A Set-Theoretic Approach to Multi-Task Execution and Prioritization

Gennaro Notomista[1], Siddharth Mayya[1], Mario Selvaggio[2], María Santos[1], and Cristian Secchi[3]

***Abstract*— Executing multiple tasks concurrently is important in many robotic applications. Moreover, the prioritization of tasks is essential in applications where safety-critical tasks need to precede application-related objectives, in order to protect both the robot from its surroundings and vice versa. Furthermore, the possibility of switching the priority of tasks during their execution gives the robotic system the flexibility of changing its objectives over time. In this paper, we present an optimization-based task execution and prioritization framework that lends itself to the case of time-varying priorities as well as variable number of tasks. We introduce the concept of *extended set-based tasks*, encode them using control barrier functions, and execute them by means of a constrained-optimization problem, which can be efficiently solved in an online fashion. Finally, we show the application of the proposed approach to the case of a redundant robotic manipulator.**

I. INTRODUCTION

The ability of executing multiple tasks simultaneously constitutes an essential aspect of many robotic systems. Indeed, it becomes necessary in all those cases where, besides the accomplishment of application-related goals, safety requirements have to be fulfilled. The concept of *redundancy* (see, e.g., [1]) is what makes the concurrent execution of a set of tasks feasible. If a robot is redundant with respect to a task, there exist multiple configurations of the robot that allow it to achieve that task. Within a set of tasks that are to be executed simultaneously, it is generally desirable to establish a *prioritized stack*. For instance, if tasks are safety-critical—such as avoiding joint limits of robotic manipulators, or obstacles in the case of mobile robots—they must take precedence over the execution of other objectives—e.g., visual servoing or trajectory tracking [2].

In this paper, we present an optimization-based control framework that allows robots to execute and prioritize a set of tasks. Moreover, this formulation lends itself to be applied to the case of time-varying task priorities as well as variable number of tasks to be executed. As a matter of fact, the prioritization order within the stack of tasks need not be unyielding. The importance of some tasks over others may evolve over time to reflect endogenous changes of the system—related, for instance, to changes in the application objective—or exogenous factors—such as human teleoperators in shared-control applications [3]. Due to the fact that the discrete nature of priority switches may induce discontinuities in the robot controller (see, e.g., [4]), the problem of time-varying priorities has been addressed in many multi-task execution frameworks, as will be discussed more in detail in the next section. Owing to its formulation, the approach presented in this paper intrinsically allows task priorities to be swapped, and tasks to be inserted and removed, in a continuous fashion.

By extending the definition of set-based tasks [5], we propose an optimization-based approach that encodes tasks by means of control barrier functions [6]. The latter have been introduced in their modern formulation in [7] for ensuring safety of dynamical systems, intended as the forward invariance property of a subset of the state space of the system. Additionally, in [8], it has been shown that they can be also employed to achieve set stability. In the approach proposed in this paper, we leverage these two properties, and extend them to the time-varying case for input-output dynamical systems, in order to encode a large variety of robotic tasks as *extended set-based* tasks. Moreover, under mild assumptions, we prove the continuity of the robot controller required to execute a variable number of tasks whose priorities change over time. Finally, even though the proposed framework applies to multiple kinds of robotic systems, in this paper we focus on the case of robotic manipulators (see, e.g., [9] for a similar approach used for humanoid robots, or [10] for an example of application to the multi-robot domain).

II. LITERATURE REVIEW

Specifying the behavior of robots in the task space is generally simpler and more intuitive than doing it in the input space. Consequently, building a controller in the task space and mapping the control input into the joint space using the (pseudo-)inverse of the Jacobian is very convenient (see e.g., [11], [12]). This kind of approach has been extended for dealing with humanoids [13] and multi-robot systems [14]. The implementation of the main task usually involves fewer degree-of-freedom (DoF) than the ones available. Thus, it is possible to exploit the so-called *task redundancy* of the system in order to implement extra, secondary, tasks. This can be done by arranging the secondary tasks into a hierarchical stack of tasks and to exploit the Jacobian null space projection for finding the joint velocities that implement the secondary tasks without interfering with the main task, as shown in [12], [14].

When the hierarchy of tasks to be executed is dynamic, it is important to allow a smooth, stable and efficient tran-

[1]G. Notomista, S. Mayya, and M. Santos are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, Georgia, USA {g.notomista, siddharth.mayya, maria.santos}@gatech.edu

[2]M. Selvaggio is with the Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy mario.selvaggio@unina.it

[3]C. Secchi is with the Department of Science and Methods of Engineering, University of Modena and Reggio Emilia, Modena, Italy cristian.secchi@unimore.it

sition strategy between the tasks to be implemented and, consequently, on the controls provided to the robot. A lot of work has been done in order to include this feature in the operational space approach. As shown in [4], for instance, task swapping leads to a discontinuity in the control action. Consequently, effective strategies for smoothing the control action during task swapping have been proposed in [4], [15], but their reactivity is limited. As shown in [16] and [17], it is possible to achieve control continuity by scaling or modifying the tasks to be achieved. In [18], [19] and [20], modified projectors are introduced for implementing a soft prioritization that allows to smoothly rearrange the, possibly relaxed, tasks.

In order to handle dynamic task priorities, in [21] a prioritized task regulation framework based on a sequence of quadratic programs has been proposed. Hierarchical quadratic programming has also been exploited in [22], [23]. The main advantage of recasting the task regulation into a sequence or quadratic programs is the higher freedom in specifying the tasks with respect to the operational framework. Moreover, the approach affords swapping the tasks in real time at the cost of solving a number of increasingly complex quadratic programs, which can be challenging for real-time execution. Jacobian-based task execution and prioritization, as well as the hierarchical quadratic programming approach, shares multiple features with the method presented in this paper. In the next section, we show how Jacobian-based tasks can be encoded using our notion of extended set-based tasks. Moreover, the framework we propose encodes tasks using constraints of an optimization-based controller—just like in the hierarchical quadratic programming case—but with the advantage of solving a single optimization problem. This is achieved by leveraging control barrier functions, originally formulated to ensure a safety property through the selection of a proper control input that can be found by solving a simple convex optimization problem [6]. Control barrier functions have been successfully exploited for controlling the behavior of robots in multiple domains (see, e.g., [24], [25], [26], [27], [28]), and they naturally allow the execution of multiple tasks [29], [30].

III. TASK EXECUTION AND PRIORITIZATION

The objective of this section is to develop the proposed task-execution and task-prioritization framework and to illustrate its ability to synthesise continuous-controllers even when task priorities are time-varying.

Although the proposed formulation can be applied to other types of robotic systems (see, e.g., [10] or [31]), in this paper, we will focus on n-DoF robotic manipulators. For these types of robots, the so-called *Jacobian-based* tasks are generally described by the following expression:

$$\dot{\sigma} = J(q)\dot{q},$$

where $q \in \mathbb{R}^n$ is the vector of generalized coordinates representing the state of the robot, $\sigma(q) \in \mathbb{R}^m$ is a function of the robot state, and $J(q) \in \mathbb{R}^{m\times n}$ is its Jacobian. In its basic meaning, a Jacobian-based task is said to be accomplished when the value of $\sigma(q)$ is regulated to a desired value $\sigma_0(q)$.

A. Extended Set-Based Tasks

Set-based tasks have been introduced in [22] as tasks characterized by a desired *area of satisfaction*, which corresponds to the task being executed. Typical tasks which are also examples of set-based tasks consist in avoidance of joint limits or obstacles, in which the areas of satisfaction are represented by the intervals of allowed joint angles and the collision-free space, respectively. In this definition of tasks, we notice an analogy with the concept of *forward invariance* (also referred to as *safety*) in the dynamical system literature [32]. A set-based task can be seen as keeping a desired set forward invariant, i.e., ensuring that the state of the robot never leaves the set. In the following definition, we propose an extension of this concept of set-based tasks in order to include the possibility of executing a task by going towards a set as well—which, in the dynamical system literature, corresponds to the well-known concept of set *stability*.

Definition 1 (Extended Set-Based Task)**.** *An extended set-based task is a task characterized by a set $\mathcal{C} \subset \mathcal{T}$, where $\mathcal{T}$ is the task space, which can be expressed as the zero superlevel set of a continuously differentiable function $h\colon \mathcal{T}\times\mathbb{R}_{\geq 0} \to \mathbb{R}$ as follows:*

$$\mathcal{C} = \{\sigma \in \mathcal{T}\colon h(\sigma, t) \geq 0\}. \tag{1}$$

The goal is rendering the time-varying set forward invariant and asymptotically stable.

The following example ties the concept of extended set-based tasks to the one of Jacobian-based tasks.

Example 2 (Generalization of Jacobian-based tasks—Part I)**.** *Consider the Jacobian-based task encoded by the following differential kinematic equation:*

$$\dot{\sigma}_0 = J_0(q)\dot{q}, \tag{2}$$

where $\dot{\sigma}_0$ is a desired function we want the robot to track. Let the function h used in Definition 1 be defined as

$$h(\sigma_0, t) = -\frac{1}{2}\|\sigma - \sigma_0(t)\|^2, \tag{3}$$

where the function $\sigma_0(t)$ is given by $\sigma_0(t) = \int_0^t \dot{\sigma}_0(\tau)\mathrm{d}\tau$. Assuming that the $\dot{\sigma}_0$ specified in (2) *is a continuous function of time, h is continuously differentiable with respect to both its arguments, as requested by the Definition 1.*

We notice that the set $\mathcal{C}$, zero superlevel of h, is given by $\{\sigma \in \mathcal{T}\colon \sigma = \sigma_0(t)\}$. Therefore, rendering $\mathcal{C}$ asymptotically stable and forward invariant corresponds to the original Jacobian-based task being accomplished. In Example 5 in Section III-B, a method is proposed for the execution of this task using control barrier functions.

B. Extended Set-Based Task Execution

In this section, we propose a framework to execute the extended set-based tasks defined above which will ultimately allow for time-varying prioritization among tasks in Section

III-C. Throughout this paper, we will suppose we can model the robot with a control affine dynamical system. Moreover, we will assume we have access to an output variable, $\sigma \in \mathcal{T}$, which represents the task variable. Therefore, the model of the robot is given by

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ \sigma = k(x), \end{cases} \tag{4}$$

where $x \in \mathcal{X} \subseteq \mathbb{R}^n$, $u \in \mathcal{U} \subseteq \mathbb{R}^p$, $\sigma \in \mathcal{T}$, f and g are Lipschitz-continuous vector fields, and $k\colon \mathbb{R}^n \to \mathcal{T}$ is a smooth map representing the task forward kinematics.

The model (4) encompasses both dynamic and kinematic models of robotic manipulators. In case the nonlinear second-order dynamic model of a robot is considered (see, e.g., [1]), by defining the joint angles and velocities as the state $x = [x_1, x_2]^{\mathrm{T}} = [q, \dot{q}]^{\mathrm{T}}$ and the joint torques as the input u, the system can be brought to control affine form. If, instead, a velocity-resolved robot control scheme is considered [1], then the system follows the single-integrator dynamics $\dot{x} = u$—which is control affine too—, where the state x is the vector of joint coordinates and the input u is the vector of joint velocities. Without loss of generality, in the following we will focus on the kinematic model of robotic manipulator, assuming that we can control its joint velocities.

Based on a previously developed method for executing tasks [31], we now present an optimization-based framework required to execute extended set-based tasks. As described in Section I, we consider tasks whose execution can be encoded as the minimization of a continuously-differentiable cost function $C\colon \mathcal{T} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$. This can be written as the following optimization problem:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ C(\sigma, t) \\ &\text{subject to } \dot{x} = f(x) + g(x)u \\ &\qquad\qquad \sigma = k(x). \end{aligned} \tag{5}$$

Following the approach proposed in [31] and [10], this problem can be reformulated as a constraint-based optimization problem where the robot minimizes its control input subject to a constraint which enforces the execution of the task itself. These constraints are enforced using Control Barrier Functions (CBFs) (see [6] for a comprehensive overview on the subject). CBFs perfectly lend themselves to encode extended set-based tasks as they are able to ensure both set safety and set stability. These properties will be recalled in the following for the notion of CBFs extended to encompass constraints on the output of a dynamical system.

Definition 3 (Output Time-Varying Control Barrier Function, based on [6] and [33])**.** *Let $\mathcal{C} \subset \mathcal{D} \subset \mathcal{T}$ be the superlevel set of a continuously differentiable function $h : \mathcal{D} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$, contained in a domain $\mathcal{D}$. Then, h is an output time-varying control barrier function—in the following referred to simply as CBF—if there exists a Lipschitz continuous extended class $\mathcal{K}_\infty$ function ([6]) γ such that for the control system* (4)*, for all $\sigma \in \mathcal{D}$,*

$$\sup_{u \in \mathcal{U}} \left\{ \frac{\partial h}{\partial t} + \frac{\partial h}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \frac{\partial h}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u \right\} \geq -\gamma(h(\sigma)). \tag{6}$$

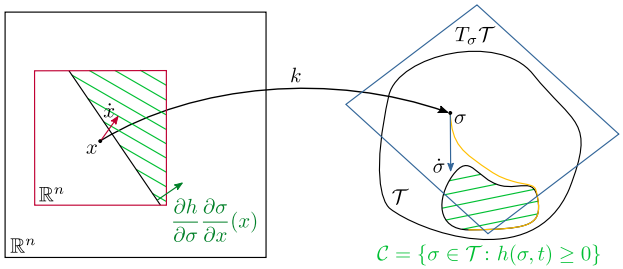

Fig. 1. Pictorial representation of the differential constraints introduced in (6), for a time-invariant CBF. The set $\mathcal{C}$ to be rendered forward invariant and asymptotically stable is a subset of the task space $\mathcal{T}$. The constraint (6) can be equivalently written as $\frac{\partial h}{\partial \sigma}\dot{\sigma} \geq -\gamma(h(\sigma, t))$, which is a affine in $\dot{\sigma}$, constraining it to lie in a half plane in the tangent space $T_\sigma\mathcal{T}$ of the task space. The velocity $\dot{\sigma}$ is transformed into the velocity $\dot{x}$ in the tangent space $T_x\mathbb{R}^n \cong \mathbb{R}^n$ of the state space. In this space, the constraint (6) is affine in $\dot{x}$, which has to lie in the green hatched half plane.

With this definition of CBF, we can now state the main theorem whose results will be used in the following sections to derive the multi-task execution and prioritization framework.

Theorem 4 (based on [6] and [33])**.** *Let $\mathcal{C} \subset \mathcal{T}$ be a set defined as the superlevel set of a continuously differentiable function $h : \mathcal{D} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$. If h is a CBF on $\mathcal{D}$ according to Definition 3, then any Lipschitz continuous controller $u(x, t) \in \mathcal{V}(x, t)$, where $\mathcal{V}(x, t) = \{u(x, t) \colon \frac{\partial h}{\partial t} + \frac{\partial h}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \frac{\partial h}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u + \gamma(h(\sigma)) \geq 0\}$, for the system* (4)*, renders the set $\mathcal{C}$ forward invariant. Additionally, the set $\mathcal{C}$ is asymptotically stable in $\mathcal{D}$.*

In the case of robotic manipulators, Definition 3 and Theorem 4 describe how CBFs ensure the forward invariance and the asymptotic stability of a subset of the task space $\mathcal{T}$ by enforcing constraints on the joint velocities in the tangent space of the state space. This is depicted in Fig. 1, where the set to be rendered asymptotically stable is marked by a green hatching. The velocity $\dot{\sigma}$ is transformed to the velocity $\dot{x}$ by means of the Jacobian $J(q)$ of the transformation k. Then, the inequality in (6) corresponds to constraining the vector $\dot{x}$ to lie in the half plane of $\mathbb{R}^n$ hatched in green.

Proceeding similarly to [31], it can be shown how a control input $u(t)$ generated using (5) is equivalent to solving the following constrained optimization problem:

$$\begin{aligned} &\underset{u}{\text{minimize}}\ \|u\|^2 \\ &\text{subject to } \frac{\partial h}{\partial t} + \frac{\partial h}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \frac{\partial h}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u \\ &\qquad\qquad + \gamma(h(\sigma, t)) \geq 0, \end{aligned} \tag{7}$$

where $h(\sigma, t) = -C(\sigma, t)$ is a CBF. See [31] for a detailed discussion on this equivalence.

Example 5 (Generalization of Jacobian-based tasks—Part II)**.** *The execution of the task introduced in Example 2 can be done by ensuring the asymptotic stability of the set $\mathcal{C}$ defined in* (1)*. This property, in turn, can be achieved by treating h as a CBF. As a result, by the definition of the set $\mathcal{C}$ in* (1) *and the function h in* (3)*, ensuring the asymptotic stability of the set $\mathcal{C}$ corresponds to the following condition: $\sigma(t) \to \sigma_0(t)$, as $t \to \infty$.*

We can now use the optimization-based formulation (7)*, in order to synthesize a velocity controller $\dot{q}$ that is able to render the set $\mathcal{C}$ asymptotically stable as well as forward invariant. The controller solution of*

$$\begin{aligned} &\underset{\dot{q}}{\text{minimize}} \ \|\dot{q}\|^2 \\ &\text{subject to } (\sigma - \sigma_0(t))^{\mathrm{T}} J_0(q)\dot{q} \geq \\ &\qquad -\gamma(h(\sigma_0, t)) - (\sigma - \sigma_0(t))^{\mathrm{T}} \dot{\sigma}_0(t) \end{aligned} \tag{8}$$

guarantees the asymptotic stability of $\mathcal{C}$, which is equivalent to the execution of the Jacobian-based task (2)*. Moreover, owing to its convexity, solving the optimization problem* (8) *can be done very efficiently [34]. Therefore, this approach also lends itself to applications where real-time requirements are prescribed.*

In the next section, we demonstrate how the formulation in (7) can be extended to allow executing and prioritizing multiple tasks.

C. Extended Set-Based Multi-Task Prioritization

Consider a set of tasks, denoted as $T_1, \ldots, T_M$, that need to be executed and are encoded by cost functions $C_1, \ldots, C_M$, respectively. The execution of these tasks can be achieved by solving:

$$\begin{aligned} &\underset{u}{\text{minimize}} \ \|u\|^2 \\ &\text{subject to } \frac{\partial h_m}{\partial t} + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u \\ &\qquad + \gamma(h_m(\sigma, t)) \geq 0 \quad \forall m \in \{1, \ldots, M\}, \end{aligned} \tag{9}$$

where $h_m(\sigma, t) = -C_m(\sigma, t)$. While this formulation provides a convenient way to compose multiple tasks together, the solution to such an optimization problem is not guaranteed to exist due to the incompatibility of conflicting task requirements. We now modify this formulation and introduce a slack variable δ_m corresponding to each task, which denotes the extent to which the constraint corresponding to task T_m can be relaxed. Thus, for the set of M tasks $T_1, \ldots, T_M$, (9) is now modified as:

$$\begin{aligned} &\underset{u,\delta}{\text{minimize}} \ \|u\|^2 + l\|\delta\|^2 \\ &\text{subject to } \frac{\partial h_m}{\partial t} + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u \\ &\qquad + \gamma(h_m(\sigma, t)) \geq -\delta_m \quad \forall m \in \{1, \ldots, M\}, \end{aligned} \tag{10}$$

where $\delta = [\delta_1, \ldots, \delta_M]^T$ denotes the slack variables corresponding to each task executed by the robot, and $l \geq 0$ is a scaling constant. Within this framework, a natural way to introduce priorities is to enforce relative constraints among the slack variables corresponding to the different tasks that need to be performed. For example, if the robot were to perform task T_m with the highest priority (often referred to using the partial order notation $T_m \prec T_n \ \forall n \in \{1, \ldots, M\}, n \neq m$), the additional constraint $\delta_m \leq \delta_n/\kappa$, $\kappa > 1$, $\forall n \in \{1, \ldots, M\}, n \neq m$ in (10) would imply that task T_m is relaxed to a lesser extent—thus performed with a higher priority—than the other tasks. The relative scale between the functions h_m encoding the tasks should be considered while choosing the value of κ. In general, such prioritizations among tasks can be encoded through an additional linear constraint $K\delta \geq 0$ to be enforced in the optimization problem (10). In the following, we refer to K as the *prioritization matrix*, which encodes the pairwise inequality constraints among the slack variables, thus fully specifying the prioritization stack among the tasks.

D. Examples of Applications

We now consider three examples in order to highlight how the proposed framework can effectively enable the prioritization and execution of multiple safety- as well as non-safety-critical tasks. Each scenario presented below considers three tasks, some of which are safety-critical—i.e., they always need to be executed, regardless of the prioritization stack. The three examples share the formulation in (10) with the additional constraint $K\delta \geq 0$.

1) 3 Non-Safety-Critical Prioritized Tasks: The 3 non-safety-critical tasks are encoded by the CBFs

$$h_m : \mathcal{T}_m \times \mathbb{R}_{\geq 0} \to \mathbb{R}, \qquad m \in \{1, 2, 3\}, \tag{11}$$

where $\mathcal{T}_m$ is the task space related to task m. The values of the entries of the prioritization matrix K have to be chosen in order to encode the desired priorities between the tasks. For instance, if the stack of tasks is $T_1 \prec T_3 \prec T_2$, then the rows of K have to encode the constraints $\delta_1 \leq \delta_3/\kappa$ and $\delta_3 \leq \delta_2/\kappa$ (as described in Sect. III-C), i.e.,

$$K = \begin{bmatrix} -1 & 0 & 1/\kappa \\ 0 & 1/\kappa & -1 \end{bmatrix}.$$

2) 1 Safety-Critical Task and 2 Non-Safety-Critical Prioritized Tasks: Assume task T_1 is safety-critical and tasks T_2 and T_3 are not, where each T_m is encoded as in (11). In this case, $\delta_1 \overset{!}{=} 0$ and the prioritization matrix K is given by

$$K(t) = \begin{cases} \begin{bmatrix} 0 & -1 & 1/\kappa \end{bmatrix} & \text{if } T_2 \prec T_3, \\ \begin{bmatrix} 0 & 1/\kappa & -1 \end{bmatrix} & \text{if } T_2 \succ T_3. \end{cases}$$

Notice that, as task T_1 is safety-critical, it will always be executed as its corresponding slack variable is set to 0, whereas tasks T_2 and T_3 will be executed only if their slack variables can be driven to 0. This condition is analogous to the notions of task *independence* and *orthogonality* ([35]).

3) 2 Safety-Critical Tasks and 1 Non-Safety-Critical Task: Now, assume that tasks T_1 and T_2 are safety-critical and task T_3 is not. In this case, there is no need for prioritizing. In fact, by definition of safety-critical tasks, T_1 and T_2 must be

always executed ($\delta_1 = \delta_2 \overset{!}{=} 0$). Task T_3 will be executed only if it is compatible with the execution of the first two (see discussion in Sect. III-D.2). Note further that, since the safety-critical tasks T_1 and T_2 do not have to be prioritized, they could even be combined in a single task using the techniques developed in [29] or [30].

These examples demonstrate how our framework can encode the prioritization of different task stacks, especially when multiple tasks might be safety-critical. As the objective of this paper is to illustrate how such a formulation can also allow for dynamically evolving task prioritizations, the next section considers a time-varying prioritization matrix $K(t)$, and demonstrates how this lends itself to the synthesis of continuous controllers for task switching and task insertion/removal.

E. Time-Varying Priorities

As the need of switching priorities between tasks in an online fashion arises in multiple applications (see discussions in Section I and II), in this section, the application of the control framework proposed in this paper to the case of dynamic priorities is shown. The main problem when priorities are switched is that, due to the intrinsically discrete nature of the order among tasks, discontinuities in the controller might arise during the switching transient [4]. As discussed in Section II, several methods have been proposed to mitigate this effect, which are tailored to the specific task execution or prioritization approach they target.

In the following proposition, we give sufficient conditions to ensure the continuity of the robot control input during the priority reordering of a stack of extended set-based tasks executed using the optimization-based control framework presented in this paper.

Proposition 6 (Continuity of the controller during task priority switching)**.** *Consider the following optimization problem:*

$$\begin{aligned}
&[u^*(t), \delta^*(t)] = \\
&\underset{u,\delta}{\arg\min} \quad \|u\|^2 + l\|\delta\|^2 \\
&\text{subject to } \frac{\partial h_m}{\partial t} + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u \\
&\qquad + \gamma(h_m(\sigma, t)) \geq -\delta_m \quad \forall m \in \{1, \dots, M\} \\
&\qquad K(t)\delta \geq 0,
\end{aligned} \tag{12}$$

where $l > 0$, $h_m = -C_m$, $m = 1, \dots, M$ encode M tasks through the continuously differentiable cost functions C_m, and $K \colon \mathbb{R}_{\geq 0} \to \mathbb{R}^{N_p \times M}$ is a mapping that, for each time instant t, provides a $N_p \times M$ prioritization matrix, N_p being the number of constraints required to characterize the desired task prioritizations[1]*. If K is Lipschitz continuous in time, then the controller $u^*(t)$, solution of* (12)*, is Lipschitz continuous in time.*

Proof. As discussed in Section III-B and in [10], thanks to the presence of the slack variables δ_m and the absence of additional constraints on u, the optimization problem (12) is always feasible. Moreover, by the assumption of $l > 0$, the objective function is strictly convex. Furthermore, by assumption, all constraints in (12) are Lipschitz continuous with respect to time. Then, Theorem 1 in [36] ensures that the solution of (12) is Lipschitz continuous. □

[1]Notice that $N_p \leq M^2$ as any other pairwise constraint would be redundant.

A limiting case of priority switching consists in the insertion or the removal of a task from a given stack of tasks. Using the optimization-based control framework proposed in this paper, the following proposition gives sufficient conditions required in order to perform such a maneuver while ensuring the continuity of the velocity controller of the robot.

Proposition 7 (Continuity of the controller during task insertion)**.** *Consider the following optimization problem:*

$$\begin{aligned}
&[u^*(t), \delta^*(t)] = \\
&\underset{u,\delta}{\arg\min} \quad \|u\|^2 + l\|\delta\|^2 \\
&\text{subject to } \frac{\partial h_m}{\partial t} + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \frac{\partial h_m}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u \\
&\qquad + \gamma(h_m(\sigma, t)) \geq -\delta_m \quad \forall m \in \{1, \dots, M-1\} \\
&\qquad \frac{\partial h_M}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u + \delta_M \\
&\qquad \geq \left(-\frac{\partial h_M}{\partial t} - \frac{\partial h_M}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \gamma(h_M(\sigma, t)) \right) \rho(t) \\
&\qquad K(t)\delta \geq 0,
\end{aligned} \tag{13}$$

where $l \geq 0$, $\rho \colon \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$, and $K \colon \mathbb{R}_{\geq 0} \to \mathbb{R}^{N_p \times M}$ is a mapping defined as in Proposition 6.

Let $t_{ins} \geq 0$ be the time at which task T_M needs to be inserted. If K and ρ are continuous functions, with $\rho \equiv 0$ on $(-\infty, t_{ins}]$ and $\rho \equiv 1$ on $[t_{ins} + \Delta t_{ins}, \infty)$, then the controller u^*, solution of (13)*, is a continuous function of time and allows task T_M to be inserted at time $t_{ins} + \Delta t_{ins}$ in the stack of tasks with any desired priority.*

Proof. For $t \leq t_{\text{ins}}$, as $\rho(t) \equiv 0$, the constraint

$$\begin{aligned}
&\frac{\partial h_M}{\partial \sigma}\frac{\partial \sigma}{\partial x} g(x)u + \delta_M \\
&\geq \left(-\frac{\partial h_M}{\partial t} - \frac{\partial h_M}{\partial \sigma}\frac{\partial \sigma}{\partial x} f(x) + \gamma(h_M(\sigma, t)) \right) \rho(t)
\end{aligned} \tag{14}$$

related to task T_M is not active. In fact, $(u, \delta) = (0, 0)$ satisfies (14) and would achieve a (global) minimum of the cost function. For $t > t_{\text{ins}}$, the constraint (14) is effectively inserted in the optimization problem. As, by hypotheses, $\rho(t)$ is continuous and $\rho(t) = 0$ for $t \leq t_{\text{ins}}$, $\rho(t) \to 0$ as $t \to t_{\text{ins}}$. Therefore, the right hand side of (14) also converges to 0 as $t \to t_{\text{ins}}$. Thus, by Corollary 3.1 in [37], the solution $u^*(t)$ is continuous at $t = t_{\text{ins}}$. Continuity for all $t > t_{\text{ins}}$ stems from the continuity of ρ and of all the other parameters of the optimization problem. For $t \geq t_{\text{ins}} + \Delta t_{\text{ins}}$, the condition $\rho \equiv 1$ allows task T_M to be executed together with the other tasks with priorities encoded by the prioritization matrix $K(t)$. Hence, the controller $u^*(t)$, solution of (13), is a continuous velocity controller for the robot, that is able

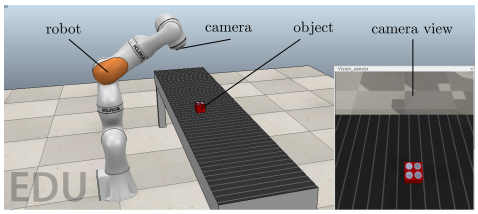

Fig. 2. Experimental setup: a 7-DoF manipulator performing multiple tasks in an simulated industrial environment [38]. A camera mounted on the end-effector is used to perform visual servoing tasks.

to insert task T_M into the stack of tasks with any desired priority. □

Remark 8. *A result analogous to Proposition 13 can be stated to show the continuity of the robot controller when a task is removed from the stack of prioritized tasks, rather than inserted. If task T_M is to be removed at the time t_{rem}, $\rho(t)$, besides being continuous, has to go to 0 as $t \to t_{rem}$.*

In the next section, we present the results of a simulated experiment performed using a manipulator arm. The experiment includes insertion of safety- as well as non-safety-critical tasks in a time-varying prioritized stack of tasks.

IV. EXPERIMENTS

The experimental setup is shown in Fig. 2. A 7-DoF arm is employed to perform joint control, position control of the robot end-effector, as well as visual servoing tasks. These include both safety- and non-safety-critical tasks, as described in detail in the following. Therefore, the considered task spaces are the joint space, the Cartesian space, and the projective plane.

To this end, let $q \in \mathbb{R}^7$ denote the vector of the robot generalized coordinates, $p \in \mathbb{R}^3$ the end-effector Cartesian position, and $s \in \mathbb{R}^2$ the coordinates of a feature on the image plane. Following the formulation in Example 2, these tasks can be encoded through the following CBFs: $h_T(z) = -\gamma_T \|z - z_d\|^2$. In the expression of $h_T(z)$, $\gamma_T > 0$, and z is a placeholder for q in the joint space task (T_q), p in the Cartesian space task (T_p), and s in the visual servoing task (T_s). The value of z_d denotes the desired reference value for z. Additionally, one safety-critical task is considered, namely keeping the joint variables within an upper and a lower bound. The corresponding CBF used in this case for each joint i is given by

$$h_{T,i}(z_i) = \gamma_T \left(z_i^+ - z_i\right)\left(z_i - z_i^-\right),$$

where z_i^+ and z_i^- denote the i-th upper and lower limit, respectively, of the variable z_i. In the joint space, these bounds represent physical joint limits. Notice how keeping the value of $z_i \in [z_i^-, z_i^+]$ corresponds to $h_T(z) \geq 0$, which can be enforced using the constraint-based formulation in (13).

The results of a pilot experiment are shown in Fig. 3. At the top, the values of the CBFs corresponding to joint control (h_q), position control (h_p) and visual servoing (h_s) tasks are reported. The objective is that of driving their values to 0

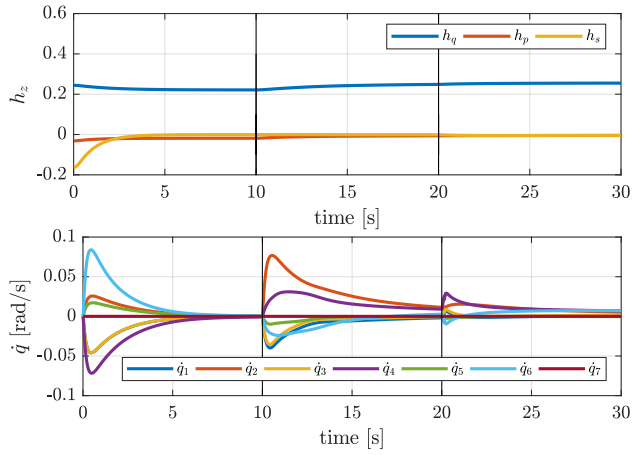

Fig. 3. Graphs of the values of the CBFs of the 3 considered tasks (top), and of the joint velocity controller (bottom). The vertical lines at $t = 10$ s correspond to insertion of the position task, whereas the ones at $t = 20$ s indicate the switch of priorities between the position control and the visual servoing tasks. In the latter case, it can be observed how the presented task prioritization framework is suitable to switch task priorities even when tasks have not been completely accomplished yet.

and/or keep them non-negative. As can be seen, the value of h_q is always kept positive, corresponding to the condition in which joint variables do not go beyond their limits. The values of h_p and h_s, instead, are driven to 0 in order to execute the prioritized tasks. At the bottom of Fig. 3, the joint velocity controller is plotted, showing the continuity property guaranteed by the optimization problem (13). The prioritization of the tasks during the course of the experiment is as follows. The safety-critical task T_q always has highest priority. At $t = 0$ s, T_v is inserted: at this point, the stack of tasks is $T_q \prec T_v$. Then, at $t = 10$ s, T_p is inserted at the lowest priority level, so that the stack of tasks becomes $T_q \prec T_v \prec T_p$. Finally, at $t = 20$ s, T_v and T_p are swapped, making the stack of tasks $T_q \prec T_p \prec T_v$.

Thus, the extended-based task prioritization framework presented in this paper guarantees the continuity of the robot velocity controller when applied to task insertion and priority switching scenarios. If the parameters of the optimization problem (13) (i.e., ρ and K) vary smoothly, it is possible to guarantee the smoothness of the velocity controller. This can be used to ensure continuity of the torque controls.

V. CONCLUSIONS

In this paper, we presented an optimization-based framework to execute and prioritize multiple robotic tasks. We extended the definition of set-based tasks and proposed a method to execute them, which leverages control barrier functions. Time-varying priorities, as well as task insertion and removal, were handled by continuously changing the optimization problem required to synthesize the robot controller. Guarantees on continuity were provided under mild assumptions on the parameters of the optimization problem. Although the presented framework can be applied to robotic systems of different nature, its effectiveness was demonstrated through a simulated experiment involving a manipulator arm robot.

REFERENCES

[1] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[2] P. Di Lillo, F. Arrichiello, G. Antonelli, and S. Chiaverini, "Safety-related tasks within the set-based task-priority inverse kinematics framework," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 6130–6135.

[3] M. Selvaggio, P. Robuffo Giordano, F. Ficuciello, and B. Siciliano, "Passive task-prioritized shared-control teleoperation with haptic guidance," in *2019 International Conference on Robotics and Automation*, May 2019, pp. 430–436.

[4] F. Keith, P. B. Wieber, N. Mansard, and A. Kheddar, "Analysis of the discontinuities in prioritized tasks-space control under discrete task scheduling operations," *IEEE International Conference on Intelligent Robots and Systems*, pp. 3887–3892, 2011.

[5] S. Moe, G. Antonelli, A. R. Teel, K. Y. Pettersen, and J. Schrimpf, "Set-based tasks within the singularity-robust multiple task-priority inverse kinematics framework: General formulation, stability analysis, and experimental results," *Frontiers in Robotics and AI*, vol. 3, p. 16, 2016.

[6] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, June 2019, pp. 3420–3431.

[7] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 6271–6278.

[8] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015.

[9] A. D. Ames and M. Powell, "Towards the unification of locomotion and manipulation through control lyapunov functions and quadratic programs," in *Control of Cyber-Physical Systems*. Springer, 2013, pp. 219–240.

[10] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," in *2019 18th European Control Conference (ECC)*, June 2019, pp. 2071–2076.

[11] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," *IEEE Journal on Robotics and Automation*, vol. 3, no. 1, pp. 43–53, February 1987.

[12] B. Siciliano and J. . E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Fifth International Conference on Advanced Robotics*, June 1991, pp. 1211–1216 vol.2.

[13] N. Mansard, A. Remazeilles, and F. Chaumette, "Continuity of varying-feature-set control laws," *IEEE Transactions on Automatic Control*, vol. 54, no. 11, pp. 2493–2505, Nov 2009.

[14] G. Antonelli, F. Arrichiello, and S. Chiaverini, "Experiments of formation control with multirobot systems using the null-space-based behavioral control," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1173–1182, Sep. 2009.

[15] G. Jarqun, A. Escande, G. Arechavaleta, T. Moulard, E. Yoshida, and V. Parra-Vega, "Real-time smooth task transitions for hierarchical inverse kinematics," in *2013 13th IEEE-RAS International Conference on Humanoid Robots*, Oct 2013, pp. 528–533.

[16] J. Lee, N. Mansard, and J. Park, "Intermediate desired value approach for task transition of robots in kinematic control," *IEEE Transactions on Robotics*, vol. 28, no. 6, pp. 1260–1277, 2012.

[17] F. Flacco, A. De Luca, and O. Khatib, "Control of Redundant Robots under Hard Joint Constraints: Saturation in the Null Space," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 637–654, 2015.

[18] M. Liu, S. Hak, and V. Padois, "Generalized projector for task priority transitions during hierarchical control," *IEEE International Conference on Robotics and Automation*, no. June, pp. 768–773, 2015.

[19] N. Dehio, D. Kubus, and J. J. Steil, "Continuously shaping projections and operational space tasks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2018, pp. 5995–6002.

[20] N. Dehio and J. J. Steil, "Dynamically-consistent Generalized Hierarchical Control," *IEEE International Conference on Robotics and Automation*, pp. 1141–1147, 2019.

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

[22] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, vol. 33, no. 7, pp. 1006–1028, 2014.

[23] S. Kim, K. Jang, S. Park, Y. Lee, S. Y. Lee, and J. Park, "Continuous task transition approach for robot controller based on hierarchical quadratic programming," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1603–1610, 2019.

[24] G. Notomista, S. F. Ruf, and M. Egerstedt, "Persistification of robotic tasks using control barrier functions," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 758–763, April 2018.

[25] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, no. 3, pp. 661–674, June 2017.

[26] L. Guerrero-Bonilla and V. Kumar, "Realization of r -robust formations in the plane using control barrier functions," *IEEE Control Systems Letters*, vol. 4, no. 2, pp. 343–348, April 2020.

[27] L. Lindemann and D. V. Dimarogonas, "Control barrier functions for multi-agent systems under conflicting local signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 3, pp. 757–762, July 2019.

[28] C. Talignani Landi, F. Ferraguti, S. Costi, M. Bonf, and C. Secchi, "Safety barrier functions for human-robot interaction with industrial manipulators," in *2019 18th European Control Conference*, June 2019, pp. 2565–2570.

[29] L. Wang, A. D. Ames, and M. Egerstedt, "Multi-objective compositions for collision-free connectivity maintenance in teams of mobile robots," in *2016 IEEE 55th Conference on Decision and Control*, 2016, pp. 2659–2664.

[30] P. Glotfelter, J. Corts, and M. Egerstedt, "Boolean composability of constraints and control synthesis for multi-robot systems via nonsmooth control barrier functions," in *2018 IEEE Conference on Control Technology and Applications*, Aug 2018, pp. 897–902.

[31] G. Notomista and M. Egerstedt, "Constraint-driven coordinated control of multi-robot systems," in *2019 American Control Conference (ACC)*, July 2019, pp. 1990–1996.

[32] H. K. Khalil, *Nonlinear control*. Pearson New York, 2015.

[33] G. Notomista and M. Egerstedt, "Persistification of robotic tasks," *arXiv preprint arXiv:1903.05810*, 2019.

[34] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[35] G. Antonelli, "Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 985–994, Oct 2009.

[36] B. Morris, M. J. Powell, and A. D. Ames, "Sufficient conditions for the lipschitz continuity of qp-based multi-objective control of humanoid robots," in *52nd IEEE Conference on Decision and Control*, 2013, pp. 2920–2926.

[37] G. Lee, N. Tam, and N. Yen, "Continuity of the solution map in quadratic programs under linear perturbations," *Journal of Optimization Theory and Applications*, vol. 129, no. 3, pp. 415–423, 2006.

[38] E. Rohmer, S. P. N. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov 2013, pp. 1321–1326.