tasks without being limited in any way by the set-based task, which is *inactive*. The proposed algorithm therefore considers only the system's equality tasks according to equation (33) as long as the resulting solution stays within this desired set or, should it be outside, does not further increase the distance to the set. If this is not the case, the set-based task is actively inserted into the task priority to be handled. A key aspect of the proposed solution is the *tangent cone* and *extended tangent cone* to a set. The tangent cone to the set D in **Figure 1** at the point $\sigma \in D$ is defined as

$$T_D(\sigma) = \begin{cases} [0,\infty) & \sigma = \sigma_{\min} \\ \mathbb{R} & \sigma \in (\sigma_{\min}, \sigma_{\max}) \\ (-\infty, 0] & \sigma = \sigma_{\max} \end{cases}. \tag{24}$$

The left-end point in **Figure 1** corresponds to $\sigma_{\min}$, in which case the tangent cone is defined as $[0,\infty)$. Similarly, the right-end point of **Figure 1** represents $\sigma_{\max}$, and in this point, the tangent cone is defined as $(-\infty, 0]$. Note that if $\dot{\sigma}(t) \in T_D(\sigma(t)) \forall t \geq t_0$, then this implies that $\sigma(t) \in D \forall t \geq t_0$. If σ is in the interior of D, the derivative is always in the tangent cone, as this is defined as $\mathbb{R}$. If $\sigma = \sigma_{\min}$, the task is at the lower border of the set. In this case, if $\dot{\sigma} \in [0, \infty)$, then σ will either stay on the border, or move into the interior of the set. Similarly, if $\sigma = \sigma_{\max}$ and $\dot{\sigma} \in (-\infty, 0]$, σ will not leave D.

We define the extended tangent cone to D at the point $\sigma \in \mathbb{R}$ as

$$T_{\mathbb{R},D}(\sigma) = \begin{cases} [0,\infty) & \sigma \leq \sigma_{\min} \\ \mathbb{R} & \sigma \in (\sigma_{\min}, \sigma_{\max}) \\ (-\infty, 0] & \sigma \geq \sigma_{\max} \end{cases}. \tag{25}$$

The definition of the extended tangent cone is very similar to the tangent cone, but it is defined for $\sigma \in \mathbb{R}$, not just $\sigma \in D$. For $\sigma \notin D$, $\dot{\sigma} \in T_{\mathbb{R},D}$ implies that σ either stays constant or moves closer to D.

Due to the fact that a set-based task can be either active or inactive, a system with j set-based tasks has 2^j possible combinations of set-based tasks being active/inactive. These combinations are referred to as "modes" of the system, and the proposed algorithm must switch between modes to fulfill the equality tasks while ensuring that the set-based tasks are not violated. The modes are sorted by increasing restrictiveness. The more set-based tasks that are active in a mode, the more restrictive it is. Hence, in the first mode, no set-based tasks are active. This corresponds to considering only the system equality tasks [equation (22)]. In the 2^jth mode, all set-based tasks are active.

Throughout this section, we consider a robotic system with n DOFs and k equality tasks of m_i DOFs each for $i \in \{1, \ldots, k\}$. Equality task i is denoted $\boldsymbol{\sigma}_i$, the task error is defined as $\tilde{\boldsymbol{\sigma}}_i \triangleq \boldsymbol{\sigma}_{i,\text{des}} - \boldsymbol{\sigma}_i$. Furthermore, the system has j set-based tasks. The first and jth set-based tasks are denoted σ_a and σ_x, respectively (where x represents the jth letter of the alphabet). We consider the state-vector $\boldsymbol{z} \in \mathbb{R}^l$, where

$$l = n + j + \sum_{i=1}^{k} m_i, \tag{26}$$

and

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\sigma}_\text{sb} \\ \tilde{\boldsymbol{\sigma}}_\text{eb} \end{bmatrix} = \begin{bmatrix} q_1 \\ \vdots \\ q_n \\ \sigma_\text{a} \\ \vdots \\ \sigma_\text{x} \\ \tilde{\boldsymbol{\sigma}}_1 \\ \vdots \\ \tilde{\boldsymbol{\sigma}}_k \end{bmatrix}. \tag{27}$$

Here, $\boldsymbol{\sigma}_\text{sb}$ and $\tilde{\boldsymbol{\sigma}}_\text{eb}$ are defined as vectors containing the set-based tasks and the corresponding valid set for $\boldsymbol{z}$ in which all set-based tasks are satisfied is defined as

$$C := \mathbb{R}^n \times C_1 \times C_2 \times \ldots \times C_j \times \mathbb{R}^{l-n-j}, \tag{28}$$

where

$$C_1 := [\sigma_{\text{a},\min}, \sigma_{\text{a},\max}] \tag{29}$$
$$\vdots$$
$$C_j := [\sigma_{\text{x},\min}, \sigma_{\text{x},\max}] \tag{30}$$

Assumption 1: When an additional task is considered, the task Jacobian is independent with respect to the Jacobian obtained by stacking all the higher-priority tasks, i.e.,

$$\rho\left({\boldsymbol{J}^\text{A}_{12..(i-1)}}^\dagger\right) + \rho\left(\boldsymbol{J}_i^\dagger\right) = \rho\left(\begin{bmatrix}{\boldsymbol{J}^\text{A}_{12..(i-1)}}^\dagger & \boldsymbol{J}_i^\dagger\end{bmatrix}\right) \tag{31}$$

for $i \in \{2, \ldots, (j+k)\}$ where, $\rho(\cdot)$ is the rank of the matrix and k is the total number of tasks. Furthermore, the task gains are chosen according to Antonelli (2009). For the specific case of $(j+k) = 3$, the task gains are chosen as $\boldsymbol{\Lambda}_1 = \lambda_1 \boldsymbol{I}_{m_1}$, $\boldsymbol{\Lambda}_2 = \lambda_2 \boldsymbol{I}_{m_2}$, and $\boldsymbol{\Lambda}_3 = \lambda_3 \boldsymbol{I}_{m_3}$ for the first, second, and third priority task, respectively, with

$$\lambda_1 > 0 \tag{32}$$

$$\lambda_2 > \max\left(0, \frac{\overline{\lambda}_{21} - \underline{\lambda}_{11}}{\underline{\lambda}_{22}} \lambda_1\right) \tag{33}$$

$$\lambda_3 > \max\left(0, \frac{\overline{\lambda}_{31} - \underline{\lambda}_{11}}{\underline{\lambda}_{33}} \lambda_1, \frac{\overline{\lambda}_{32} - \underline{\lambda}_{22}}{\underline{\lambda}_{33}} \lambda_2\right), \tag{34}$$

where $\overline{\lambda}_{ij}$ and $\underline{\lambda}_{ij}$ denote the largest and smallest singular value of the matrix $\boldsymbol{P}_{ij}$, respectively, and

$$\begin{aligned} \boldsymbol{P}_{11} &= \boldsymbol{I}_{m_1}, & \boldsymbol{P}_{22} &= \boldsymbol{J}_2 \boldsymbol{N}_1 \boldsymbol{J}_2^\dagger, \\ \boldsymbol{P}_{21} &= \boldsymbol{J}_2 \boldsymbol{J}_1^\dagger, & \boldsymbol{P}_{32} &= \boldsymbol{J}_3 \boldsymbol{N}_1 \boldsymbol{J}_2^\dagger, \\ \boldsymbol{P}_{31} &= \boldsymbol{J}_3 \boldsymbol{J}_1^\dagger, & \boldsymbol{P}_{33} &= \boldsymbol{J}_3 \boldsymbol{N}_{12} \boldsymbol{J}_3^\dagger. \end{aligned} \tag{35}$$

For practical purposes, Assumption 1 requires that the tasks are compatible. For instance, if the system is given one end-effector position tracking task and one collision avoidance task, and the desired trajectory moves through the obstacle, the tasks are clearly not compatible. In this case, Assumption 1 is not satisfied, and the system will fulfill the highest-priority task.

4.1. High-Priority Set-Based Tasks

4.1.1. One Set-Based Task, *k* Equality Tasks

For simplicity, we first consider a robotic system with a single high-priority set-based task $\sigma_a \in \mathbb{R}$. In this section, we choose σ_a as a collision avoidance task as an example, with the goal of avoiding a circular obstacle at a constant position $\boldsymbol{p}_o$ with radius $r > 0$. The task is defined as the distance between the end effector and the obstacle center. The kinematics and Jacobian are given in Antonelli (2014):

$$\sigma_a = \sqrt{(\boldsymbol{p}_o - \boldsymbol{p}_e)^T(\boldsymbol{p}_o - \boldsymbol{p}_e)} \tag{36}$$

$$\dot{\sigma}_a = \boldsymbol{J}_a\dot{\boldsymbol{q}} = -\frac{(\boldsymbol{p}_o - \boldsymbol{p}_e)^T}{||\boldsymbol{p}_o - \boldsymbol{p}_e||}\boldsymbol{J}\dot{\boldsymbol{q}}. \tag{37}$$

Here, $\boldsymbol{p}_e$ denotes the position of the end effector and $\boldsymbol{J}$ is the corresponding position Jacobian. For this specific example, we define

$$C_1 = [\varepsilon, \infty) \tag{38}$$

for an $\varepsilon > r$ and the set C as in equation (28) with $j = 1$ and C_1 as defined above. In C, the set-based task is limited to $[\varepsilon, \infty)$. Thus, the set-based task is always satisfied for $\boldsymbol{z} \in C$.

For a system with one high-priority set-based task, two modes must be considered:

(1) Ignoring the set-based task and considering only the equality tasks.
(2) Freezing the set-based task as first priority and considering the equality tasks as second priority.

Mode 1 is the "default" solution, whereas mode 2 should be activated only when it is necessary to prevent the set-based task from being violated. Using the multiple task-priority inverse kinematics framework presented in Section 3.1, mode 1 corresponds to equation (22) and results in the following system:

$$\begin{aligned}
\dot{\boldsymbol{q}} &= \boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{12}^{\mathrm{A}}\boldsymbol{J}_3^\dagger\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 \\
&\quad + \ldots + \boldsymbol{N}_{12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k
\end{aligned} \tag{39}$$

$$\Downarrow$$

$$\dot{\sigma}_a = \boldsymbol{J}_a\dot{\boldsymbol{q}} = \boldsymbol{J}_a(\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}_{12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k) \tag{40}$$

$$\begin{aligned}
\dot{\tilde{\boldsymbol{\sigma}}}_{eb} &= -\dot{\boldsymbol{\sigma}}_{eb} = -\begin{bmatrix}\dot{\boldsymbol{\sigma}}_1\\ \dot{\boldsymbol{\sigma}}_2\\ \vdots\\ \dot{\boldsymbol{\sigma}}_k\end{bmatrix} = -\begin{bmatrix}\boldsymbol{J}_1\\ \boldsymbol{J}_2\\ \vdots\\ \boldsymbol{J}_k\end{bmatrix}\dot{\boldsymbol{q}} \\
&= -\begin{bmatrix}\boldsymbol{J}_1(\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\\ \boldsymbol{J}_2(\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\\ \vdots\\ \boldsymbol{J}_k(\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\boldsymbol{\sigma}_k)\end{bmatrix} \\
&= -\begin{bmatrix}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1\\ \boldsymbol{J}_2\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_2\boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2\\ \vdots\\ \boldsymbol{J}_k\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_k\boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{J}_k\boldsymbol{N}_{12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\boldsymbol{\sigma}_k\end{bmatrix} \\
&= -\begin{bmatrix}\boldsymbol{\Lambda}_1 & \boldsymbol{0}_{m_1\times m_2} & \cdots & \boldsymbol{0}_{m_1\times m_k}\\ \boldsymbol{J}_2\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1 & \boldsymbol{J}_2\boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{0}_{m_2\times m_k}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{J}_k\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1 & \boldsymbol{J}_k\boldsymbol{N}_1\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{J}_k\boldsymbol{N}_{12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\sigma}}_1\\ \tilde{\boldsymbol{\sigma}}_2\\ \vdots\\ \tilde{\boldsymbol{\sigma}}_k\end{bmatrix} \\
&= -\boldsymbol{M}_1\tilde{\boldsymbol{\sigma}}_{eb}
\end{aligned} \tag{41}$$

$$\dot{\boldsymbol{z}} = \begin{bmatrix}\dot{\boldsymbol{q}}\\ \dot{\sigma}_a\\ \dot{\tilde{\boldsymbol{\sigma}}}_{eb}\end{bmatrix} \triangleq \begin{bmatrix}\boldsymbol{f}_{11}(\boldsymbol{z})\\ f_{12}(\boldsymbol{z})\\ \boldsymbol{f}_{13}(\boldsymbol{z})\end{bmatrix} = \boldsymbol{f}_1(\boldsymbol{z}). \tag{42}$$

The matrix $\boldsymbol{M}_1$ is positive definite by Assumption 1 (Antonelli, 2009).

In mode 1, the set-based task evolves freely according to $\dot{\sigma}_a = f_{12}(\boldsymbol{z})$, and the time evolution of $\boldsymbol{z}$ should follow the vector field $\boldsymbol{f}_1(\boldsymbol{z})$ as long as the solution $\boldsymbol{z}$ stays in C. If following $\boldsymbol{f}_1(\boldsymbol{z})$ would result in $\boldsymbol{z}$ leaving the set C, avoiding the obstacle is considered the first priority task and mode 2 is activated. This can only occur on the border of C, that is for $\sigma_a = \varepsilon$, and when $\dot{\sigma}_a = f_{12}(\boldsymbol{z}) < 0$. To avoid the obstacle, an equality task with the goal of keeping the current distance to the obstacle is added as the highest-priority task. In this case, the desired task value $\sigma_{a,des}$ is equal to the current task value $\sigma_a = \varepsilon$. Thus, the task error is $\tilde{\sigma}_a = \sigma_{a,des} - \sigma_a \equiv 0$. The system is then defined by the following equations:

$$\begin{aligned}
\dot{\boldsymbol{q}} &= \boldsymbol{J}_a^\dagger\tilde{\sigma}_a + \boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \boldsymbol{N}_{a12}^{\mathrm{A}}\boldsymbol{J}_3^\dagger\boldsymbol{\Lambda}_3\tilde{\boldsymbol{\sigma}}_3 \\
&\quad + \ldots + \boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \\
&= \boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k
\end{aligned} \tag{43}$$

$$\Downarrow$$

$$\dot{\sigma}_a = \boldsymbol{J}_a\dot{\boldsymbol{q}} = \boldsymbol{J}_a(\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \ldots + \boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k) = 0 \tag{44}$$

$$\begin{aligned}
\dot{\tilde{\boldsymbol{\sigma}}}_{eb} &= -\dot{\boldsymbol{\sigma}}_{eb} = -\begin{bmatrix}\dot{\boldsymbol{\sigma}}_1\\ \dot{\boldsymbol{\sigma}}_2\\ \vdots\\ \dot{\boldsymbol{\sigma}}_k\end{bmatrix} = -\begin{bmatrix}\boldsymbol{J}_1\\ \boldsymbol{J}_2\\ \vdots\\ \boldsymbol{J}_k\end{bmatrix}\dot{\boldsymbol{q}} \\
&= -\begin{bmatrix}\boldsymbol{J}_1(\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\\ \boldsymbol{J}_2(\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\\ \vdots\\ \boldsymbol{J}_k(\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\end{bmatrix} \\
&= -\begin{bmatrix}\boldsymbol{J}_1\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1\\ \boldsymbol{J}_2\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_2\boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2\\ \vdots\\ \boldsymbol{J}_k\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_k\boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{J}_k\boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k\end{bmatrix} \\
&= -\begin{bmatrix}\boldsymbol{J}_1\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1 & \boldsymbol{0}_{m_1\times m_2} & \cdots & \boldsymbol{0}_{m_1\times m_k}\\ \boldsymbol{J}_2\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1 & \boldsymbol{J}_2\boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2 & \cdots & 0_{m_2\times m_k}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{J}_k\boldsymbol{N}_a\boldsymbol{J}_1^\dagger\boldsymbol{\Lambda}_1 & \boldsymbol{J}_k\boldsymbol{N}_{a1}^{\mathrm{A}}\boldsymbol{J}_2^\dagger\boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{J}_k\boldsymbol{N}_{a12..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^\dagger\boldsymbol{\Lambda}_k\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\sigma}}_1\\ \tilde{\boldsymbol{\sigma}}_2\\ \vdots\\ \tilde{\boldsymbol{\sigma}}_k\end{bmatrix} \\
&= -\boldsymbol{M}_2\tilde{\boldsymbol{\sigma}}_{eb}
\end{aligned} \tag{45}$$

$$\dot{\boldsymbol{z}} = \begin{bmatrix}\dot{\boldsymbol{q}}\\ \dot{\sigma}_a\\ \dot{\tilde{\boldsymbol{\sigma}}}_{eb}\end{bmatrix} \triangleq \begin{bmatrix}\boldsymbol{f}_{21}(\boldsymbol{z})\\ 0\\ \boldsymbol{f}_{23}(\boldsymbol{z})\end{bmatrix} = \boldsymbol{f}_2(\boldsymbol{z}). \tag{46}$$

The matrix $\boldsymbol{M}_2$ can be seen as a principal submatrix of the general matrix $\boldsymbol{M}$ in equation (60) in Antonelli (2009), which is shown to be positive definite given Assumption 1. Thus, $\boldsymbol{M}_2$ is also positive definite. Furthermore, as can be seen by equation (44), the joint velocities [equation (43)] ensure that the distance to the obstacle is kept constant ($\dot{\sigma}_{\mathrm{a}} = 0$).

Let $\boldsymbol{T}_C(\boldsymbol{z})$ denote the tangent cone to C at the point $\boldsymbol{z} \in C$:

$$\boldsymbol{T}_C(\boldsymbol{z}) = \begin{cases} \mathbb{R}^l & \boldsymbol{z} \in P \\ \mathbb{R}^n \times [0, \infty) \times \mathbb{R}^{l-n-1} & \boldsymbol{z} = C \backslash P \end{cases} \tag{47}$$

Consider the continuous functions $\boldsymbol{f}_1, \boldsymbol{f}_2 : C \to \mathbb{R}^l$ as defined in equations (42) and (46). The time evolution of $\boldsymbol{z}$ follows the vector field $\boldsymbol{f}_1(\boldsymbol{z})$ as long as the solution $\boldsymbol{z}$ stays in C (mode 1). Using Lemma 5.26 (Goebel et al., 2012) on the system $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z})$, we know that such a solution exists when $\boldsymbol{f}_1(\boldsymbol{x}) \cap \boldsymbol{T}_C(\boldsymbol{x}) \neq \emptyset$ for all $\boldsymbol{x}$ near $\boldsymbol{z}$ (restricting $\boldsymbol{x}$ to C). Hence, we define the set

$$S := \{\boldsymbol{z} \in C : \exists \text{ a neighborhood U of } \boldsymbol{z} : \boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in C \cap U\}. \tag{48}$$

The discontinuous function $\boldsymbol{f} : C \to \mathbb{R}^l$

$$\boldsymbol{f}(\boldsymbol{z}) := \begin{cases} \boldsymbol{f}_1(\boldsymbol{z}) & \boldsymbol{z} \in S \\ \boldsymbol{f}_2(\boldsymbol{z}) & \boldsymbol{z} \in C \backslash S \end{cases} \tag{49}$$

then describes our system. The differential equation $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ then corresponds to following $\boldsymbol{f}_1$ (mode 1) as long as $\boldsymbol{z}$ stays in C and following $\boldsymbol{f}_2$ (mode 2) otherwise. In mode 2, σ_{a} is frozen, so the $(n+1)$th element in $\boldsymbol{f}_2(\boldsymbol{z}) \equiv 0$. Consequently, $\boldsymbol{f}_2(\boldsymbol{z}) \in \boldsymbol{T}_C(\boldsymbol{z}) \, \forall \boldsymbol{z} \in C$. This implies that C is strongly forward invariant for $\dot{\boldsymbol{z}} = \boldsymbol{f}_2(\boldsymbol{z})$, i.e., that $\boldsymbol{z}(t_0) \in C \Rightarrow \boldsymbol{z}(t) \in C \, \forall t \geq t_0$.

In other words, the set S contains the points $\boldsymbol{z}$ in C such that $\boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x})$ for $\boldsymbol{x}$ in C that are near $\boldsymbol{z}$. At the border of C, $\sigma_{\mathrm{a}} = \varepsilon$ and the distance between the end effector and the obstacle center is at the minimum allowed value. Therefore, the $(n+1)$th element of $\boldsymbol{f}_1(\boldsymbol{z})$, corresponding to $\dot{\sigma}_{\mathrm{a}}$, must be zero or positive for $\boldsymbol{z}$ to stay in S. If it is not, mode 2 is activated, which freezes the distance to the obstacle at the border of C. This remains the solution until following $\boldsymbol{f}_1(\boldsymbol{z})$ will result in $\dot{\sigma}_{\mathrm{a}} \geq 0$, i.e., that the distance between the end effector and obstacle will remain constant or increase.

4.1.2. j Set-Based Tasks, k Equality Tasks

We now consider the general case of j high-priority set-based tasks and consider the state vector $\boldsymbol{z} \in \mathbb{R}^l$ and the corresponding valid set as defined in equations (27) and (28), respectively. Note that in the case that a set-based task σ is a joint limit for joint i, $i \in \{1, \ldots, n\}$, then $\sigma = q_i$ and should therefore not be included in the vector $\boldsymbol{\sigma}_{\mathrm{sb}}$, as it is already included in the state vector through $\boldsymbol{q}$. With j set-based tasks, it is necessary to consider the 2^j solutions resulting from all combinations of activating and deactivating every set-based task. These are presented in **Table 1**.

All modes have certain commonalities:

(1) All active set-based tasks are frozen.
(2) Inactive set-based tasks do not affect the behavior of the system.
(3) All matrices $\boldsymbol{M}_i$ for $i \in \{1, \ldots, 2^j\}$ are either equal to or a principal submatrix of the general matrix $\boldsymbol{M}$ in equation (60) in Antonelli (2009). Consequently, given Assumption 1, the matrices $\boldsymbol{M}_i$ are positive definite.

Consider the corresponding tangent cone to the set C in equation (28).

$$\boldsymbol{T}_C(\boldsymbol{z}) = \mathbb{R}^n \times T_{C_1}(\boldsymbol{z}) \times T_{C_2}(\boldsymbol{z}) \times \ldots \times T_{C_j}(\boldsymbol{z}) \times \mathbb{R}^{l-n-j}, \tag{50}$$

where $T_{C_i}(\boldsymbol{z})$ for $i \in \{1, \ldots, j\}$ is defined as in equation (24). Defining the sets

$$\begin{aligned} S_1 &:= \{\boldsymbol{z} \in C : \exists \text{ a neighborhood U of } \boldsymbol{z} : \\ &\quad \boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\} && (51) \\ S_2 &:= \{\boldsymbol{z} \in C \backslash S_1 : \exists \text{ a nbhd U of } \boldsymbol{z} : \\ &\quad \boldsymbol{f}_2(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\} && (52) \\ S_3 &:= \{\boldsymbol{z} \in C \backslash (S_1 \cup S_2) : \exists \text{ a nbhd U of } \boldsymbol{z} : \\ &\quad \boldsymbol{f}_3(\boldsymbol{x}) \in \boldsymbol{T}_C(\boldsymbol{x}) \forall \boldsymbol{x} \in C \cap U\} && (53) \\ &\vdots \\ S_{2^j} &:= C \backslash \left(S_1 \cup S_2 \cup S_3 \cup \ldots \cup S_{2^j - 1}\right), && (54) \end{aligned}$$

the discontinuous equation $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ with $\boldsymbol{f} : C \to \mathbb{R}^l$ defined as

$$\boldsymbol{f}(\boldsymbol{z}) := \begin{cases} \boldsymbol{f}_1(\boldsymbol{z}) & \boldsymbol{z} \in S_1 \\ \boldsymbol{f}_2(\boldsymbol{z}) & \boldsymbol{z} \in S_2 \\ \boldsymbol{f}_3(\boldsymbol{z}) & \boldsymbol{z} \in S_3 \\ \vdots & \vdots \\ \boldsymbol{f}_{2^j}(\boldsymbol{z}) & \boldsymbol{z} \in S_{2^j} \end{cases} \tag{55}$$

then defines our system.

TABLE 1 | System equations for the resulting 2^j modes for *j* high-priority set-based tasks.

Mode	Description	Equations
1	No set-based tasks active	$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}^{\mathrm{A}}_{12..(k-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ $\Rightarrow \dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_1 \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}$ $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z})$
2	σ_{a} active	$\dot{\boldsymbol{q}} = \boldsymbol{N}_{\mathrm{a}} \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}^{\mathrm{A}}_{\mathrm{a}12..(\mathrm{k}-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ $\Rightarrow \dot{\sigma}_{\mathrm{a}} = 0$ $\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_2 \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}$ $\boldsymbol{z} = \boldsymbol{f}_2(\boldsymbol{z})$
3	σ_{b} active	$\dot{\boldsymbol{q}} = \boldsymbol{N}_{\mathrm{b}} \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}^{\mathrm{A}}_{\mathrm{b}12..(\mathrm{k}-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ $\Rightarrow \dot{\sigma}_{\mathrm{b}} = 0$ $\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_3 \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}$ $\dot{\boldsymbol{z}} = \boldsymbol{f}_3(\boldsymbol{z})$
$\ddots$	$\ddots$	$\ddots$
2^j	All set-based tasks active	$\dot{\boldsymbol{q}} = \boldsymbol{N}^{\mathrm{A}}_{\mathrm{ab..x}} \boldsymbol{J}_1^{\dagger} \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + \ldots + \boldsymbol{N}^{\mathrm{A}}_{\mathrm{ab..x}12..(\mathrm{k}-1)} \boldsymbol{J}_k^{\dagger} \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$ $\Rightarrow \dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{sb}} = 0$ $\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_{2^j} \tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}$ $\dot{\boldsymbol{z}} = \boldsymbol{f}_{2^j}(\boldsymbol{z})$

4.2. Low-Priority Set-Based Tasks

4.2.1. One Set-Based Task, k Equality Tasks

For simplicity, we first consider a robotic system with a single low-priority set-based task $\sigma_{\mathrm{a}} \in \mathbb{R}$ with priority between equality tasks $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ and a valid set $\sigma_{\mathrm{a}} \in [\sigma_{\mathrm{a,min}}, \sigma_{\mathrm{a,max}}]$.

Consider the state-vector $\boldsymbol{z} \in \mathbb{R}^l$ and the closed set C, where l, $\boldsymbol{z}$, and C are defined in equations (26)–(28), respectively.

Similar to Section 4.1.1, one set-based task leads to two modes, and in the first mode only the equality tasks are considered. Hence, mode 1 corresponds exactly to equations (39)–(42) and reduces the equality task error dynamics to $\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} = -\boldsymbol{M}_1\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}}$. In mode 2, the set-based task is actively handled, and as for the higher-priority set-based tasks in Section 4.1.1, the goal is to freeze it at its current value. Hence, $\tilde{\sigma}_{\mathrm{a}} \equiv 0$ and the joint velocities for mode 2 are given as

$$\begin{aligned}\dot{\boldsymbol{q}} &= \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_1\boldsymbol{J}_{\mathrm{a}}^{\dagger}\Lambda_{\mathrm{a}}\tilde{\sigma}_{\mathrm{a}} + \boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 \\ &\quad + \ldots + \boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \\ &= \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k. \quad (56)\end{aligned}$$

In Section 4.1.1, it was shown that mode 2 implies $\dot{\sigma}_{\mathrm{a}} = 0$, so if this mode is activated on the border of C ($\sigma_{\mathrm{a}} = \sigma_{\mathrm{a,min}}$ or $\sigma_{\mathrm{a}} = \sigma_{\mathrm{a,max}}$), σ_{a} will indeed be frozen and thereby remain in C. However, when σ_{a} is a low-priority task, the same guarantee cannot be made. Consider the evolution of σ_{a} in case of equation (56):

$$\begin{aligned}\dot{\sigma}_{\mathrm{a}} &= \boldsymbol{J}_{\mathrm{a}}\dot{\boldsymbol{q}} = \boldsymbol{J}_{\mathrm{a}}(\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k) \\ &= \boldsymbol{J}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \quad (57)\end{aligned}$$

Equation (57) is not exactly equal to zero. Rather, the evolution of σ_{a} is influenced by the higher-priority equality task. Thus, lower-priority set-based tasks cannot be guaranteed to be satisfied, as they can not be guaranteed to be frozen at any given time. Even so, they should still be actively handled by attempting to freeze them, as this might result in these tasks deviating less from their valid set than if they are ignored completely. Thus, we define the system mode 2 as equation (56):

$$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k \quad (58)$$

$$\Downarrow$$

$$\dot{\sigma}_{\mathrm{a}} = \boldsymbol{J}_{\mathrm{a}}\dot{\boldsymbol{q}} = \boldsymbol{J}_{\mathrm{a}}\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \quad (59)$$

$$\begin{aligned}\dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}} &= -\dot{\boldsymbol{\sigma}}_{\mathrm{eb}} = -\begin{bmatrix}\dot{\boldsymbol{\sigma}}_1\\ \dot{\boldsymbol{\sigma}}_2\\ \vdots\\ \dot{\boldsymbol{\sigma}}_k\end{bmatrix} = -\begin{bmatrix}\boldsymbol{J}_1\\ \boldsymbol{J}_2\\ \vdots\\ \boldsymbol{J}_k\end{bmatrix}\dot{\boldsymbol{q}} \\ &= -\begin{bmatrix}\boldsymbol{J}_1(\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\boldsymbol{\sigma}_k)\\ \boldsymbol{J}_2(\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\\ \vdots\\ \boldsymbol{J}_k(\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2 + \ldots + \boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k)\end{bmatrix} \\ &= -\begin{bmatrix}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1\\ \boldsymbol{J}_2\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + \boldsymbol{J}_2\boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2\tilde{\boldsymbol{\sigma}}_2\\ \vdots\\ \boldsymbol{J}_k\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + + \ldots + \boldsymbol{J}_k\boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\tilde{\boldsymbol{\sigma}}_k\end{bmatrix} \\ &= -\begin{bmatrix}\boldsymbol{\Lambda}_1 & \boldsymbol{0}_{m_1\times m_2} & \cdots & \boldsymbol{0}_{m_1\times m_k}\\ \boldsymbol{J}_2\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1 & \boldsymbol{J}_2\boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{0}_{m_2\times m_k}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{J}_k\boldsymbol{J}_1^{\dagger}\boldsymbol{\Lambda}_1 & \boldsymbol{J}_k\boldsymbol{N}_{1\mathrm{a}}^{\mathrm{A}}\boldsymbol{J}_2^{\dagger}\boldsymbol{\Lambda}_2 & \cdots & \boldsymbol{J}_k\boldsymbol{N}_{1\mathrm{a}2..(k-1)}^{\mathrm{A}}\boldsymbol{J}_k^{\dagger}\boldsymbol{\Lambda}_k\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{\sigma}}_1\\ \tilde{\boldsymbol{\sigma}}_2\\ \vdots\\ \tilde{\boldsymbol{\sigma}}_k\end{bmatrix} \quad (60) \\ &= -\boldsymbol{M}_2\tilde{\boldsymbol{\sigma}}_{\mathrm{eb}} \quad (61)\end{aligned}$$

$$\dot{\boldsymbol{z}} = \begin{bmatrix}\dot{\boldsymbol{q}}\\ \dot{\sigma}_{\mathrm{a}}\\ \dot{\tilde{\boldsymbol{\sigma}}}_{\mathrm{eb}}\end{bmatrix} \triangleq \begin{bmatrix}\boldsymbol{f}_{21}(\boldsymbol{z})\\ \boldsymbol{f}_{22}(\boldsymbol{z})\\ \boldsymbol{f}_{23}(\boldsymbol{z})\end{bmatrix} = \boldsymbol{f}_2(\boldsymbol{z}). \quad (62)$$

Similar to Section 4.1.1, the matrix $\boldsymbol{M}_2$ is positive definite given Assumption 1.

Consider the continuous functions $\boldsymbol{f}_1, \boldsymbol{f}_2 : C \to \mathbb{R}^l$ as defined in equations (42) and (62). The time evolution of $\boldsymbol{z}$ should follow the vector field $\boldsymbol{f}_1(\boldsymbol{z})$ as long as the solution $\boldsymbol{z}$ stays in C (mode 1). However, since it cannot be guaranteed that $\boldsymbol{z}$ stays in C, mode 1 should also be active if $\boldsymbol{z} \notin C$ and following $\boldsymbol{f}_1(\boldsymbol{z})$ will result in $\boldsymbol{z}$ keeping its current distance to or moving closer to C. This is mathematically expressed for the system $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z})$ as $\boldsymbol{f}_1(\boldsymbol{x}) \cap \boldsymbol{T}_{\mathbb{R},C}(\boldsymbol{x}) \neq \emptyset$ for all $\boldsymbol{x}$ near $\boldsymbol{z}$, where

$$\boldsymbol{T}_{\mathbb{R},C}(\boldsymbol{z}) = \mathbb{R}^n \times T_{\mathbb{R},C_1}(\boldsymbol{z}) \times \mathbb{R}^{l-n-1}. \quad (63)$$

and $T_{\mathbb{R},C_1}(\boldsymbol{z})$ is the extended tangent cone to C_1 in equation (29) at the point σ_{a} as defined in equation (25). Hence, we define

$$\begin{aligned}S := \{\boldsymbol{z} \in \mathbb{R}^l : \exists \text{ a neighborhood U of } \boldsymbol{z} : \\ \boldsymbol{f}_1(\boldsymbol{x}) \in \boldsymbol{T}_{\mathbb{R},C}(\boldsymbol{x})\ \forall \boldsymbol{x} \in U\}. \quad (64)\end{aligned}$$

The discontinuous function $\boldsymbol{f} : C \to \mathbb{R}^l$

$$\boldsymbol{f}(\boldsymbol{z}) := \begin{cases}\boldsymbol{f}_1(\boldsymbol{z}) & \boldsymbol{z} \in S\\ \boldsymbol{f}_2(\boldsymbol{z}) & \boldsymbol{z} \in \mathbb{R}^l \backslash S\end{cases} \quad (65)$$

then describes our system. The differential equation $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ then corresponds to following $\boldsymbol{f}_1(\boldsymbol{z})$ (mode 1) if one of three conditions are satisfied:

(1) $\sigma_{\mathrm{a}} \in (\sigma_{\mathrm{a,min}}, \sigma_{\mathrm{a,max}})$
(2) $\sigma_{\mathrm{a}} \geq \sigma_{\mathrm{a,max}}$ and $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z}) \Rightarrow \dot{\sigma}_{\mathrm{a}} \leq 0$
(3) $\sigma_{\mathrm{a}} \leq \sigma_{\mathrm{a,min}}$ and $\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z}) \Rightarrow \dot{\sigma}_{\mathrm{a}} \geq 0$.

If none of these conditions hold, mode 2 is activated and $\dot{\boldsymbol{z}} = \boldsymbol{f}_2(\boldsymbol{z})$. In mode 2, the set-based task is actively handled by attempting to freeze it, but since the $(n+1)$th element in $\boldsymbol{f}_2(\boldsymbol{z})$ is not identically equal to 0, this cannot be guaranteed. However, as shown in equation (59), as soon as the higher-priority equality task $\boldsymbol{\sigma}_1$ converges to its desired value, σ_{a} is indeed frozen in mode 2 and is not affected by the lower-priority equality tasks.

4.2.2. j Set-Based Task, k Equality Tasks

In this section, we consider a system with j low-priority set-based tasks. Consider the state-vector $\boldsymbol{z} \in \mathbb{R}^l$, where l and $\boldsymbol{z}$ are defined in equations (95) and (27), respectively. We assume that the set-based tasks are labeled so that σ_{b} always has lower priority than σ_{a}, etc. Furthermore, assume that σ_{a} has priority after equality task

$\boldsymbol{\sigma}_{p_\mathrm{a}}$ for some $p_\mathrm{a} = \{1, \ldots, k\}$, σ_b has priority after equality task $\boldsymbol{\sigma}_{p_\mathrm{b}}$ for some $p_\mathrm{b} = \{p_a, \ldots, k\}$, and so forth. The resulting 2^j modes of the system are presented in **Table 2**.

For example, for a system with $j = 3$ set-based tasks and $k = 6$ equality tasks, $p_\mathrm{a} = 3$ and $p_\mathrm{b} = p_\mathrm{c} = 5$, the $2^j = 8$th mode would be

$$\begin{aligned} \dot{q} &= J_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + N_1 J_2^\dagger \boldsymbol{\Lambda}_2 \tilde{\boldsymbol{\sigma}}_2 + N_{12}^\mathrm{A} J_3^\dagger \boldsymbol{\Lambda}_3 \tilde{\boldsymbol{\sigma}}_3 \\ &\quad + N_{123\mathrm{a}}^\mathrm{A} J_4^\dagger \boldsymbol{\Lambda}_4 \tilde{\boldsymbol{\sigma}}_4 + N_{123\mathrm{a}4}^\mathrm{A} J_5^\dagger \boldsymbol{\Lambda}_5 \tilde{\boldsymbol{\sigma}}_5 + N_{123\mathrm{a}45\mathrm{bc}}^\mathrm{A} J_6^\dagger \boldsymbol{\Lambda}_6 \tilde{\boldsymbol{\sigma}}_6 \end{aligned} \tag{66}$$

All matrices $\boldsymbol{M}_i$ in **Table 2** for $i \in \{1, \ldots 2^j\}$ are either equal to or a principal submatrix of the general matrix $\boldsymbol{M}$ in equation (60) in Antonelli (2009). Consequently, given Assumption 1, the matrices $\boldsymbol{M}_i$ are positive definite.

Consider the sets C_1-C_j defined by equations (29) and (30) with corresponding extended tangent cones as defined in equation (25). If the ith set-based task is inactive in a given mode $\boldsymbol{f}$, the corresponding $(n+i)$th element of vector field $\boldsymbol{f}(\boldsymbol{x})$ must be in $T_{\mathbb{R},C_i}(\boldsymbol{x})$ for all $\boldsymbol{x}$ near $\boldsymbol{z}$. If the same task is active, there is no requirement to the same element of the vector field. In mode 1, all the set-based tasks are inactive, thus all the $(n+1)$ to $(n+j)$ elements of $\boldsymbol{f}_1(\boldsymbol{z})$ must be in their respective extended tangent cones for this mode to be chosen. In mode 2, only σ_a is active, so in this case all the $(n+2)$ to $(n+j)$ elements of $\boldsymbol{f}_2(\boldsymbol{z})$ must be in their respective extended tangent cones if mode 2 should be the active mode. Thus, we define

$$\begin{aligned} S_1 := \Big\{ &\boldsymbol{z} \in \mathbb{R}^l : \exists \text{ a neighborhood U of } \boldsymbol{z} : \\ &\boldsymbol{f}_1(\boldsymbol{x}) \in \Big\{ \mathbb{R}^n \times T_{\mathbb{R},C_1}(\boldsymbol{x}) \\ &\times \ldots \times T_{\mathbb{R},C_j}(\boldsymbol{x}) \times \mathbb{R}^{l-n-j} \Big\} \forall \boldsymbol{x} \in U \Big\} \end{aligned} \tag{67}$$

$$\begin{aligned} S_2 := \Big\{ &\boldsymbol{z} \in \mathbb{R}^l \backslash S_1 : \exists \text{ a nbhd U of } \boldsymbol{z} : \\ &\boldsymbol{f}_2(\boldsymbol{x}) \in \Big\{ \mathbb{R}^n \times \mathbb{R} \times T_{\mathbb{R},C_2}(\boldsymbol{x}) \\ &\times \ldots \times T_{\mathbb{R},C_j}(\boldsymbol{x}) \times \mathbb{R}^{l-n-j} \Big\} \forall \boldsymbol{x} \in U \Big\} \end{aligned} \tag{68}$$

$$\begin{aligned} S_3 := \Big\{ &\boldsymbol{z} \in \mathbb{R}^l \backslash (S_1 \cup S_2) : \exists \text{ a nbhd U of } \boldsymbol{z} : \\ &\boldsymbol{f}_3(\boldsymbol{x}) \in \Big\{ \mathbb{R}^n \times T_{\mathbb{R},C_1}(\boldsymbol{x}) \times \mathbb{R} \times T_{\mathbb{R},C_3} \\ &\times \ldots \times T_{\mathbb{R},C_j}(\boldsymbol{x}) \times \mathbb{R}^{l-n-j} \Big\} \forall \boldsymbol{x} \in U \Big\} \end{aligned} \tag{69}$$

$$\vdots$$

$$S_{2^j} := C \backslash \left(S_1 \cup S_2 \cup S_3 \cup \ldots \cup S_{2^j-1} \right). \tag{70}$$

The discontinuous equation $\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{z})$ with $\boldsymbol{f} : C \to \mathbb{R}^l$ defined as

$$f(z) := \begin{cases} \boldsymbol{f}_1(z) & z \in S_1 \\ \boldsymbol{f}_2(z) & z \in S_2 \\ \boldsymbol{f}_3(z) & z \in S_3 \\ \vdots & \vdots \\ \boldsymbol{f}_{2^j}(z) & z \in S_{2^j} \end{cases} \tag{71}$$

then defines our system.

TABLE 2 | System equations for the resulting 2^j modes for j low-priority set-based tasks.

Mode	Description
1	No set-based tasks active
	$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + .. + \boldsymbol{N}_{12..(k-1)}^\mathrm{A} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$
	$\Rightarrow \boldsymbol{\sigma}_\mathrm{eb} = -\boldsymbol{M}_1 \tilde{\boldsymbol{\sigma}}_\mathrm{eb}$
	$\dot{\boldsymbol{z}} = \boldsymbol{f}_1(\boldsymbol{z})$
2	$\boldsymbol{\sigma}_\mathrm{a}$ active
	$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + .. + \boldsymbol{N}_{12..(p_\mathrm{a}-1)}^\mathrm{A} \boldsymbol{J}_{p_\mathrm{a}}^\dagger \boldsymbol{\Lambda}_{p_\mathrm{a}} \boldsymbol{\sigma}_{p_\mathrm{a}} + \boldsymbol{N}_{12..p_\mathrm{a}\mathrm{a}}^\mathrm{A} \boldsymbol{J}_{p_\mathrm{a}+1}^\dagger \boldsymbol{\Lambda}_{p_\mathrm{a}+1} \tilde{\boldsymbol{\sigma}}_{p_\mathrm{a}+1}$
	$+ .. + \boldsymbol{N}_{12..p_\mathrm{a}\mathrm{a}(p_\mathrm{a}+1)..(k-1)}^\mathrm{A} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \boldsymbol{\sigma}_k$
	$\Rightarrow \dot{\tilde{\boldsymbol{\sigma}}}_\mathrm{eb} = -\boldsymbol{M}_2 \tilde{\boldsymbol{\sigma}}_\mathrm{eb}$
	$\dot{\boldsymbol{z}} = \boldsymbol{f}_2(\boldsymbol{z})$
3	$\boldsymbol{\sigma}_\mathrm{b}$ active
	$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + .. + \boldsymbol{N}_{12..(p_\mathrm{b}-1)}^\mathrm{A} \boldsymbol{J}_{p_\mathrm{b}}^\dagger \boldsymbol{\Lambda}_{p_\mathrm{b}} \boldsymbol{\sigma}_{p_\mathrm{b}} + \boldsymbol{N}_{12..p_\mathrm{b}\mathrm{b}}^\mathrm{A} \boldsymbol{J}_{p_\mathrm{b}+1}^\dagger \boldsymbol{\Lambda}_{p_\mathrm{b}+1} \tilde{\boldsymbol{\sigma}}_{p_\mathrm{b}+1}$
	$+ .. + \boldsymbol{N}_{12..p_\mathrm{b}\mathrm{b}(p_\mathrm{b}+1)..(k-1)}^\mathrm{A} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \tilde{a}_k$
	$\Rightarrow \dot{\tilde{\boldsymbol{\sigma}}}_\mathrm{eb} = -\boldsymbol{M}_3 \tilde{\boldsymbol{\sigma}}_\mathrm{eb}$
	$\dot{\boldsymbol{z}} = \boldsymbol{f}_3(\boldsymbol{z})$
$\ddots$	$\ddots$
2^j	All set-based tasks active
	$\dot{\boldsymbol{q}} = \boldsymbol{J}_1^\dagger \boldsymbol{\Lambda}_1 \tilde{\boldsymbol{\sigma}}_1 + .. + \boldsymbol{N}_{12..(p_\mathrm{a}-1)}^\mathrm{A} \boldsymbol{J}_{p_\mathrm{a}}^\dagger \boldsymbol{\Lambda}_{p_\mathrm{a}} \boldsymbol{\sigma}_{p_\mathrm{a}} + \boldsymbol{N}_{12..p_\mathrm{a}\mathrm{a}}^\mathrm{A} \boldsymbol{J}_{p_\mathrm{a}+1}^\dagger \boldsymbol{\Lambda}_{p_\mathrm{a}+1} \tilde{\boldsymbol{\sigma}}_{p_\mathrm{a}+1}$
	$+ .. + \boldsymbol{N}_{12..p_\mathrm{a}\mathrm{a}(p_\mathrm{a}+1)..p_\mathrm{b}\mathrm{b}}^\mathrm{A} \boldsymbol{J}_{p_\mathrm{b}+1}^\dagger \boldsymbol{\Lambda}_{p_\mathrm{b}+1} \tilde{\boldsymbol{\sigma}}_{p_\mathrm{b}+1}$
	$+ .. + \boldsymbol{N}_{12..p_\mathrm{a}\mathrm{a}(p_\mathrm{a}+1)..p_\mathrm{b}\mathrm{b}(p_\mathrm{b}+1)....p_\mathrm{x}\mathrm{x}(p_\mathrm{x}+1)..(k-1)}^\mathrm{A} \boldsymbol{J}_k^\dagger \boldsymbol{\Lambda}_k \tilde{\boldsymbol{\sigma}}_k$
	$\Rightarrow \tilde{\boldsymbol{\sigma}}_\mathrm{eb} = -\boldsymbol{M}_{2^j} \tilde{\boldsymbol{\sigma}}_\mathrm{eb}$
	$\dot{\boldsymbol{z}} = \boldsymbol{f}_{2^j}(\boldsymbol{z})$

4.3. Combination of High- and Low-Priority Set-Based Tasks

Consider an n DOF robotic system with k equality tasks of m_i DOFs for $i \in \{1, \ldots, k\}$ and j set-based tasks $\in \mathbb{R}$, where $j_x \leq j$ of them are high priority and $j - j_x$ are low-priority at any given priority level. Denote that the j_x th, $(j_x + 1)$th, and jth set-based task as σ_x, σ_y, and σ_z, respectively, where σ_x is the last high-priority set-based task, σ_y is the first low-priority set-based task, and σ_z is the lowest priority set-based task [x, y, and z represent the j_xth, $(j_x + 1)$th, and jth letter of the alphabet, respectively]. Consider the state-vector $\boldsymbol{z} \in \mathbb{R}^l$, where

$$l = n + j + \sum_{i=1}^{k} m_i, \tag{72}$$

and

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{q} \\ \boldsymbol{\sigma}_\mathrm{sb} \\ \tilde{\boldsymbol{\sigma}}_\mathrm{eb} \end{bmatrix} = \left[q_1, .., q_n, \sigma_\mathrm{a}, .., \sigma_\mathrm{x}, \sigma_\mathrm{y}, .., \sigma_\mathrm{z}, \tilde{\boldsymbol{\sigma}}_1^T, .., \tilde{\boldsymbol{\sigma}}_k^T \right]^T. \tag{73}$$

It can be shown that the 2^j resulting modes from all possible combinations of active and inactive set-based tasks will reduce the error dynamics of the equality tasks to the form

$$\dot{\tilde{\boldsymbol{\sigma}}}_\mathrm{eb} = -\boldsymbol{M}_i \tilde{\boldsymbol{\sigma}}_\mathrm{eb} \tag{74}$$

for $i = \{1, \ldots, 2^j\}$. The analysis is similar to the above sections and all matrices $\boldsymbol{M}_i$ are either equal to or a principal submatrix of the general matrix $\boldsymbol{M}$ in equation (60) in Antonelli (2009). Therefore, by Assumption 1, the matrices $\boldsymbol{M}_i$ are positive definite.

Furthermore, as described in detail in Section 4.1, all the active high-priority set-based tasks are frozen in all modes.

Consider the sets

$$C_1 := [\sigma_{\mathrm{a,min}}, \sigma_{\mathrm{a,max}}], \tag{75}$$
$$\vdots$$
$$C_{j_x} := [\sigma_{\mathrm{x,min}}, \sigma_{\mathrm{x,max}}], \tag{76}$$
$$C_{j_x+1} := [\sigma_{\mathrm{y,min}}, \sigma_{\mathrm{y,max}}], \tag{77}$$
$$\vdots$$
$$C_j := [\sigma_{\mathrm{z,min}}, \sigma_{\mathrm{z,max}}], \tag{78}$$
$$C := \mathbb{R}^n \times C_1 \times \ldots \times C_{j_x} \times \mathbb{R}^{j-j_x} \times \mathbb{R}^{l-n-j} \tag{79}$$

and

$$T_{C_1}(z) = \begin{cases} [0,\infty) & \sigma_{\mathrm{a}} = \sigma_{\mathrm{a,min}} \\ \mathbb{R} & \sigma_{\mathrm{a}} \in (\sigma_{\mathrm{a,min}}, \sigma_{\mathrm{a,max}}) \\ (-\infty, 0] & \sigma_{\mathrm{a}} = \sigma_{\mathrm{a,max}} \end{cases}, \tag{80}$$
$$\vdots$$
$$T_{C_{j_x}}(z) = \begin{cases} [0,\infty) & \sigma_{\mathrm{x}} = \sigma_{\mathrm{x,min}} \\ \mathbb{R} & \sigma_{\mathrm{x}} \in (\sigma_{\mathrm{x,min}}, \sigma_{\mathrm{x,max}}) \\ (-\infty, 0] & \sigma_{\mathrm{x}} = \sigma_{\mathrm{j,max}} \end{cases} \tag{81}$$
$$T_{\mathbb{R},C_{j_x+1}}(z) = \begin{cases} [0,\infty) & \sigma_{\mathrm{y}} \le \sigma_{\mathrm{y,min}} \\ \mathbb{R} & \sigma_{\mathrm{y}} \in (\sigma_{\mathrm{y,min}}, \sigma_{\mathrm{y,max}}) \\ (-\infty, 0] & \sigma_{\mathrm{y}} \ge \sigma_{\mathrm{y,max}} \end{cases} \tag{82}$$
$$\vdots$$
$$T_{\mathbb{R},C_j}(z) = \begin{cases} [0,\infty) & \sigma_{\mathrm{z}} \le \sigma_{\mathrm{z,min}} \\ \mathbb{R} & \sigma_{\mathrm{z}} \in (\sigma_{\mathrm{z,min}}, \sigma_{\mathrm{z,max}}) \\ (-\infty, 0] & \sigma_{\mathrm{z}} \ge \sigma_{\mathrm{z,max}} \end{cases}. \tag{83}$$
$$\boldsymbol{T}_{C_{\mathrm{h}}}(z) = T_{C_1}(z) \times T_{C_2}(z) \times \ldots \times T_{C_{j_x}}(z) \tag{84}$$
$$\boldsymbol{T}_C(z) = \mathbb{R}^n \times \boldsymbol{T}_{C_{\mathrm{h}}}(z) \times \mathbb{R}^{j-j_x} \times \mathbb{R}^{l-n-j}. \tag{85}$$

Note that $z \in C$ only implies that all the high-priority set-based tasks are within their respective desired sets. Furthermore, the tangent cones are considered for the high-priority set-based tasks, and the extended tangent cones for the low-priority ones.

$$S_1 := \{z \in C : \exists \text{ a neighborhood U of } z : \tag{86}$$
$$f_1(x) \in \left\{ \underbrace{\mathbb{R}^n \times \boldsymbol{T}_{C_{\mathrm{h}}}(x)}_{\triangleq T_{C_{\mathrm{qh}}}(x)} \times T_{\mathbb{R},C_{j_x+1}}(x) \right.$$
$$\left. \times \ldots \times \underbrace{\boldsymbol{T}_{\mathbb{R},C_j}(x) \times \mathbb{R}^{l-n-j}}_{\triangleq T_{\mathbb{R},C_{j\sigma}}(x)} \right\} \forall x \in C \cap U \Big\}$$
$$S_2 := \{z \in C : \backslash S_1 : \exists \text{ a nbhd U of } z : \tag{87}$$
$$f_2(x) \in \left\{ \boldsymbol{T}_{C_{\mathrm{qh}}}(x) \times T_{\mathbb{R},C_{j_x+1}}(x) \times \ldots \times \boldsymbol{T}_{\mathbb{R},C_{j\sigma}}(x) \right\} \forall x \in C \cap U \Big\}$$
$$\vdots$$
$$S_{j_x+1} := \left\{ z \in C \backslash (S_1 \cup \ldots \cup S_{j_x}) : \exists \text{ a nbhd U of } z : \right. \tag{88}$$
$$f_{j_x+1}(x) \in \left\{ \boldsymbol{T}_{C_{\mathrm{qh}}}(x) \times T_{\mathbb{R},C_{j_x+1}}(x) \right.$$
$$\left. \times \ldots \times \boldsymbol{T}_{\mathbb{R},C_{j\sigma}}(x) \right\} \forall x \in C \cap U \Big\}$$
$$S_{j_x+2} := \left\{ z \in C \backslash (S_1 \cup \ldots \cup S_{j_x+1}) : \exists \text{ a nbhd U of } z : \right. \tag{89}$$
$$f_{j_x+2}(x) \in \left\{ \boldsymbol{T}_{C_{\mathrm{qh}}}(x) \times \mathbb{R} \times T_{\mathbb{R},C_{j_x+2}}(x) \right.$$
$$\left. \times \ldots \times \boldsymbol{T}_{\mathbb{R},C_{j\sigma}}(x) \right\} \forall x \in C \cap U \Big\}$$
$$S_{j_x+3} := \left\{ z \in C \backslash (S_1 \cup \ldots \cup S_{j_x+2}) : \exists \text{ a nbhd U of } z : \right. \tag{90}$$
$$f_{j_x+3}(x) \in \left\{ \boldsymbol{T}_{C_{\mathrm{qh}}}(x) \times T_{\mathbb{R},C_{j_x+1}}(x) \times \mathbb{R} \times T_{\mathbb{R},C_{j_x+3}}(x) \right.$$
$$\times \ldots \times \boldsymbol{T}_{\mathbb{R},C_{j\sigma}}(x) \; \forall x \in C \cap U \Big\}$$
$$\vdots$$
$$S_{2^j} := C \backslash \left(S_1 \cup S_2 \cup S_3 \cup \ldots \cup S_{2^j-1} \right). \tag{91}$$

The discontinuous equation $\dot{z} = f(z)$ with $f : C \to \mathbb{R}^l$ defined as

$$f(z) := \begin{cases} f_1(z) & z \in S_1 \\ f_2(z) & z \in S_2 \\ f_3(z) & z \in S_3 \\ \vdots & \vdots \\ f_{2^j}(z) & z \in S_{2^j} \end{cases} \tag{92}$$

then defines our system. Similar to the analysis of the previous sections, in the sets S_i, $i \in \{1, \ldots, 2^j\}$, the elements of the vector flows $f_i(z)$ corresponding to the high-priority set-based tasks are required to be in their respective tangent cone in all modes, and the elements corresponding to the low-priority set-based tasks are required to be in the corresponding extended tangent cone only in the modes where that task is inactive.

5. STABILITY ANALYSIS

To study the behavior of the discontinuous differential equations [equation (92)], we look at the corresponding constrained differential inclusion:

$$z \in C \quad \dot{z} \in F(z) \tag{93}$$

where

$$F(z) := \bigcap_{\delta > 0} \overline{\mathrm{co} f((z + \delta\mathbb{B}) \cap C)} \quad \forall z \in C. \tag{94}$$

$\mathbb{B}$ is a unit ball in $\mathbb{R}^{\dim(z)}$ centered at the origin. $\mathrm{co}\overline{P}$ denotes the closed convex hull of the set P, or in other words, the smallest closed convex set containing P. The state evolution of z is the Krasovskii solution of the discontinuous differential equation (55) [Definition 4.2 (Goebel et al., 2012)].

5.1. Convergence of Equality Tasks

The first part of the stability proof considers the convergence of the system equality tasks when switching between 2^j possible modes of the system.

If V is a continuously differentiable Lyapunov function for $\dot{z} = f_i(z)$ for all $i \in \{1, \ldots, 2^j\}$, then V is a Lyapunov function for $\dot{z} \in F(z)$. The following equation holds as all $f_i(z)$ are continuous functions:

$$f \in F(z) \Rightarrow f = \lambda_1 f_1(z) + \lambda_2 f_2(z) + \ldots + \lambda_{2^j} f_{2^j}(z) \qquad (95)$$

for some $\sum_{i=1}^{2^j} \lambda_i = 1, \lambda_i \geq 0$. Consider the Lyapunov function candidate for the equality task errors:

$$V(\tilde{\boldsymbol{\sigma}}_{\text{eb}}) = \frac{1}{2}\tilde{\boldsymbol{\sigma}}_{\text{eb}}^T \tilde{\boldsymbol{\sigma}}_{\text{eb}}. \qquad (96)$$

Using equation (95) and the system equations given in **Tables 1** and **2** for high-priority and low-priority set-based tasks, respectively, we find that the time derivative of V is given by

$$\begin{aligned}\dot{V} &= \tilde{\boldsymbol{\sigma}}_{\text{eb}}^T(\lambda_1(-\boldsymbol{M}_1\tilde{\boldsymbol{\sigma}}_{\text{eb}}) + \lambda_2(-\boldsymbol{M}_2\tilde{\boldsymbol{\sigma}}_{\text{eb}}) + \ldots + \lambda_{2^j}(-\boldsymbol{M}_{2^j}\tilde{\boldsymbol{\sigma}}_{\text{eb}})) \\ &= -\tilde{\boldsymbol{\sigma}}_{\text{eb}}^T(\lambda_1\boldsymbol{M}_1 + \lambda_2\boldsymbol{M}_2 + \ldots + \lambda_{2^j}\boldsymbol{M}_{2^j})\tilde{\boldsymbol{\sigma}}_{\text{eb}} \\ &= -\tilde{\boldsymbol{\sigma}}_{\text{eb}}^T\boldsymbol{Q}\tilde{\boldsymbol{\sigma}}_{\text{eb}}.\end{aligned} \qquad (97)$$

The convex combination $\boldsymbol{Q}$ of positive-definite matrices is also positive definite. Therefore, $\dot{V}$ is negative definite and $\tilde{\boldsymbol{\sigma}}_{\text{eb}} = \boldsymbol{0}$ is a globally asymptotically stable equilibrium point in all modes. Thus, the equality task errors $\tilde{\boldsymbol{\sigma}}_{\text{eb}}$ asymptotically converge to zero when switching between modes. Furthermore, if $\boldsymbol{q}$ belongs to a compact set, the equilibrium point $\tilde{\boldsymbol{\sigma}}_{\text{eb}} = \boldsymbol{0}$ is exponentially stable.

5.2. Satisfaction of Set-Based Tasks and Existence of Solution

The second part of the stability proof considers the satisfaction of the set-based tasks and the existence of a valid solution.

The analysis made in Section 4.2.1 concluded that the evolution of active low-priority set-based tasks is influenced by the higher-priority equality tasks. Unlike the high-priority set-based tasks they cannot be frozen at any given time, and therefore they cannot be guaranteed to be satisfied. We have defined a closed set C in equation (79) within which all high-priority set-based tasks are satisfied at all times. As long as the system solution $z \in C$, these tasks are not violated.

For a system described by equation (92), $\dot{z} = f_1(z)$ as long as the solution $z \in C$ and the lower-priority set-based tasks outside of their respective desired sets do not move further away from them. If the system reaches the boundary of C and remaining in mode 1 would cause z to leave C, another mode is activated. If neither of the vector fields $f_1 - f_{2^j-1}$ will result in z staying in C, the chosen solution is $\dot{z} = f_{2^j}(z)$, for which it has been shown that $\dot{\sigma}_\text{a}$ to $\dot{\sigma}_\text{x} \equiv 0$. Therefore, $f_{2^j}(z) \in T_C(z) \forall z \in C$, and C is strongly forward invariant for $\dot{z} = f_{2^j}(z)$. Thus, there will always exist a solution $z \in C$ and the high-priority set-based tasks are consequently always satisfied.

6. IMPLEMENTATION

This section presents the practical implementation of the proposed algorithm and discusses the computational load of running it.

In the stability analysis, only regulation equality tasks are considered, that is tasks with $\dot{\boldsymbol{\sigma}}_{\text{des}} \equiv \boldsymbol{0}$. For practical purposes, one can also apply this algorithm for time-varying equality tasks as in equation (21). An example of this is presented in Section 7. Furthermore, for the stability analysis, the lower-priority set-based tasks are handled the same way as the high-priority ones and are attempted frozen. However, as this can not be guaranteed, they might exceed their desired set. With this in mind, for practical purposes, these tasks can be handled by defining $\sigma_{\text{des}} = \sigma_{\max}$ and $\sigma_{\text{des}} = \sigma_{\min}$ if $\sigma \geq \sigma_{\max}$ and $\leq \sigma_{\min}$, respectively. Should σ leave the set, the solution will actively attempt to bring σ back to the border of its valid set (rather than simply freezing it outside the set) by having a non-zero error feedback σ as it does in, for instance, equation (56).

In the implementation, the Boolean function `in_T_RC` defined in **Algorithm 1** is used to check if the time-derivative of a set-based task σ with a valid set $C = [\sigma_{\min}, \sigma_{\max}]$ is in the modified tangent cone of C, i.e., if $\dot{\sigma} \in T_{\mathbb{R},C}(\sigma)$. The algorithm `in_T_RC` is illustrated in **Figure 2**.

Note that in the implementation of the algorithm, `in_T_RC` is used as a check both for the high-priority and low-priority set-based task. This is to handle small numerical inaccuracies that result from discretization of a continuous system. Furthermore, given initial conditions outside the valid set C, the chosen implementation is still well-defined for all set-based tasks.

ALGORITHM 1 | The Boolean function `in_T_RC`.

```
Input: $\dot{\sigma}$, $\sigma$, $\sigma_{\min}$, $\sigma_{\max}$
1  if $\sigma_{\min} < \sigma < \sigma_{\max}$ then
2      return True;
3  else if $\sigma \leq \sigma_{\min}$ and $\dot{\sigma} \geq 0$ then
4      return True;
5  else if $\sigma \leq \sigma_{\min}$ and $\dot{\sigma} < 0$ then
6      return False;
7  else if $\sigma \geq \sigma_{\max}$ and $\dot{\sigma} \leq 0$ then
8      return True;
9  else
10     return False;
11 end
```

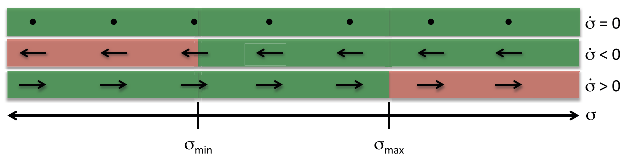

FIGURE 2 | Graphic illustration of the function `in_T_RC` with return value `True` shown in green and `False` in red. The function only returns `False` when σ is outside or on the border of its valid set and the derivative points away from the valid set.

TABLE 3 | Table illustrating the activation of mode based on the Boolean variables as defined in equation (98).

Priorities (i)	a_i	b_i	c_i	...	z_i	Active mode
1	1	1	1	...	1	$\boldsymbol{f}_1$
2	–	1	1	...	1	$\boldsymbol{f}_2$
3	1	–	1	...	1	$\boldsymbol{f}_3$
4	1	1	–	...	1	$\boldsymbol{f}_4$
:	:	:	:	$\ddots$	:	:
$j+1$	1	1	1	...	–	$\boldsymbol{f}_{(j+1)}$
$j+2$	–	–	1	...	1	$\boldsymbol{f}_{(j+2)}$
$j+3$	–	1	–	...	1	$\boldsymbol{f}_{(j+3)}$
:	:	:	:	$\ddots$	:	:
2^j	–	–	–	$\ddots$	–	$\boldsymbol{f}_{2^j}$

The algorithm iterates through the priorities and checks if the Boolean variables satisfy the criteria for activating that mode. The highest-priority possible is activated.

For a system with j set-based tasks, the 2^j modes must be calculated every time step. For practical purposes, it is only necessary to calculate the part of the state vector $\boldsymbol{z}$ that corresponds to $\boldsymbol{q}$, so we calculate this using $\dot{\boldsymbol{q}} = \boldsymbol{f}_i(\boldsymbol{q})$ for $i \in \{1, \ldots, 2^j\}$. Then, to decide what mode to activate, `in_T_RC` is used to check if every inactive set-based task of that mode returns `True`. If it does, that mode is picked. If not, the next mode is considered. Recall that $\dot{\sigma}_\mathrm{k} = \boldsymbol{J}_\mathrm{k}\dot{\boldsymbol{q}}$ and denote the jth set-based task as σ_z. Define the Boolean variables ϕ_i

$$\phi_i \triangleq \texttt{in\_T\_RC}\left(\boldsymbol{J}_\phi \boldsymbol{f}_i, \sigma_\phi, \sigma_{\phi,\min}, \sigma_{\phi,\max}\right) \quad \text{for } i \in \left\{1, \ldots, 2^j\right\} \text{ and } \phi \text{ in } \{\mathrm{a}, \ldots, \mathrm{z}\} \tag{98}$$

The active mode is chosen according to **Table 3**, where the algorithm iterates through priority 1 to 2^j and activates the first mode that satisfies the criteria presented in **Table 3**. The modes are assigned priority according to their restrictiveness. The more set-based tasks are active, the more restrictive it is. Therefore, the least restrictive mode is $\boldsymbol{f}_1$, where no set-based tasks are active. From there, $\boldsymbol{f}_2 - \boldsymbol{f}_{(j+1)}$ are the modes where one set-based task is active, etc. In **Table 3**, a hyphen indicates that the Boolean value does not need to be checked because in that mode, the corresponding set-based task is active and therefore satisfied by definition (in case of a high-priority set-based task) or actively handled to the best of the system's ability (in case of a low-priority set-based task).

6.1. Computational Load

Differently from Escande et al. (2014), the proposed algorithm is deterministic from the computational load aspect. It is possible, in fact, to provide an estimate with respect to the main parameters of the system. In particular, it is seen that the most critical variable for the computational load is the number of DOFs of the system.

The cost of a certain operation is denoted as $c(\cdot)$, the following assumptions have been done:

- sum and subtraction have zero cost,
- the cost of matrix multiplication between a $m \times p$ and a $p \times n$ is proportional to mpn,
- the cost of matrix inversion of a $n \times n$ is proportional to n^3.

Denoting the number of DOFs of the system as n and the generic task dimension as m_x, the following holds:

$$c\left(\boldsymbol{J}_\mathrm{x}^\dagger\right) = \alpha_1 n m_\mathrm{x}^2 + \alpha_2 m_\mathrm{x}^3 \tag{99}$$

$$c\left(\boldsymbol{N}_\mathrm{x}\right) = \alpha_1 n m_\mathrm{x}^2 + \alpha_2 m_\mathrm{x}^3 + \alpha_3 n^2 m_\mathrm{x} \tag{100}$$

$$c\left(\dot{\boldsymbol{q}}_\mathrm{x}\right) = \alpha_1 n m_\mathrm{x}^2 + \alpha_2 m_\mathrm{x}^3 + \alpha_4 n m_\mathrm{x} \tag{101}$$

By considering two generic a and b tasks, each projection within the null-space costs:

$$c\left(\boldsymbol{N}_\mathrm{a}\dot{\boldsymbol{q}}_\mathrm{b}\right) = \alpha_1 n m_\mathrm{a}^2 + \alpha_2 m_\mathrm{a}^3 + \alpha_3 n^2 m_\mathrm{a} + + \alpha_1 n m_\mathrm{b}^2 + \alpha_2 m_\mathrm{b}^3 + \alpha_4 n m_\mathrm{b} + \alpha_5 n^2. \tag{102}$$

In the proposed method, this computational cost is present as many times as there are equality plus active set-based tasks. However, it is easy to recognize that this cost is *small* for the higher-priority task and grows when the lowest priority are projected into the null space of the stacked Jacobian of all the higher priorities. The worst case, from the computational aspect, arises thus when m_a or m_b are as large as possible, i.e., $n - 1$, in which case

$$c\left(\boldsymbol{N}_\mathrm{a}\dot{\boldsymbol{q}}_\mathrm{b}\right) \propto n^3. \tag{103}$$

By assuming that all the DOFs are exploited, it is possible to verify that this cost arises both in the classical equality-task-priority inverse kinematics and in the extension to set-based tasks presented here.

The recursive computation of the null spaces given in Antonelli et al. (2015) does not change the dependence of the cost with respect to the number of DOFs of the system, since it implies a matrix multiplication between two $n \times n$ matrices whose cost still is proportional to n^3.

Overall, the cost is proportional to n^3.

One additional computation cost arising for the set-based extension in this paper is the check whether or not one of the possible solutions is violating one of the set-based tasks. This implies re-projection of $\dot{\boldsymbol{q}}_\mathrm{des}$ by means of $\boldsymbol{J}_\mathrm{x}$ at an individual cost of $n m_\mathrm{x}$. However, we only consider scalar set-based tasks, for which $m_\mathrm{x} = 1$ and the cost is proportional to n.

Furthermore, for j set-based tasks, in the worst-case scenario, it is necessary to compute 2^j possible solutions. However, in many cases, some of the 2^j modes have common terms, which will reduce the overall cost, since these terms only need to be computed once. Furthermore, simulations and experimental results confirm that the algorithm is feasible and sufficiently fast for real-time applications.

It is also worth noticing that optimized algorithms for matrix inversion exist but are not considered here. In fact, optimized methods usually exhibit different performances based on the input, and in general, they have a smaller cost than the costs considered here.

7. EXPERIMENTAL RESULTS

This section presents experimental results that were obtained by running the proposed algorithm on a 6 DOF manipulator from Universal Robots called UR5. Two examples are presented that illustrate the implementation and effectiveness of the proposed method.

7.1. UR5 and Control Setup

The UR5 is a manipulator with 6 revolute joints, and the joint angles are denoted $\boldsymbol{q} \triangleq \begin{bmatrix} q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{bmatrix}^{\mathrm{T}}$. In this paper, the Denavit–Hartenberg (D–H) parameters are used to calculate the forward kinematics. The parameters are given in Wu et al. (2014) and are presented **Table 4** with the corresponding coordinate frames illustrated in **Figure 3**. The resulting forward kinematics has been experimentally verified to confirm the correctness of the parameters.

The UR5 is equipped with a high-level controller that can control the robot both in joint and Cartesian space. In the experiments presented here, a calculated reference $\boldsymbol{q}_{\mathrm{des}}$ is sent to the high-level controller, which is assumed to function nominally such that

$$\boldsymbol{q} \approx \boldsymbol{q}_{\mathrm{des}}. \tag{104}$$

From this reference, $\dot{\boldsymbol{q}}_{\mathrm{des}}$ and $\ddot{\boldsymbol{q}}_{\mathrm{des}}$ are extrapolated and sent with $\boldsymbol{q}_{\mathrm{des}}$ to the low-level controller.

The structure of the system is illustrated in **Figure 4**. The algorithm described in Section 6 is implemented in the kinematic controller block. For every time step, a reference for the joint velocities is calculated and integrated to desired joint angles $\boldsymbol{q}_{\mathrm{des}}$. This is used as input to the dynamic controller, which in turn applies torques to the joint motors. Note that the actual state $\boldsymbol{q}$ is not used for feedback to the kinematic control block. When the current state is used as input for the kinematic controller, the kinematic and dynamic loops are coupled and the gains designed for the kinematic control alone to satisfy Assumption 1 can not be used. This result in uneven motion, and therefore the kinematic control block receives the previous reference as feedback, which leads to much nicer behavior, and is a good approximation because the dynamic controller tracks the reference with very high precision. This is a standard method of implementation for industry robots when kinematic control is used.

The communication between the implemented algorithm and the industrial manipulator system occurs through a TCP/IP connection, which operates at 125 Hz. The kinematic control block is implemented using python, which is a very suitable programing language for the task. The TCP/IP connection is very simple to set up in python. Furthermore, python has several libraries that can handle different math and matrix operations.

7.2. Implemented Tasks

Three tasks make up the basis for the experiments: position control, collision avoidance, and field of view (FOV). Position control is implemented as both an equality and set-based task and collision avoidance and FOV as set-based tasks.

7.2.1. Position Control

The position of the end-effector relative to the base coordinate frame is given by the forward kinematics. The analytical expression can be found through the homogeneous transformation matrix (Spong et al., 2005) using the D–H parameters given in **Table 4**. The task is then defined by

$$\boldsymbol{\sigma}_{\mathrm{pos}} = \boldsymbol{f}(\boldsymbol{q}) \in \mathbb{R}^3 \tag{105}$$

$$\dot{\boldsymbol{\sigma}}_{\mathrm{pos}} = \boldsymbol{J}_{\mathrm{pos}}(\boldsymbol{q})\dot{\boldsymbol{q}} = \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{q}}\dot{\boldsymbol{q}}, \tag{106}$$

where the function $\boldsymbol{f}(\boldsymbol{q})$ is given by the forward kinematics.

TABLE 4 | Table of the D–H parameters of the UR5.

Joint	a_i [m]	α_i [rad]	d_i [m]	θ_i [rad]
1	0	$\pi/2$	0.089	q_1
2	−0.425	0	0	q_2
3	−0.392	0	0	q_3
4	0	$\pi/2$	0.109	q_4
5	0	$-\pi/2$	0.095	q_5
6	0	0	0.082	q_6

The corresponding coordinate systems can be seen in ***Figure 3***.

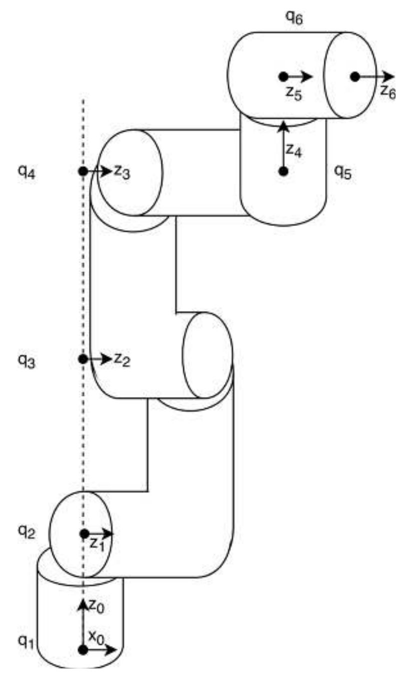

FIGURE 3 | Coordinate frames corresponding to the D–H parameters in Table 4.

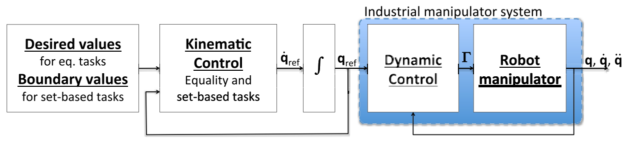

FIGURE 4 | The control structure of the experiments. The tested algorithm is implemented in the kinematic controller block.

7.2.2. Collision Avoidance

To avoid a collision between the end effector and an object at position $\boldsymbol{p}_o \in \mathbb{R}^3$, the distance between them is used as a task:

$$\sigma_{\mathrm{CA}} = \sqrt{(\boldsymbol{p}_o - \boldsymbol{\sigma}_{\mathrm{pos}})^{\mathrm{T}}(\boldsymbol{p}_o - \boldsymbol{\sigma}_{\mathrm{pos}})} \in \mathbb{R} \tag{107}$$

$$\dot{\sigma}_{\mathrm{CA}} = \boldsymbol{J}_{\mathrm{CA}}(\boldsymbol{q})\dot{\boldsymbol{q}} = -\frac{(\boldsymbol{p}_o - \boldsymbol{\sigma}_{\mathrm{pos}})^{\mathrm{T}}}{\sigma_{\mathrm{CA}}}\boldsymbol{J}_{\mathrm{pos}}(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{108}$$

7.2.3. Field of View

The field of view is defined as the outgoing vector of the end effector, i.e., the z_6-axis in **Figure 3**. This vector expressed in base coordinates is denoted $a \in \mathbb{R}^3$ and can be found through the homogeneous transformation matrix using the D–H parameters:

$$a = g(q) \in \mathbb{R}^3 \tag{109}$$

$$\dot{a} = J_{\mathrm{FOV,3DOF}}(q)\dot{q} = \frac{\mathrm{d}g}{\mathrm{d}q}\dot{q} \tag{110}$$

FOV is a useful task when directional devices or sensors are mounted on the end effector, and they are desired to point in a certain direction $a_{\mathrm{des}} \in \mathbb{R}^3$. The task is defined as the norm of the error between $\boldsymbol{a}$ and $\boldsymbol{a}_{\mathrm{des}}$:

$$\sigma_{\mathrm{FOV}} = \sqrt{(\boldsymbol{a}_{\mathrm{des}} - \boldsymbol{a})^{\mathrm{T}}(\boldsymbol{a}_{\mathrm{des}} - \boldsymbol{a})} \in \mathbb{R} \tag{111}$$

$$\dot{\sigma}_{\mathrm{FOV}} = J_{\mathrm{FOV}}(q)\dot{q} = -\frac{(\boldsymbol{a}_{\mathrm{des}} - \boldsymbol{a})^{\mathrm{T}}}{\sigma_{\mathrm{FOV}}} J_{\mathrm{FOV,3DOF}}(q)\dot{q} \tag{112}$$

Note in equations (108) and (112) that J_{CA} and J_{FOV} are not defined for $\sigma_{\mathrm{CA}} = 0$ and $\sigma_{\mathrm{FOA}} = 0$, respectively. In the implementation, this is solved by adding a small $\varepsilon > 0$ to the denominator of these two Jacobians, thereby ensuring that division by zero does not occur. Furthermore, the method presented in this paper ensures collision avoidance; hence, σ_{CA} will never be zero. Also, note that alternative FOV functions exist which do not suffer from this representation singularity.

7.3. Example 1

In this example, the system has been given two waypoints for the end effector to reach. This is the sole equality task σ_1 of this example:

$$p_{\mathrm{w1}} = [0.486\ \mathrm{m} \quad -0.066\ \mathrm{m} \quad -0.250\ \mathrm{m}]^{\mathrm{T}}, \text{ and} \tag{113}$$

$$p_{\mathrm{w2}} = [0.320\ \mathrm{m} \quad 0.370\ \mathrm{m} \quad -0.250\ \mathrm{m}]^{\mathrm{T}}. \tag{114}$$

A circle of acceptance (COA) of 0.02 m is implemented for switching from $\boldsymbol{\sigma}_{1,\mathrm{des}} = \boldsymbol{\sigma}_{\mathrm{pos,des}} = p_{\mathrm{w1}}$ to $\boldsymbol{\sigma}_{1,\mathrm{des}} = \boldsymbol{\sigma}_{\mathrm{pos,des}} = p_{\mathrm{w2}}$. The task gain matrix has been chosen as

$$\boldsymbol{\Lambda}_{\mathrm{pos}} = \mathrm{diag}([0.3 \quad 0.3 \quad 0.3]). \tag{115}$$

Furthermore, two obstacles have been introduced; hence, two collision avoidance tasks are necessary. In this example, these tasks are high-priority and are denoted σ_{a} and σ_{b}, respectively. The obstacles are positioned at

$$p_{\mathrm{o1}} = [0.40\ \mathrm{m} \quad -0.25\ \mathrm{m} \quad -0.33\ \mathrm{m}]^{\mathrm{T}}, \text{ and} \tag{116}$$

$$p_{\mathrm{o2}} = [0.40\ \mathrm{m} \quad 0.15\ \mathrm{m} \quad -0.33\ \mathrm{m}]^{\mathrm{T}}, \tag{117}$$

and have a radius of 0.18 and 0.15 m, respectively. This radius is used as the minimum value of the set-based collision avoidance task to ensure that the end effector is never closer to the obstacle center than the allowed radius, see **Table 5**. Because the task is only considered as a high-priority set-based task, it is not necessary to choose a task gain.

FOV is implemented as a low-priority set-based task $\sigma_{\mathrm{c}} = \sigma_{\mathrm{FOV}}$, with a maximum value to limit the error between $\boldsymbol{a}$ and $\boldsymbol{a}_{\mathrm{des}}$.

TABLE 5 | Implemented tasks in example 1 sorted by decreasing priority.

Name	Task description	Type	Valid set *C*
σ_{a}	Collision avoidance	Set-based	$C_{\mathrm{a}} = [0.18, \infty)$
σ_{b}	Collision avoidance	Set-based	$C_{\mathrm{b}} = [0.15, \infty)$
$\boldsymbol{\sigma}_1$	Position	Equality	–
σ_{c}	Field of view	Set-based	$C_{\mathrm{c}} = [0, 0.2622]$

Note that since σ_c is defined as the norm of the error between desired and actual field of view vector, it is by definition greater than or equal to zero, which is the lower boundary of C_c. The maximum boundary is given by the maximum allowed error between these two vectors, in this case corresponding to 15° between them.

Here, the maximum value for the set-based FOV task is set as 0.2622. This corresponds to allowing the angle between $\boldsymbol{a}_{\mathrm{des}}$ and $\boldsymbol{a}$ being 15° or less. If the FOV error exceeds, the maximum value of 0.2622, rather than attempting to freeze the task at its current value, an effort is made to push it back to the boundary of the valid set:

$$\tilde{\sigma}_{\mathrm{c}} = \sigma_{\mathrm{FOV,max}} - \sigma_{\mathrm{c}} = 0.2622 - \sigma_{\mathrm{FOV}} \tag{118}$$

with

$$\boldsymbol{a}_{\mathrm{des}} \equiv \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}}. \tag{119}$$

The gain for this task is $\boldsymbol{\Lambda}_{\mathrm{FOV}} = 1$. The priority of the tasks is given below in the table below.

A system with 3 set-based tasks normally have $2^3 = 8$ modes to consider. However, in this case, the two obstacles have no points of intersection. Hence, it will never be necessary to freeze both σ_{a} and σ_{b} simultaneously, and thus the system has 6 modes:

$$\text{Mode 1}: \dot{q}_{\mathrm{des}} = f_1 \triangleq J_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{120}$$

$$\text{Mode 2}: \sigma_{\mathrm{a}} \text{ active}, \dot{q}_{\mathrm{des}} = f_2 \triangleq N_{\mathrm{a}}J_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 \tag{121}$$

$$\text{Mode 3}: \sigma_{\mathrm{b}} \text{ active}, \dot{q}_{\mathrm{des}} = f_3 \triangleq N_{\mathrm{b}}J_1^{\dagger}\Lambda_1\tilde{\sigma}_1 \tag{122}$$

$$\text{Mode 4}: \sigma_{\mathrm{c}} \text{ active}, \dot{q}_{\mathrm{des}} = f_4 \triangleq J_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + N_1J_{\mathrm{c}}^{\dagger}\Lambda_{\mathrm{c}}\tilde{\sigma}_{\mathrm{c}} \tag{123}$$

$$\text{Mode 5}: \sigma_{\mathrm{a}}, \sigma_{\mathrm{c}} \text{ active}, \dot{q}_{\mathrm{des}} = f_5 \triangleq N_{\mathrm{a}}J_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + N_{\mathrm{a1}}^{\mathrm{A}}J_{\mathrm{c}}^{\dagger}\Lambda_{\mathrm{c}}\tilde{\sigma}_{\mathrm{c}} \tag{124}$$

$$\text{Mode 6}: \sigma_{\mathrm{b}}, \sigma_{\mathrm{c}} \text{ active}, \dot{q}_{\mathrm{des}} = f_6 \triangleq N_{\mathrm{b}}J_1^{\dagger}\boldsymbol{\Lambda}_1\tilde{\boldsymbol{\sigma}}_1 + N_{\mathrm{b1}}^{\mathrm{A}}J_{\mathrm{c}}^{\dagger}\Lambda_{\mathrm{c}}\tilde{\sigma}_{\mathrm{c}} \tag{125}$$

Denote the Boolean variables

$$\begin{aligned} \phi_i &\triangleq \texttt{in\_T\_RC}\ (J_\phi f_i, \sigma_\phi, \sigma_{\phi,\min}, \sigma_{\phi,\max}) \\ &\quad \text{for } i \in \{1, \ldots, 6\} \text{ and } \phi \in \{\mathrm{a}, \ldots, \mathrm{c}\} \end{aligned} \tag{126}$$

The active mode is then chosen by **Table 6**.

The results of Example 1 are shown in **Figure 5**. The position task is fulfilled as predicted by the theory, i.e., the two waypoints are reached by the end effector. **Figures 5A,B** confirm this. In **Figure 5B**, it can be seen that the error between the actual and desired end position converges toward zero. Then, at around $t = 12$ s, the end effector is within the circle of acceptance of p_{w1}. Thus, the desired position switches to p_{w2}. The end effector immediately starts moving toward the new waypoint

TABLE 6 | Table illustrating the activation of mode in example 1.

Priorities (*i*)	a_i	b_i	c_i	Active mode
1	1	1	1	$\boldsymbol{f}_1$
2	–	1	1	$\boldsymbol{f}_2$
3	1	–	1	$\boldsymbol{f}_3$
4	1	1	–	$\boldsymbol{f}_4$
5	–	1	–	$\boldsymbol{f}_5$
6	1	–	–	$\boldsymbol{f}_6$

Note that the tasks σ_a and σ_b are never active in the same mode (an active task is indicated by a hyphen) because the tasks are collision avoidance tasks in the case where the obstacles are not overlapping.

and converges to it at around $t = 35$ s. Furthermore, the end effector avoids the two obstacles by locking the distance to the obstacle center at the obstacle radius until the other active tasks drive the end effector away from the obstacle center on their own accord. This can be seen in **Figure 5A**, and it is also confirmed by **Figure 5D**: the set-based collision avoidance tasks never exceed the valid sets C_{a} and C_{b}, but freeze on the boundary of these sets.

Figure 5C displays the active mode over time and confirms that mode changes coincide with set-based tasks either being activated (frozen on boundary/leaving valid set) or deactivated (unfrozen/approaching valid set). An increase in mode means that a new set-based task has been activated and vice versa.

Even though the initial condition of σ_{c} is outside C_{c}, the task is not active from $t = 0$ s because other, less restrictive modes, naturally bring the task closer to and eventually into C_{c}. However, at around $t = 13$ s, the system enters mode 4 and activates σ_{c} because mode 1–3 would all lead to σ_{c} leaving C_{c}. Due to the fact that σ_{c} is a low-priority set-based task, it still exceeds its maximum value in spite of the system activating the task, as predicted by the theory. However, by keeping the task active, eventually σ_{c} converges back to the boundary of C_{c}.

7.4. Example 2

In this example, the system has been given a time-varying trajectory for the end effector to track:

$$\boldsymbol{\sigma}_{1,\mathrm{des}}(t) = \boldsymbol{\sigma}_{\mathrm{pos,des}}(t) = \begin{bmatrix} 0.5\sin^2(0.1t) + 0.2 \\ 0.5\cos(0.1t) + 0.25\sin(0.1t) \\ 0.5\sin(0.1t)\cos(0.1t) + 0.1 \end{bmatrix} \tag{127}$$

The task gain matrix has been chosen as

$$\boldsymbol{\Lambda}_{\mathrm{pos}} = \mathrm{diag}([0.15 \quad 0.15 \quad 0.15]). \tag{128}$$

In addition, the system is also given a valid workspace: a minimum and maximum value for the x, y and z position of the end effector. This is implemented as three set-based tasks that are equal to the first, second, and third element of $\boldsymbol{\sigma}_{\mathrm{pos}}$, respectively. These set-based tasks are all implemented as high-priority, as shown in the **Table 7** below.

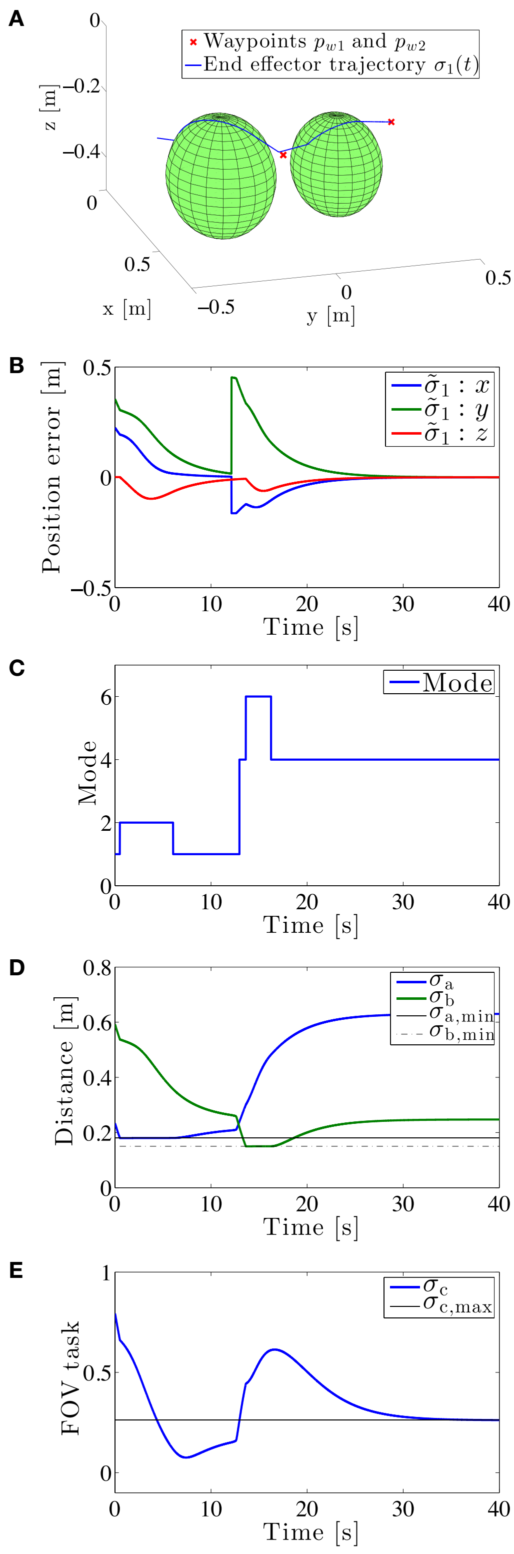

FIGURE 5 | Logged data from Example 1. (A) End-effector trajectory. The two waypoints plotted in red. The figure shows that the end-effector moves around the obstacles to reach its desired position. **(B)** Position error $\tilde{\sigma}_1$ in *x*, *y*, and *z*-direction. **(C)** Active mode over time. **(D)** End effector distance to the two obstacle centers. **(E)** Field of view task.

A system with 3 set-based tasks has $2^3 = 8$ modes to consider:

$$\text{Mode 1}: \dot{q}_{\text{des}} = f_1 \triangleq J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{129}$$

$$\text{Mode 2}: \sigma_{\text{a}} \text{ active}, \dot{q}_{\text{des}} = f_2 \triangleq N_{\text{a}}J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{130}$$

$$\text{Mode 3}: \sigma_{\text{b}} \text{ active}, \dot{q}_{\text{des}} = f_3 \triangleq N_{\text{b}}J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{131}$$

$$\text{Mode 4}: \sigma_{\text{c}} \text{ active}, \dot{q}_{\text{des}} = f_4 \triangleq N_{\text{c}}J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{132}$$

$$\text{Mode 5}: \sigma_{\text{a}}, \sigma_{\text{b}} \text{ active}, \dot{q}_{\text{des}} = f_5 \triangleq N_{\text{ab}}^{\text{A}}J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{133}$$

$$\text{Mode 6}: \sigma_{\text{a}}, \sigma_{\text{c}} \text{ active}, \dot{q}_{\text{des}} = f_6 \triangleq N_{\text{ac}}^{\text{A}}J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{134}$$

$$\text{Mode 7}: \sigma_{\text{b}}, \sigma_{\text{c}} \text{ active}, \dot{q}_{\text{des}} = f_7 \triangleq N_{\text{bc}}^{\text{A}}J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{135}$$

$$\text{Mode 8}: \sigma_{\text{a}}, \sigma_{\text{b}}, \sigma_{\text{c}} \text{ active}, \dot{q}_{\text{des}} = f_8 \triangleq N_{\text{abc}}^{\text{A}}J_1^{\dagger}\Lambda_1\tilde{\boldsymbol{\sigma}}_1 \tag{136}$$

Denote the Boolean variables

$$\phi_i \triangleq \texttt{in\_T\_RC}\left(J_\phi f_i, \sigma_\phi, \sigma_{\phi,\min}, \sigma_{\phi,\max}\right) \\ \text{for } i \in \{1, \ldots, 8\} \text{ and } \phi \in \{\text{a}, \ldots, \text{c}\} \tag{137}$$

The active mode is then chosen according to **Table 8**.

The results of Example 2 are shown in **Figures 6** and **7**. The system is unable to track the equality task perfectly, as the desired trajectory moves in and out of the allowed workspace (**Figures 6A** and **7**). Mathematically, this is explained by the fact that the set-based tasks and the position equality task are not linearly independent (in fact, they are equal to each other). Thus, Assumption 1 is not fulfilled, and the equality task errors can no longer be guaranteed to converge to zero. However, as seen in **Figure 6C**, the end effector stays within the valid workspace at all times by freezing on the boundary when following the trajectory would bring it outside of the valid workspace. **Figure 6B** displays the active mode over time. Mode changes clearly correspond to set-based tasks being activated/deactivated on the border of their valid sets.

Videos of the experimental results can be viewed online.[1]

[1]https://www.dropbox.com/sh/x92agg4n7ly9hdc/AAC-xh8DyJM5X6jknJhyoJdca?dl=0

TABLE 7 | Implemented tasks in example 2 sorted by decreasing priority.

Name	Task description	Type	Valid set C
σ_a	Limited workspace x-direction	Set-based	$C_a = [0.1, 0.6]$
σ_b	Limited workspace y-direction	Set-based	$C_b = [-0.5, 0.4]$
σ_c	Limited workspace z-direction	Set-based	$C_c = [-0.3, 0.25]$
$\boldsymbol{\sigma}_1$	Position	Equality	–

TABLE 8 | Table illustrating the activation of mode in example 2.

Priorities (i)	a_i	b_i	c_i	Active mode
1	1	1	1	f_1
2	–	1	1	f_2
3	1	–	1	f_3
4	1	1	–	f_4
5	–	–	2	f_5
6	–	1	–	f_6
7	1	–	–	f_7
8	–	–	–	f_8

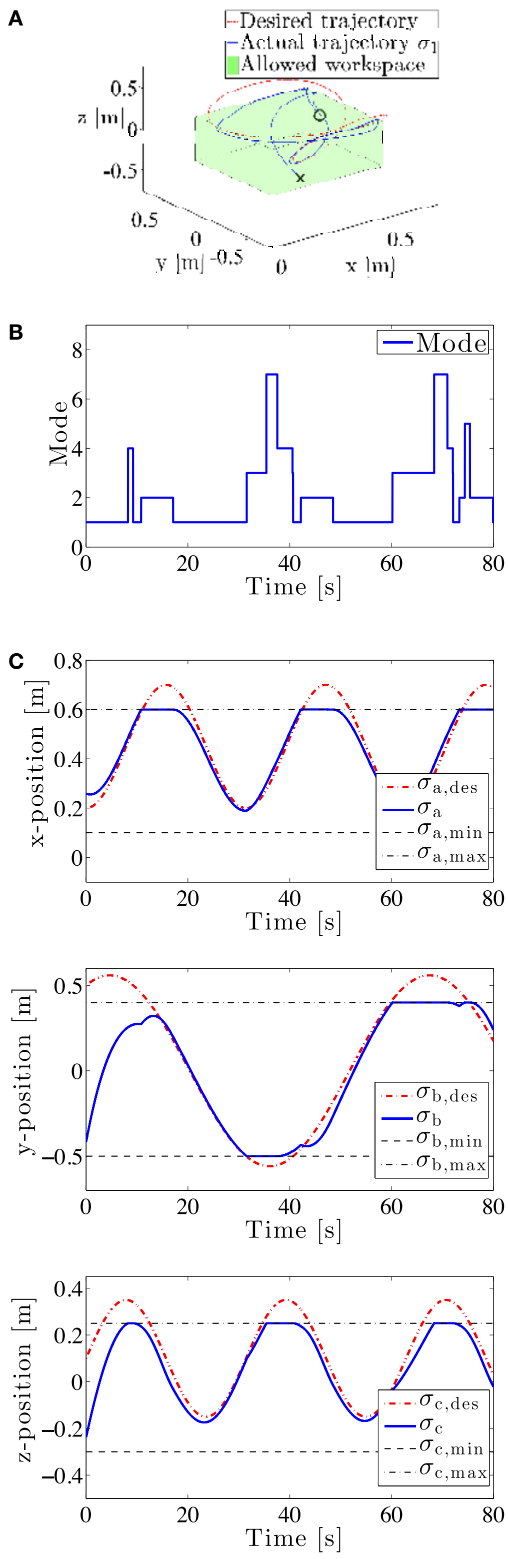

FIGURE 6 | Logged data from example 2. (A) Desired and actual end trajectory plotted in red and blue, respectively. The allowed workspace is illustrated by the green box. Initial and end position marked by x and o, respectively. **(B)** Active mode over time. **(C)** Reference and actual end effector position in x, y, and z-direction with valid workspace limits.

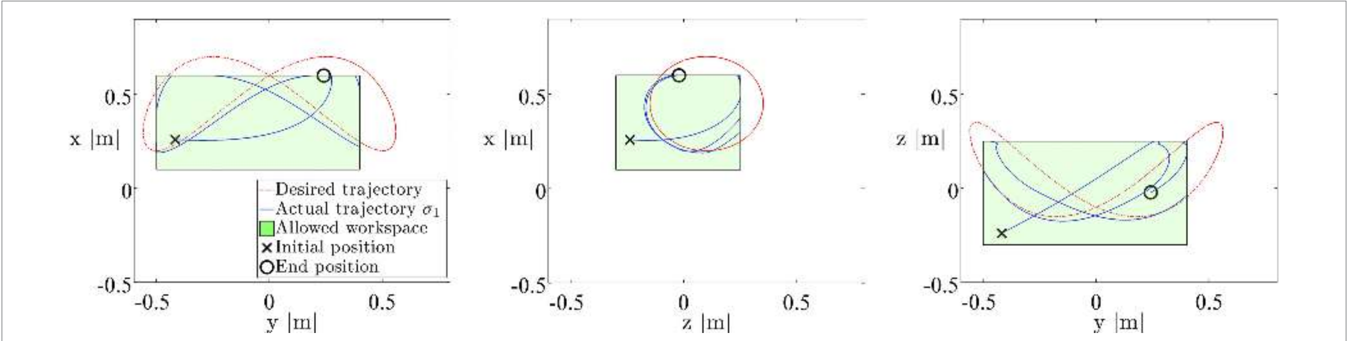

FIGURE 7 | Actual vs. desired end effector trajectory from example 1 plotted in the *xy*-, *xz*-, and *yz*-plane, respectively. A three-dimensional plot is given in **Figure 6A**.

8. CONCLUSION

A general robotic system is controlled in joint space but has desired behavior (tasks) specified in task space. The singularity-robust multiple task-priority inverse kinematics framework has been developed as a way to generate a reference trajectory in joint space to fulfill several tasks in a prioritized order simultaneously. This framework has been developed for *equality tasks*, which are tasks that assign an exact desired value to the controlled variable (e.g., the end-effector configuration). However, for a general robotic system, several goals may be described as *set-based tasks*, i.e., tasks with a desired interval [*area of satisfaction* (Escande et al., 2014)]. Two classic, yet important examples are joint limits and obstacle avoidance. Additional examples are manipulability, dexterity, and field of view (for directional sensors, such as a camera mounted on the end effector).

This paper presents an extension to the singularity-robust multiple task-priority inverse kinematics framework that enables set-based tasks to be handled directly. The proposed method allows a general number of scalar set-based tasks to be handled with a given priority within a number of equality tasks. The main purpose of this method is to fulfill the system's equality tasks while ensuring that the set-based tasks are always satisfied, i.e., contained in their valid set. A mathematical framework is presented and concluded with a stability analysis, in which it is proven that high-priority set-based tasks remain in their valid set at all times, whereas lower-priority set-based tasks cannot be guaranteed to be satisfied due to the influence of the higher-priority equality tasks. Furthermore, it is proven that the equality task errors converge asymptotically to zero given certain assumptions.

A practical implementation of the proposed algorithm is presented along with an analysis of the computational cost. Finally, experimental results are presented where two examples have been implemented on a UR5 manipulator. The experimental results confirm the theory and prove the validity of the proposed method for practical purposes.

Future work includes expanding the algorithm to achieve smooth transitions between switches while maintaining the strict priority of all tasks and still guaranteeing satisfaction of the high-priority set-based tasks.

AUTHOR CONTRIBUTIONS

SM: writing the paper, developing the theory, and running experiments. GA: writing the paper, developing the theory, providing feedback to theoretical and experimental results, and feedback on written paper. ART: developing the theory and feedback on written paper. KYP: developing the theory, providing feedback to theoretical and experimental results, and feedback on written paper. JS: running experiments and feedback on written paper.

FUNDING

This work was supported by the Research Council of Norway through the Center of Excellence funding scheme, project number 223254, by the Italian *Ministero dell'Istruzione*, *dell'Università e della Ricerca*, *PRIN 2010–2011* through the project MARIS (protocol 2010FBLHRJ) and by the European Community through the projects ARCAS (FP7-287617), EuRoC (FP7-608849), DexROV (H2020-635491), and AEROARMS (H2020-644271).

REFERENCES

Antonelli, G. (2008). "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," in *Proc. 2008 IEEE International Conference on Robotics and Automation* (Pasadena, CA: IEEE), 1993–1998.

Antonelli, G. (2009). Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Trans. Robot.* 25, 985–994. doi:10.1109/TRO.2009.2017135

Antonelli, G. (2014). *Underwater Robots*, 3rd Edn. Heidelberg: Springer Tracts in Advanced Robotics, Springer-Verlag.

Antonelli, G., Arrichiello, F., and Chiaverini, S. (2008). The null-space-based behavioral control for autonomous robotic systems. *J. Intell. Serv. Robot.* 1, 27–39. doi:10.1007/s11370-007-0002-3

Antonelli, G., Moe, S., and Pettersen, K. (2015). "Incorporating set-based control within the singularity-robust multiple task-priority inverse kinematics," in *Proc. 23th Mediterranean Conference on Control and Automation* (Torremolinos: IEEE), 1132–1137.

Azimian, H., Looi, T., and Drake, J. (2014). "Closed-loop inverse kinematics under inequality constraints: application to concentric-tube manipulators," in *Proc. 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)* (Chicago, IL: IEEE), 498–503.

Baizid, K., Giglio, G., Pierri, F., Trujillo, M., Antonelli, G., Caccavale, F., et al. (2015). "Experiments on behavioral coordinated control of an unmanned aerial vehicle manipulator system," in *IEEE International Conference on Robotics and Automation (ICRA)* (Seattle, WA: IEEE), 4680–4685.

Buss, S. R. (2009). *Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares Methods*. San Diego, CA: University of California. Available at: https://www.math.ucsd.edu/∼sbuss/ResearchWeb/ikmethods/iksurvey.pdf

Chiaverini, S. (1997). Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Trans. Robot. Autom.* 13, 398–410. doi:10.1109/70.585902

Chiaverini, S., Oriolo, G., and Walker, I. D. (2008). "Kinematically redundant manipulators," in *Springer Handbook of Robotics*, eds B. Siciliano and O. Khatib (Heidelberg: Springer-Verlag), 245–268.

de Lasa, M., Mordatch, I., and Hertzmann, A. (2010). Feature-based locomotion controllers. *ACM Trans. Graph.* 29, 131:1–131:10. doi:10.1145/1778765.1781157

Escande, A., Mansard, N., and Wieber, P.-B. (2014). Hierarchical quadratic programming: fast online humanoid-robot motion generation. *Int. J. Robot. Res.* 33, 1006–1028. doi:10.1177/0278364914521306

Faverjon, B., and Tournassoud, P. (1987). "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Proc. 1987 IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 4 (Raleigh, NC: IEEE), 1152–1159.

Goebel, R., Sanfelice, R., and Teel, A. (2012). *Hybrid Dynamical Systems: Modeling, Stability, and Robustness*. Princeton, NJ: Princeton University Press.

Golub, G., and Van Loan, C. (1996). *Matrix Computations*, 3rd Edn. Baltimore, MD: The Johns Hopkins University Press.

Ioannou, P., and Sun, J. (1996). *Robust Adaptive Control*. Upper River Saddle, NJ: Prentice Hall, Inc.

Kanoun, O., Lamiraux, F., and Wieber, P. (2011). Kinematic control of redundant manipulators: generalizing the task-priority framework to inequality task. *IEEE Trans. Robot.* 27, 785–792. doi:10.1109/TRO.2011.2142450

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.* 5, 90. doi:10.1177/027836498600500106

Liégeois, A. (1977). Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Syst. Man Cybern.* 7, 868–871. doi:10.1109/TSMC.1977.4309644

Maciejewski, A., and Klein, C. (1985). Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *Int. J. Robot. Res.* 4, 109–117. doi:10.1177/027836498500400308

Mansard, N., and Chaumette, F. (2007). Task sequencing for high-level sensor-based control. *IEEE Trans. Robot. Autom.* 23, 60–72. doi:10.1109/TRO.2006.889487

Mansard, N., Khatib, O., and Kheddar, A. (2009). A unified approach to integrate unilateral constraints in the stack of tasks. *IEEE Trans. Robot.* 25, 670–685. doi:10.1109/TRO.2009.2020345

Moe, S., Antonelli, G., Pettersen, K. Y., and Schrimpf, J. (2015a). "Experimental results for set-based control within the singularity-robust multiple task-priority inverse kinematics framework," in *Proc. 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO 2015)* (Zhuhai).

Moe, S., Teel, A., Antonelli, G., and Pettersen, K. (2015b). "Stability analysis for set-based control within the singularity-robust multiple task-priority inverse kinematics framework," in *Proc. 54th IEEE Conference on Decision and Control* (Osaka).

Nakamura, Y., Hanafusa, H., and Yoshikawa, T. (1987). Task-priority based redundancy control of robot manipulators. *Int. J. Robot. Res.* 6, 3–15. doi:10.1177/027836498700600103

Siciliano, B. (1990). Kinematic control of redundant robot manipulators: a tutorial. *J. Intell. Robot. Syst.* 3, 201–212. doi:10.1007/BF00126069

Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics: Modelling, Planning and Control*. Springer-Verlag.

Siciliano, B., and Slotine, J.-J. (1991). "A general framework for managing multiple tasks in highly redundant robotic systems," in *Proc. 5th International Conference on Advanced Robotics* (Pisa: IEEE), 1211–1216.

Simetti, E., Casalino, G., Torelli, S., Sperindé, A., and Turetta, A. (2013). Floating underwater manipulation: developed control methodology and experimental validation within the TRIDENT project. *J. Field Robot.* 31, 364–385. doi:10.1002/rob.21497

Spong, M., Hutchinson, S., and Vidyasagar, M. (2005). *Robot Modeling and Control*. Hoboken, NJ: Wiley.

Whitney, D. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man Mach. Syst.* 10, 47–52. doi:10.1109/TMMS.1969.299896

Wu, H., Tizzano, W., Andersen, T. T., Andersen, N. A., and Ravn, O. (2014). Hand-eye calibration and inverse kinematics of robot arm using neural network. *Adv. Intell. Syst. Comput.* 274, 581–591. doi:10.1007/978-3-319-05582-4_50

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.