



HAL
open science

A Stable Method for Task Priority Adaptation in Quadratic Programming via Reinforcement Learning

Andrea Testa, Marco Laghi, Edoardo del Bianco, Gennaro Raiola, Enrico Mingo Hoffman, Arash Ajoudani

► **To cite this version:**

Andrea Testa, Marco Laghi, Edoardo del Bianco, Gennaro Raiola, Enrico Mingo Hoffman, et al.. A Stable Method for Task Priority Adaptation in Quadratic Programming via Reinforcement Learning. 2023. hal-04280264v1

HAL Id: hal-04280264

<https://hal.science/hal-04280264v1>

Preprint submitted on 10 Nov 2023 (v1), last revised 29 Feb 2024 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Stable Method for Task Priority Adaptation in Quadratic Programming via Reinforcement Learning

Andrea Testa^{1,2,3}, Marco Laghi^{1,2}, Edoardo Del Bianco^{1,2,3}, Gennaro Raiola^{1,2},
Enrico Mingo Hoffman^{2,4}, and Arash Ajoudani^{1,2}

Abstract—In emerging manufacturing facilities, robots must enhance their flexibility. They are expected to perform complex jobs, showing different behaviors on the need, all within unstructured environments, and without requiring reprogramming or setup adjustments. To address this challenge, we introduce the A3CQP, a non-strict hierarchical Quadratic Programming (QP) controller. This controller seamlessly combines both motion and interaction functionalities, with priorities dynamically and autonomously adapted through a Reinforcement Learning-based adaptation module. This module utilizes the Asynchronous Advantage Actor-Critic algorithm (A3C) to ensure rapid convergence and stable training within continuous action and observation spaces. The experimental validation, involving a collaborative peg-in-hole assembly and the polishing of a wooden plate, demonstrates the effectiveness of the proposed solution in terms of its automatic adaptability, responsiveness, and safety.

Index Terms—Optimization and Optimal Control, Reinforcement Learning, Machine Learning for Robot Control.

I. INTRODUCTION

In recent years, the rapid advancements in robotics and automation have opened up new possibilities for enhancing industrial processes and productivity. The Industry 4.0 paradigm envisions a future where intelligent robots safely work alongside humans, seamlessly adapting to diverse tasks and dynamically adjusting their behavior to meet varying demands.

In pursuit of this vision, control architectures enabling the formulation of multiple objectives, constraints, and priorities have been developed, with Quadratic Programming (QP) as an illustrative and common example in robotics. The arrangement of the task priorities, in particular, augments robot functionality with inherent flexibility, also influencing various indirect aspects, such as human safety [1]–[3]. Nevertheless, in robotics literature, the reorganization of task importance in QP is achieved either through direct human intervention by reprogramming or through pre-established passive rules for clearly defined tasks [4], [5]. This results in limited robot autonomy, leading to poor adaptation to the multifaceted phases of intricate and unstructured jobs [6]. Furthermore, while machine learning-based strategies are well-established for controller tuning to enhance performance based on defined

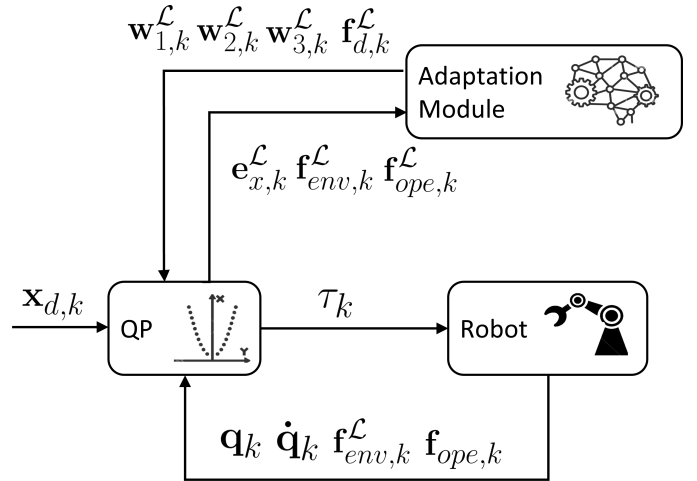


Fig. 1: The proposed A3CQP scheme. The QP controller at the time instant k reads the joint states $\mathbf{q}_k, \dot{\mathbf{q}}_k$ and the external wrenches $\mathbf{f}_{env,k}^L, \mathbf{f}_{ope,k}^L$ from the robot interface and optimizes the joint acceleration $\ddot{\mathbf{q}}_k^*$ and wrench $\mathbf{f}_{d,k}^*$ according to the desired Cartesian pose $\mathbf{x}_{d,k}$. Using these optimal variables, the control torque τ_k is computed. The adaptation module receives the system state $\mathbf{s}_k = \{e_{x,k}^L, \mathbf{f}_{env,k}^L, \mathbf{f}_{ope,k}^L\}$ with respect with the frame \mathcal{L} and returns the adapted weights of the QP cost function $\mathbf{w}_{1,k}^L, \mathbf{w}_{2,k}^L, \mathbf{w}_{3,k}^L$, and the desired contact wrench $\mathbf{f}_{d,k}^L$.

metrics [7]–[10], their potential for dynamically integrating diverse tasks remains largely unexplored.

Based on these premises, we propose the A3CQP controller, a novel approach that harnesses the power of QP to simultaneously handle diverse tasks and constraints while dynamically modulating their weights through an adaptation module employing a Reinforcement Learning (RL) strategy, specifically the Asynchronous Advantage Actor-Critic algorithm (A3C). Our choice for A3C is due to the algorithm’s promising features such as fast convergence, stable training, scalability, and demonstrated robustness [11]. This seamless integration empowers the robot to stably transition between three principal tasks: Cartesian Inverse Kinematic Task, Cartesian Admittance Control Task, and Cartesian Force Control Task. These tasks can capture a broad spectrum of behaviors, ranging from precise position tracking for manipulation to exerting desired forces during physical interactions and following operator guidance for co-manipulation [12]. The automated reasoned task selection within this pool of options enables the robot to autonomously organize itself and adapt to unforeseen situations.

We summarize our contributions as follows:

¹ Leonardo S.p.A., Via Raffaele Pieragostini, 80, 16149 Genova, Italy. E-mail: {name.surname.ext}@leonardo.com

² Istituto Italiano di Tecnologia (IIT), Via San Quirico 19D, 16163 Genova, Italy. E-mail: {name.surname}@iit.it

³ DISI, Università di Trento, Via Sommarive, 9, 38123 Povo, Italy. E-mail: {name.surname}@unitn.it

⁴ Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France. E-mail: enrico.mingo-hoffman@inria.fr

- 1) We introduce a *non-strict hierarchical* QP formulation where the different tasks are achieved according to the trade-off defined by their weights. This framework is crucial for generating consistent and predictable control inputs adherent to the specified constraints. Furthermore, it is decisive to ensure the demonstrated stability of the system, which plays a vital role in achieving optimal performance during operations and guaranteeing safety in interactions between the robot and the operator.
- 2) We present a RL-based adaptation module to dynamically adjust the weights of tasks in the QP. This module plays an essential role in imparting the robot with the capability to learn from its interactions with the environment, enabling autonomous adaptation of task priorities based on job requirements and environmental conditions.

The A3CQP controller, illustrated in Figure 1, can change the robot control mode between tasks without the need for pauses, reprogramming, or adjustments to the setup. Its versatility empowers the robot to handle multifunctional jobs, contributing to the progress of automation in industries, aligning with the vision of Industry 4.0 and 5.0 and its drive towards intelligent and adaptive manufacturing processes.

II. RELATED WORKS

After the first pioneering work on managing multiple tasks in redundant robots [13], there has been a notable surge of interest in hierarchical control. Today, in this domain, two primary approaches have been identified. The first one includes *strict hierarchies*, characterized by ordered task lists wherein the lower ones cannot interfere with the higher ones [1]. From an algorithmic point of view, this approach requires the projection of low-priority tasks in the null-space of higher-priority tasks. On the other hand, the second group examines *non-strict hierarchies*, where there is not a sharp demarcation between priority levels, and all objectives are concurrently addressed through the application of appropriate weighting [2]. This alternative approach has been suggested as more fitting and adaptable when the requirement is to dynamically adjust priorities during the transition between tasks [4], [6]. Within these frameworks, the utilization of QP optimization has become standard practice, thereby enabling the incorporation of linear constraints in both equality and inequality forms, as well as blending of strict and non-strict hierarchical techniques [14], [15].

Various strategies are employed to equip QP controllers with the capacity to adapt to dynamic environments or altered operational conditions. In [16], the authors propose a comprehensive control framework accommodating diverse parallel control modalities. Each modality addresses a different set of tasks, ordered with strict priorities. To ensure environmental compliance, the priorities remain unaltered, but a supervisor has the ability to manually switch between the control modalities. In [5], a generalized projector is presented, which can handle both strict and non-strict priorities with smooth transitions when task priorities are swapped. The evolution of task priorities over time, as well as the timing

and task transitions, need to be designed manually. In [4], the weights of soft QP tasks are adjusted according to a predefined time schedule to adapt the behavior of a humanoid robot with in-contact and not-in-contact motion phases. While methodologies for automating the selection of static priorities are well-established in the literature [7], [8], [10], progress in achieving real-time updates of these priorities for unstructured tasks has been limited. In [6], time-dependent priorities are optimized offline with respect to a predetermined job. In [17], the authors propose modulating task weights online based on movement variance to manage incompatibilities during job execution. Nevertheless, this method still requires at least one demonstration of the prescribed job, and its generalization has not been examined.

Broadly, the challenge of adjusting an agent's behavior in response to unfamiliar situations is often tackled using RL strategies [18]. Initially developed for discrete problems, these techniques can be extended to the continuous domain by harnessing deep learning. Within the field of RL, Actor-Critic methods stand as an amalgamation of value-based and policy-based approaches [19]. In cases where the Actor learns the action policy and the Critic estimates the value function, the resulting algorithm is known as Advantage Actor-Critic (A2C) [20]. In the context of deep learning-supported RL, it is of primary importance to decorrelate the input data [21]. To achieve such decorrelation, the most effective A2C algorithms employ multiple independent agents, each possessing their distinct weights. These agents interact in parallel with separate instances of the environment, updating the main network asynchronously. This method is known as Asynchronous Advantage Actor-Critic (A3C) [21]. However, further studies show that data decorrelation is tied more to parallel agents than asynchronous updates. Indeed, a synchronous and deterministic implementation of this algorithm, where each agent completes its segment of experience before an update is performed, achieves comparable performance to the original algorithm while providing the added benefit of improved GPU utilization efficiency [22]. In this study, we have utilized the latest implementation of the A3C algorithm as the basis for constructing the agent's decision-making policy. This implementation is employed to dynamically adjust the priorities of the QP tasks in response to evolving circumstances.

III. QP CONTROLLER ARCHITECTURE

This section will present a detailed description of the QP control algorithm responsible for determining the joint motor torque. Within the QP structure, we incorporate *three* distinct primary tasks that the robot controller must effectively interchange to handle multifaceted jobs like physical human-robot collaboration activities. These fundamental tasks, stacked at the same priority level of the QP, are listed below.

- 1) The Cartesian Inverse Kinematic Task (IK) is responsible for driving the manipulator toward the desired targets during unrestricted motion while ensuring safe compliance with the environment.

- 2) The Cartesian Admittance Control Task (AC) is essential for physical human-robot collaboration, allowing the robot to follow the guidance provided by the operator.
- 3) The Cartesian Force Control Task (FC) allows the application of precise desired forces when the robot enters in contact with the target workpieces or work surfaces.

During various phases of the job, such as operator guidance, workpiece interaction, or free motion, these tasks must be seamlessly activated to adjust the behavior of the robot accordingly. Throughout the IK task, it is desirable to prevent the robot from encountering singularities and to maintain a high degree of manipulability as it follows the Cartesian reference trajectory. To achieve this, we introduce an additional task called the Joint-level Control Task (JC) at the same priority level in the QP. The JC task is activated alongside the IK task and is designed to optimize the robot joint configuration within its nullspace. The variables to be optimized at each control step k include the joint accelerations $\ddot{\mathbf{q}}_k \in \mathbb{R}^n$, and the Cartesian wrench exerted by the end effector $\mathbf{f}_k \in \mathbb{R}^6$. In the event of contact, this wrench acts in opposition to the external one. The control problem is formulated as the following QP:

$$\begin{aligned}
 \min_{\ddot{\mathbf{q}}_k, \mathbf{f}_k} & \quad \underbrace{\|\mathbf{J}_k \ddot{\mathbf{q}}_k + \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k - \ddot{\mathbf{x}}_{d,k}\|_{\mathbf{w}_1}}_{\text{IK task}} + \underbrace{\|\mathbf{f}_k - \mathbf{f}_{adm,k}\|_{\mathbf{w}_2}}_{\text{AC task}} \\
 & \quad \underbrace{\|\mathbf{f}_k - \mathbf{f}_{ctc,k}\|_{\mathbf{w}_3}}_{\text{FC task}} + \underbrace{\|\ddot{\mathbf{q}}_k - \ddot{\mathbf{q}}_{d,k}\|_{\epsilon \mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k}}_{\text{JC task}}; \\
 \text{s.t.} & \quad \underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}}_k \leq \bar{\ddot{\mathbf{q}}}; \\
 & \quad \underline{\mathbf{f}} \leq \mathbf{f}_k \leq \bar{\mathbf{f}},
 \end{aligned} \tag{1}$$

with $\mathbf{J}_k, \dot{\mathbf{J}}_k \in \mathbb{R}^{6 \times n}$ the end effector Jacobian and its time derivative, $\dot{\mathbf{q}}_k \in \mathbb{R}^n$ the joint velocities, $\underline{\ddot{\mathbf{q}}}, \bar{\ddot{\mathbf{q}}} \in \mathbb{R}^n$ and $\underline{\mathbf{f}}, \bar{\mathbf{f}} \in \mathbb{R}^6$ represent the lower and upper bounds, respectively, for the optimization variables. $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3 \in \mathbb{R}^{6 \times 6}$ are weight matrices used to indicate the relative importance of the first three tasks. On the other hand, the scalar ϵ determines the relationship between the IK and the JC tasks. By choosing $\epsilon \ll 1$, the IK task is prioritized while the JC task serves as a regularization mechanism. This means that the JC task does not interfere with the ability of the robot to reach the desired target, but it does help prevent it from adopting unfavorable joint configurations near singularities. Regarding the target vectors, $\ddot{\mathbf{q}}_{d,k} \in \mathbb{R}^n$ denotes the reference joint accelerations, $\ddot{\mathbf{x}}_{d,k} \in \mathbb{R}^6$ represents the desired Cartesian end effector acceleration, while $\mathbf{f}_{adm,k}, \mathbf{f}_{ctc,k} \in \mathbb{R}^6$ are the desired admittance and contact wrenches, respectively. The Cartesian variables are expressed with respect to the world reference frame \mathcal{W} . However, in this study, we will consistently define the desired wrenches based on a local reference frame \mathcal{L} centered in \mathcal{W} but aligned with the end effector \mathcal{E} of the robot. This is convenient because the adaptation module, responsible for changing task priorities, adopts the perspective of the end effector. The reference frames introduced are shown in Figure 2. The transformation of wrenches from \mathcal{W} to \mathcal{L} is

$$\mathbf{f}_{adm,k}^{\mathcal{L}} = {}^{\mathcal{W}}D_{\mathcal{L},k}^T \mathbf{f}_{adm,k}; \quad \mathbf{f}_{ctc,k}^{\mathcal{L}} = {}^{\mathcal{W}}D_{\mathcal{L},k}^T \mathbf{f}_{ctc,k}, \tag{2}$$

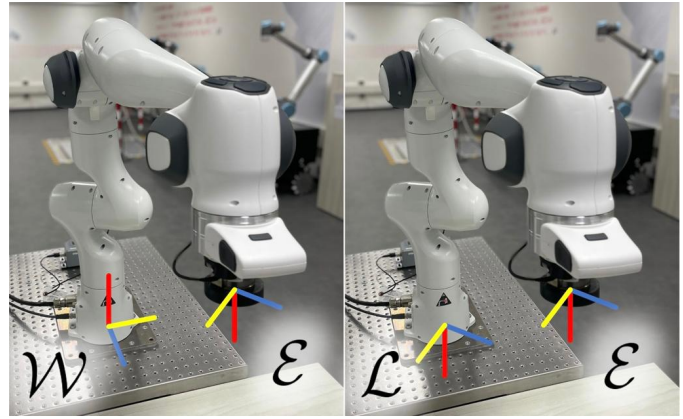


Fig. 2: Representation of the Cartesian reference frames \mathcal{W} , \mathcal{L} , and \mathcal{E} .

with ${}^{\mathcal{W}}D_{\mathcal{L},k} \in \mathbb{R}^{6 \times 6}$ being the Adjoint matrix associated to the homogeneous transformation matrix of \mathcal{L} with respect to \mathcal{W} at control step k . Since \mathcal{L} and \mathcal{W} share the same origin, ${}^{\mathcal{W}}D_{\mathcal{L},k}$ always results in a block-diagonal matrix constructed based on the rotation matrix ${}^{\mathcal{W}}R_{\mathcal{L},k} \in \mathbb{R}^{3 \times 3}$:

$${}^{\mathcal{W}}D_{\mathcal{L},k} = \begin{bmatrix} {}^{\mathcal{W}}R_{\mathcal{L},k} & 0 \\ 0 & {}^{\mathcal{W}}R_{\mathcal{L},k} \end{bmatrix}. \tag{3}$$

In the optimal control problem (1), it is important to highlight that the IK and JC tasks are responsible for computing the optimal acceleration $\ddot{\mathbf{q}}_k^*$, while the optimal wrench \mathbf{f}_k^* is obtained from the AC and FC tasks. These computed values determine the desired joint motor torque at time instant k , established by the following equation, derived by the computed-torque control paradigm [23]:

$$\begin{aligned}
 \boldsymbol{\tau}_k &= \mathbf{B}_k(\ddot{\mathbf{q}}_k^* + \ddot{\mathbf{q}}_{pd,k}) + \mathbf{J}_k^T \mathbf{f}_k^* + \mathbf{h}_k + \mathbf{g}_k; \\
 \ddot{\mathbf{q}}_{pd,k} &= \mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k (c_v(\dot{\mathbf{q}}_k^* - \dot{\mathbf{q}}_k) + c_p(\mathbf{q}_k^* - \mathbf{q}_k)).
 \end{aligned} \tag{4}$$

Here, $\mathbf{B}_k \in \mathbb{R}^{n \times n}$ represents the joint inertia matrix, $\mathbf{h}_k \in \mathbb{R}^n$ corresponds to the Coriolis and centrifugal term, and $\mathbf{g}_k \in \mathbb{R}^n$ is the gravitational term. The feedback joint acceleration, denoted as $\ddot{\mathbf{q}}_{pd,k}$, is calculated by summing the derivative and proportional feedback terms, which are scaled by the scalar gains c_v and c_p , respectively. $\dot{\mathbf{q}}_k, \mathbf{q}_k \in \mathbb{R}^n$ represent the measured joint velocity and position, while the desired variables $\dot{\mathbf{q}}_k^*, \mathbf{q}_k^* \in \mathbb{R}^n$ are obtained by integrating $\ddot{\mathbf{q}}_k^*$ as follows:

$$\dot{\mathbf{q}}_k^* = \dot{\mathbf{q}}_{k-1}^* + \Delta t \ddot{\mathbf{q}}_k^*; \tag{6}$$

$$\mathbf{q}_k^* = \mathbf{q}_{k-1}^* + \Delta t \dot{\mathbf{q}}_{k-1}^* + \frac{\Delta t^2}{2} \ddot{\mathbf{q}}_k^*. \tag{7}$$

Since the feedback acceleration is responsible for enforcing the tracking of the desired joint positions and velocities, which are computed on the basis of the IK and JC tasks, it is scaled using the same weight matrix \mathbf{w}_1 .

In the following paragraphs, we will delve into the definition of problem (1), specifically focusing on the management of task weights, the constraints, and the generation of references. Once the details have been defined, we will proceed to discuss the stability of the control framework.

A. Task weights

To effectively handle the four concurrent tasks and determine their trade-off for each degree of freedom, we need to assign weights that reflect their relevance. For this purpose, we begin by using three weight diagonal matrices $\mathbf{w}_1^{\mathcal{L}}, \mathbf{w}_2^{\mathcal{L}}, \mathbf{w}_3^{\mathcal{L}} \in \mathbb{R}^{6 \times 6}$. These matrices represent the relative importance of the IK, AC, and FC tasks, respectively, for each Cartesian degree of freedom in the local reference frame \mathcal{L} . It's important to note that the sum of the diagonal elements pertaining to the same degree of freedom always equals 1, which means that some of these elements are often zero. This zero value indicates the insignificance of specific tasks in the directions corresponding to these zero elements. For instance, when the robot is not in contact along a particular axis, the FC task in that direction becomes irrelevant. However, the simultaneous deactivation of all the tasks associated with a particular degree of freedom (e.g., the AC and FC tasks for $\mathbf{f}_z^{\mathcal{L}}$, or the IK and JC tasks for $\ddot{\mathbf{x}}_y^{\mathcal{L}}$) does not make the associated optimal control problem indeterminate, as the resulting QP Hessians are regularized. Therefore, when the degrees of freedom are deactivated, the optimization variables associated with them become null. The diagonal elements of these matrices are determined and updated online using the intelligent adaptation algorithm described in Section IV. These terms are adjusted as C^0 functions to ensure a continuous transition in the optimal control problem (1). Since the Cartesian variables in the tasks are expressed in the world frame \mathcal{W} , it is necessary to rotate the local diagonal matrices as follows:

$$\begin{aligned} \mathbf{w}_1 &= {}^{\mathcal{W}}D_{\mathcal{L}} \mathbf{w}_1^{\mathcal{L}} {}^{\mathcal{W}}D_{\mathcal{L}}^T; & \mathbf{w}_2 &= {}^{\mathcal{W}}D_{\mathcal{L}} \mathbf{w}_2^{\mathcal{L}} {}^{\mathcal{W}}D_{\mathcal{L}}^T; \\ \mathbf{w}_3 &= {}^{\mathcal{W}}D_{\mathcal{L}} \mathbf{w}_3^{\mathcal{L}} {}^{\mathcal{W}}D_{\mathcal{L}}^T. \end{aligned} \quad (8)$$

The resulting matrices $\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3$ are non-diagonal, symmetric and positive-definite.

B. Constraints

To ensure that the robot does not exert excessive force, we constrain the wrench to respect the limits $\mathbf{f}_{max}, \mathbf{f}_{min}$. Additionally, during contact phases, we aim to optimize a feasible wrench that observes the friction cone. We implement this constraint as

$$\begin{bmatrix} w_3^x \mathbf{I}_{2 \times 2} & 0 \\ 0 & w_3^y \mathbf{I}_{2 \times 2} \end{bmatrix} \mathbf{U}(\mu) \mathbf{f}_k \leq 0, \quad (9)$$

where the matrix $\mathbf{U} \in \mathbb{R}^{4 \times 6}$, which depends on the friction coefficient μ , represents the linearized friction cone in the case of single contact point [24]. The weights $w_3^x, w_3^y \in \mathbb{R}$ represent the priorities for the FC task along the Cartesian x and y directions, while $\mathbf{I}_{2 \times 2} \in \mathbb{R}^{2 \times 2}$ is the identity matrix. The continuous adjustment of these weights reflects the extent to which contact with the environment is necessary, ensuring a gradual and controlled activation of the constraint (9). Given that $\mu < 1$, this constraint is always consistent with the upper and lower limits on the wrench $\mathbf{f}_{max}, \mathbf{f}_{min}$. The final boundaries, $\underline{\mathbf{f}}$ and $\bar{\mathbf{f}}$, are determined by selecting the most limiting conditions.

When considering the joint acceleration limits $\bar{\mathbf{q}}, \bar{\dot{\mathbf{q}}}$, it is necessary to account for velocity and position constraints.

Merely constraining the integrals of the controlled variables is inadequate to ensure that the resulting integral values remain within acceptable boundaries. This traditional approach can ensure feasibility between control bounds and imposed integral constraints only when they have a relative degree of 1. So, in our scenario, the position limits would not be respected. To fulfill all constraints within an instantaneous controller, we adopt a methodology rooted in the invariance control approach [25], which has demonstrated superior performance in the current state-of-the-art.

C. References

In total, four references need to be generated, one for each cost term that makes up the cost function of the QP controller (1). In the following, we explain the definition of each of them.

1) *IK task reference*: The desired Cartesian acceleration $\ddot{\mathbf{x}}_{d,k}$ is computed as

$$\ddot{\mathbf{x}}_{d,k} = \mathbf{K}_p \mathbf{e}_{x,k} - \mathbf{K}_v \dot{\mathbf{x}}. \quad (10)$$

In this equation, $\mathbf{e}_{x,k} = \mathbf{x}_{d,k} \ominus \mathbf{x}_k$ represents the Cartesian pose error, where $\mathbf{x}_{d,k}, \mathbf{x}_k \in \mathbb{R}^3 \times \mathcal{S}^3$ are the desired and measured end effector Cartesian poses, respectively. These poses belong to the Lie group $\mathbb{R}^3 \times \mathcal{S}^3$, as we represent their translation part as vectors in \mathbb{R}^3 and their rotation part as quaternions in \mathcal{S}^3 . Their difference $\mathbf{e}_{x,k} \in \mathbb{R}^6$ is a 6-dimensional vector since the rotation part is converted to the axis-angle representation. Furthermore, the Cartesian velocity, with respect to the frame \mathcal{W} , can be expressed as a vector in the Euclidean space \mathbb{R}^6 , which is isomorphic to the Lie Algebra of the group [26]. The gains $\mathbf{K}_p, \mathbf{K}_v \in \mathbb{R}^{6 \times 6}$ are defined as

$$\begin{aligned} \mathbf{K}_p &= \begin{bmatrix} k_p \mathbf{I}_{3 \times 3} & 0 \\ 0 & k_\theta \mathbf{I}_{3 \times 3} \end{bmatrix}; \\ \mathbf{K}_v &= \begin{bmatrix} 2\sqrt{k_p} \mathbf{I}_{3 \times 3} & 0 \\ 0 & 2\sqrt{k_\theta} \mathbf{I}_{3 \times 3} \end{bmatrix}, \end{aligned} \quad (11)$$

and determine the acceleration according to the position and the velocity error. Using these matrices, we establish the reference acceleration to match that of a desired second-order dynamic system critically damped. Consequently, we can interpret the IK task as a Cartesian Impedance control task.

Therefore, the dynamics of the closed-loop system when the IK task is active can be described by considering the control torque expression (4) and the original dynamics of the manipulator in Cartesian space, represented by the equation

$$\Lambda_k (\ddot{\mathbf{x}}_k - \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k) + \mathbf{C}_k \dot{\mathbf{x}}_k + \hat{\mathbf{g}}_k = \mathbf{J}_k^{T\dagger} \boldsymbol{\tau}_k + \mathbf{f}_{ext,k}. \quad (12)$$

Here, $\Lambda_k, \mathbf{C}_k \in \mathbb{R}^{6 \times 6}$ are respectively the Cartesian inertia and Coriolis matrices, $\hat{\mathbf{g}}_k \in \mathbb{R}^6$ denotes the Cartesian gravitational force, $\mathbf{J}_k^{T\dagger} = \Lambda^T \mathbf{J} \mathbf{B}^{-1T} \in \mathbb{R}^{6 \times n}$ indicates the dynamic consistent pseudo-inverse of the transposed Jacobian [27], and $\mathbf{f}_{ext,k}$ indicates the external wrench. By substituting $\boldsymbol{\tau}_k$ with (4) and noting that $\mathbf{f}_k^* = 0$ because we are optimizing only the desired joint acceleration $\ddot{\mathbf{q}}_k^*$, the Cartesian dynamics (12) simplifies to

$$\Lambda_k \ddot{\mathbf{x}}_k = \Lambda_k (\ddot{\mathbf{x}}_{d,k} + \ddot{\mathbf{x}}_{pd,k}) + \mathbf{f}_{ext,k}, \quad (13)$$

where the feedback acceleration $\ddot{\mathbf{x}}_{pd,k} \in \mathbb{R}^6$ is defined as

$$\ddot{\mathbf{x}}_{pd,k} = \mathbf{J}_k (c_v(\dot{\mathbf{q}}_k^* - \dot{\mathbf{q}}_k) + c_p(\mathbf{q}_k^* - \mathbf{q}_k)). \quad (14)$$

Finally, by formulating the reference acceleration as in (10), we arrive at the dynamics of the closed-loop system:

$$\Lambda_k (\ddot{\mathbf{x}}_k + \mathbf{K}_v \dot{\mathbf{x}}_k + \mathbf{K}_p (-\mathbf{e}_{x,k}) + \ddot{\mathbf{x}}_{pd,k}) = \mathbf{f}_{ext,k}; \quad (15)$$

2) *AC task reference*: The desired admittance force $\mathbf{f}_{adm,k}$ combines with the operator's force to guide the manipulator towards the desired posture. This active contribution ensures compliance with the operator's intentions and reduces the effort required for interaction. To determine the desired reference for the admittance force, we need to compare the desired dynamics with the closed-loop system dynamics when the AC task is active. Therefore, we examine the baseline equation for Cartesian dynamics (12) and the control torque (4). In this context, we set $\ddot{\mathbf{q}}_k^* = \ddot{\mathbf{q}}_{pd,k} = 0$ as our focus is currently on optimizing only the desired wrench \mathbf{f}_k^* . By combining these two equations, we arrive at the following expression:

$$\Lambda_k (\ddot{\mathbf{x}}_k - \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k) = \mathbf{f}_k^* + \mathbf{f}_{ext,k}. \quad (16)$$

In order to obtain the desired response closed-loop dynamics

$$\Lambda_d \ddot{\mathbf{x}}_k + \mathbf{C}_d \dot{\mathbf{x}}_k = \mathbf{f}_{ext,k}, \quad (17)$$

where $\Lambda_d = \Lambda_d \mathbf{I}_{6 \times 6}$, $\mathbf{C}_d = C_d \mathbf{I}_{6 \times 6}$ are the desired Cartesian inertia and damping matrices, respectively, the admittance force must be

$$\mathbf{f}_{adm,k} = \mathbf{f}_k^* = ([\Lambda_k \Lambda_d^{-1}] - \mathbf{I}_{6 \times 6}) \mathbf{f}_{ext,k} - (\mathbf{C}_d \mathbf{J}_k + \Lambda_k \dot{\mathbf{J}}_k) \dot{\mathbf{q}}_k, \quad (18)$$

or, alternatively,

$$\mathbf{f}_{adm,k} = \mathbf{f}_k^* = \Lambda_k (\ddot{\mathbf{x}}_k - \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k) - \Lambda_d \ddot{\mathbf{x}}_k - (\mathbf{C}_d \mathbf{J}_k + \Lambda_k \dot{\mathbf{J}}_k) \dot{\mathbf{q}}_k. \quad (19)$$

We have opted to use the formulation (18) in order to avoid the need to calculate the Cartesian acceleration $\ddot{\mathbf{x}}_k$. Additionally, the diagonal structure of the desired inertia matrix Λ_d simplifies the matrix inversion process.

3) *FC task reference*: The desired contact force $\mathbf{f}_{ctc,k}$ is obtained by applying the required local contact force, $\mathbf{f}_{d,k}^{\mathcal{L}}$, chosen by the autonomous environment adaptation policy, to the transformation (3):

$$\mathbf{f}_{ctc,k} = {}^{\mathcal{W}}D_{\mathcal{L}} \mathbf{f}_{d,k}^{\mathcal{L}} \in \mathbb{R}^6. \quad (20)$$

The local contact force $\mathbf{f}_{d,k}^{\mathcal{L}}$ is updated online according to \mathcal{C}^0 continuity constraints and is limited to the maximum value

$$\mathbf{f}_{d,k}^{\mathcal{L}} \leq \bar{\mathbf{f}}_d^{\mathcal{L}}. \quad (21)$$

By incorporating (20) into (16), we obtain the closed-loop system dynamics when the FC task is active:

$$\Lambda_k (\ddot{\mathbf{x}}_k - \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k) = {}^{\mathcal{W}}D_{\mathcal{L}} \mathbf{f}_{d,k}^{\mathcal{L}} + \mathbf{f}_{ext,k}. \quad (22)$$

4) *JC task reference*: Finally, we compute the desired joint acceleration $\ddot{\mathbf{q}}_d$ as

$$\ddot{\mathbf{q}}_d = k_q \mathbf{e}_{q,k} - 2\sqrt{k_q} \dot{\mathbf{q}}_k, \quad (23)$$

where $\mathbf{e}_{q,k} = \mathbf{q}_d - \mathbf{q}_k$ represents the joint position error, with \mathbf{q}_d a convenient joint configuration chosen to be far from the joint limits and singularities. k_q is a scalar gain.

D. Stability

After establishing the four control references as explained earlier, we can substitute them into (1), leading to the final formulation of the control problem, represented by (24). This problem can be separated into two distinct single-variable sub-problems. The first sub-problem (24a) is responsible for determining the optimal joint acceleration $\ddot{\mathbf{q}}_k^*$, while the second sub-problem (24b) focuses on finding the optimal wrench \mathbf{f}_k^* .

$$\begin{aligned} \min_{\ddot{\mathbf{q}}_k} & \overbrace{\frac{1}{2} \ddot{\mathbf{q}}_k^T \mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k \ddot{\mathbf{q}}_k + \ddot{\mathbf{q}}_k^T \mathbf{J}_k^T \mathbf{w}_1 \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k}^{\text{IK task}} - \\ & \ddot{\mathbf{q}}_k^T \mathbf{J}_k^T \mathbf{w}_1 (\mathbf{K}_p \mathbf{e}_{x,k} - \mathbf{K}_v \dot{\mathbf{x}}_k) + \\ & \overbrace{\frac{1}{2} \ddot{\mathbf{q}}_k^T \mathbf{J}_k^T \epsilon \mathbf{w}_1 \mathbf{J}_k \ddot{\mathbf{q}}_k - \ddot{\mathbf{q}}_k^T \mathbf{J}_k^T \epsilon \mathbf{w}_1 \mathbf{J}_k (k_q \mathbf{e}_{q,k} - 2\sqrt{k_q} \dot{\mathbf{q}}_k)}^{\text{JC task}}; \\ \text{s.t.} & \quad \underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}}_k \leq \bar{\ddot{\mathbf{q}}}. \end{aligned} \quad (24a)$$

$$\begin{aligned} \min_{\mathbf{f}_k} & \overbrace{\frac{1}{2} \mathbf{f}_k^T \mathbf{w}_2 \mathbf{f}_k - \mathbf{f}_k^T \mathbf{w}_2 ([\Lambda_k \Lambda_d^{-1}] - \mathbf{I}_{6 \times 6}) \mathbf{f}_{ext,k}}^{\text{AC task}} + \\ & \mathbf{f}_k^T \mathbf{w}_2 (\mathbf{C}_d \mathbf{J}_k + \Lambda_k \dot{\mathbf{J}}_k) \dot{\mathbf{q}}_k + \\ & \overbrace{\frac{1}{2} \mathbf{f}_k^T \mathbf{w}_3 \mathbf{f}_k - \mathbf{f}_k^T \mathbf{w}_3 \mathbf{f}_{ctc,k}}^{\text{FC task}}; \\ \text{s.t.} & \quad \underline{\mathbf{f}} \leq \mathbf{f}_k \leq \bar{\mathbf{f}}. \end{aligned} \quad (24b)$$

The Karush-Kuhn-Tucker (KKT) conditions [28] linked to the two sub-problems are instrumental in deriving an analytical expression for the two optimal variables. For the first sub-problem (24a) concerning $\ddot{\mathbf{q}}_k^*$ the conditions are

$$\begin{aligned} (1 + \epsilon) \mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k \ddot{\mathbf{q}}_k^* - \mathbf{J}_k^T \mathbf{w}_1 (\mathbf{K}_p \mathbf{e}_{x,k} - \mathbf{K}_v \dot{\mathbf{x}}_k - \dot{\mathbf{J}}_k \dot{\mathbf{q}}_k) - \\ \mathbf{J}_k^T \epsilon \mathbf{w}_1 \mathbf{J}_k (k_q \mathbf{e}_{q,k} - 2\sqrt{k_q} \dot{\mathbf{q}}_k) + \bar{\boldsymbol{\mu}}_q^T + \underline{\boldsymbol{\mu}}_q^T = 0; \end{aligned} \quad (25a)$$

$$\ddot{\mathbf{q}}_k^* - \bar{\ddot{\mathbf{q}}} \leq 0; \quad \bar{\boldsymbol{\mu}}_q^T (\ddot{\mathbf{q}}_k^* - \bar{\ddot{\mathbf{q}}}) = 0; \quad \bar{\boldsymbol{\mu}}_q^T \geq 0. \quad (25b)$$

$$\ddot{\mathbf{q}}_k^* - \underline{\ddot{\mathbf{q}}} \geq 0; \quad \underline{\boldsymbol{\mu}}_q^T (\ddot{\mathbf{q}}_k^* - \underline{\ddot{\mathbf{q}}}) = 0; \quad \underline{\boldsymbol{\mu}}_q^T \leq 0. \quad (25c)$$

The KKT conditions associated to the second sub-problem (24b) concerning \mathbf{f}_k^* are:

$$\begin{aligned} [\mathbf{w}_2 + \mathbf{w}_3] \mathbf{f}_k^* - \mathbf{w}_2 ([\Lambda_k \Lambda_d^{-1}] - \mathbf{I}_{6 \times 6}) \mathbf{f}_{ext,k} + \\ \mathbf{w}_2 (\mathbf{C}_d \mathbf{J}_k + \Lambda_k \dot{\mathbf{J}}_k) \dot{\mathbf{q}}_k - \mathbf{w}_3 \mathbf{f}_{ctc,k} + \bar{\boldsymbol{\mu}}_f^T + \underline{\boldsymbol{\mu}}_f^T = 0; \end{aligned} \quad (26a)$$

$$\mathbf{f}_k^* - \bar{\mathbf{f}} \leq 0; \quad \bar{\boldsymbol{\mu}}_f^T (\mathbf{f}_k^* - \bar{\mathbf{f}}) = 0; \quad \bar{\boldsymbol{\mu}}_f^T \geq 0. \quad (26b)$$

$$\mathbf{f}_k^* - \underline{\mathbf{f}} \geq 0; \quad \underline{\boldsymbol{\mu}}_f^T (\mathbf{f}_k^* - \underline{\mathbf{f}}) = 0; \quad \underline{\boldsymbol{\mu}}_f^T \leq 0. \quad (26c)$$

The stability of the presented QP controller can be discussed using Lyapunov theory, with the following positive definite

candidate function:

$$V = \frac{1}{2} \dot{\mathbf{x}}_k^T \dot{\mathbf{x}}_k + \frac{1}{2(1+\epsilon)} \mathbf{e}_{x,k}^T \mathbf{K}_p \mathbf{e}_{x,k} + \frac{\epsilon k_q}{2(1+\epsilon)} \mathbf{e}_{q,k}^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{e}_{q,k}; \quad (27)$$

The associated time derivative is

$$\begin{aligned} \dot{V} = & \dot{\mathbf{x}}_k^T \dot{\mathbf{x}}_k + \frac{1}{1+\epsilon} \dot{\mathbf{e}}_{x,k}^T \mathbf{K}_p \mathbf{e}_{x,k} + \\ & \frac{\epsilon k_q}{1+\epsilon} \dot{\mathbf{e}}_{q,k}^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{e}_{q,k} + \frac{\epsilon k_q}{1+\epsilon} \mathbf{e}_{q,k}^T \mathbf{J}_k^T \dot{\mathbf{J}}_k \mathbf{e}_{q,k}. \end{aligned} \quad (28)$$

This equation can be rewritten by explicating the acceleration $\ddot{\mathbf{x}}_k$ using the Cartesian dynamics (12), where we incorporate the analytical expression of τ as given in (4). In this context, we substitute the optimal variables $\ddot{\mathbf{q}}_k^*$ and \mathbf{f}_k^* with their analytical expressions derived from the KKT conditions (25), (26). We also disregard the feedback joint acceleration $\ddot{\mathbf{q}}_{pd,k}$, as defined in (5), since it does not affect stability when the gains are appropriately tuned [23]. As a result, we arrive at the following form for the derivative of the Lyapunov candidate:

$$\begin{aligned} \dot{V} = & \frac{1}{1+\epsilon} \dot{\mathbf{x}}_k^T \left(\epsilon \dot{\mathbf{J}}_k \mathbf{J}_k^\dagger - \mathbf{K}_v - 2\epsilon \sqrt{k_q} \mathbf{I}_{6 \times 6} \right) \dot{\mathbf{x}}_k + \\ & \frac{1}{1+\epsilon} \dot{\mathbf{x}}_k^T \mathbf{K}_p \mathbf{e}_{x,k} + \frac{\epsilon k_q}{1+\epsilon} \dot{\mathbf{q}}_k^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{e}_{q,k} - \\ & \frac{1}{1+\epsilon} \dot{\mathbf{q}}_k^T [\mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k]^{-1} (\bar{\boldsymbol{\mu}}_q^T + \underline{\boldsymbol{\mu}}_q^T) + \\ & \frac{1}{1+\epsilon} \dot{\mathbf{e}}_{x,k}^T \mathbf{K}_p \mathbf{e}_{x,k} + \frac{\epsilon k_q}{1+\epsilon} \dot{\mathbf{e}}_{q,k}^T \mathbf{J}_k^T \mathbf{J}_k \mathbf{e}_{q,k} + \\ & \frac{\epsilon k_q}{1+\epsilon} \mathbf{e}_{q,k}^T \mathbf{J}_k^T \dot{\mathbf{J}}_k \mathbf{e}_{q,k} + E_{ext} + E_{int}; \end{aligned} \quad (29)$$

$$E_{ext} = \dot{\mathbf{x}}_k^T \boldsymbol{\Lambda}_k^{-1} \mathbf{f}_{ext,k}; \quad (30)$$

$$\begin{aligned} E_{int} = & \dot{\mathbf{x}}_k^T \boldsymbol{\Lambda}_k^{-1} \mathbf{f}_k^* = \dot{\mathbf{x}}_k^T \mathbf{P}_k \mathbf{w}_2 \left([\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_d^{-1}] - \mathbf{I}_{6 \times 6} \right) \mathbf{f}_{ext,k} + \\ & \dot{\mathbf{x}}_k^T \mathbf{P}_k \mathbf{w}_2 \left(\boldsymbol{\Lambda}_k \dot{\mathbf{J}}_k \mathbf{J}_k^\dagger - \mathbf{C}_d \right) \dot{\mathbf{x}}_k + \\ & \dot{\mathbf{x}}_k^T \mathbf{P}_k \mathbf{w}_3 \mathbf{f}_{ctc,k} - \dot{\mathbf{x}}_k^T \mathbf{P}_k (\bar{\boldsymbol{\mu}}_f^T + \underline{\boldsymbol{\mu}}_f^T); \end{aligned} \quad (31)$$

$$\mathbf{P}_k = \boldsymbol{\Lambda}_k^{-1} [\mathbf{w}_2 + \mathbf{w}_3]^{-1}. \quad (32)$$

In this context, E_{ext} represents the external energy introduced into the system as a result of interaction with the operator, while E_{int} signifies the energy internally generated by the optimal control wrench when either the AC or the FC tasks, or both, are active. Both of these energy values are normalized by the mass. The matrix $\mathbf{P}_k \in \mathbb{R}^{6 \times 6}$ is symmetric and positive definite.

It can be shown that E_{int} is limited by the following equation:

$$E_{int} < \dot{\mathbf{x}}_k^T (\boldsymbol{\Lambda}_d^{-1} - \boldsymbol{\Lambda}_k^{-1}) \mathbf{f}_{ext,k} + \dot{\mathbf{x}}_k^T \boldsymbol{\Lambda}_k^{-1} {}^W D_{\mathcal{L}} \bar{\mathbf{f}}_d^{\mathcal{L}}. \quad (33)$$

Disturbances with bounded dependencies on the robot state, like those influenced by the Cartesian inertia $\boldsymbol{\Lambda}_k$, do not affect intrinsic system stability [23]. Therefore, we can consider both E_{int} and E_{ext} as perturbations. The upper limit (33)

is determined by considering the bound described in (21) and the conditions:

$$\mathbf{P}_k \mathbf{w}_2 \leq \boldsymbol{\Lambda}_k^{-1}; \quad \mathbf{P}_k \mathbf{w}_3 \leq \boldsymbol{\Lambda}_k^{-1}; \quad (34a)$$

$$\dot{\mathbf{x}}_k^T \mathbf{P}_k \mathbf{w}_2 \left(\boldsymbol{\Lambda}_k \dot{\mathbf{J}}_k \mathbf{J}_k^\dagger - \mathbf{C}_d \right) \dot{\mathbf{x}}_k < 0; \quad (34b)$$

$$\dot{\mathbf{x}}_k^T \mathbf{P}_k (\bar{\boldsymbol{\mu}}_f^T + \underline{\boldsymbol{\mu}}_f^T) \geq 0. \quad (34c)$$

The inequality (34b) is valid since $\mathbf{P}_k \mathbf{w}_2$ derives from a multiplication of symmetric positive definite matrices, and $(\boldsymbol{\Lambda}_k \dot{\mathbf{J}}_k \mathbf{J}_k^\dagger - \mathbf{C}_d)$ is symmetric and negative definite when $\mathbf{C}_d > \boldsymbol{\Lambda}_k \dot{\mathbf{J}}_k \mathbf{J}_k^\dagger$.

The inequality (34c) holds because $\bar{\boldsymbol{\mu}}_f^T = \underline{\boldsymbol{\mu}}_f^T = 0$ whenever \mathbf{f}_k^* is far from the limits. Additionally, when $\mathbf{f}_k^* = \bar{\mathbf{f}}$, the KKT conditions (26) result in $\bar{\boldsymbol{\mu}}_f^T > 0$, and $\dot{\mathbf{x}}_k > 0$ if \mathbf{f}_k^* is performing positive work. Equivalently, when $\mathbf{f}_k^* = \underline{\mathbf{f}}$, $\underline{\boldsymbol{\mu}}_f^T < 0$ and $\dot{\mathbf{x}}_k < 0$.

The upper limit of the internally generated energy, as defined in (33), can be further reduced by taking into account that \mathbf{w}_3 only becomes active gradually when the end effector of the robot is in contact with the environment, which means $\dot{\mathbf{x}}_k^T \boldsymbol{\Lambda}_k^{-1} \mathbf{f}_{ctc,k} = 0$. Furthermore, when $\dot{\mathbf{x}}_k > 0$, $(\mathbf{C}_d - \boldsymbol{\Lambda}_k \dot{\mathbf{J}}_k \mathbf{J}_k^\dagger)^{-1} ([\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_d^{-1}] - \mathbf{I}_{6 \times 6}) \mathbf{f}_{ext,k}$, the inequality (34b) is able to compensate the positive active term $\dot{\mathbf{x}}_k^T ([\boldsymbol{\Lambda}_k \boldsymbol{\Lambda}_d^{-1}] - \mathbf{I}_{6 \times 6}) \mathbf{f}_{ext,k}$. Consequently, the upper bound for internally generated energy becomes zero.

In light of the earlier findings, we can revisit (29). By acknowledging that $\dot{\mathbf{e}}_{x,k} = -\dot{\mathbf{x}}_k$ and $\dot{\mathbf{e}}_{q,k} = -\dot{\mathbf{q}}_k$, we can streamline the equation as follows:

$$\begin{aligned} \dot{V} = & \frac{1}{1+\epsilon} \dot{\mathbf{x}}_k^T \left(\epsilon \dot{\mathbf{J}}_k \mathbf{J}_k^\dagger - \mathbf{K}_v - 2\epsilon \sqrt{k_q} \mathbf{I}_{6 \times 6} \right) \dot{\mathbf{x}}_k + \\ & \frac{1}{1+\epsilon} \dot{\mathbf{q}}_k^T [\mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k]^{-1} (\bar{\boldsymbol{\mu}}_q^T + \underline{\boldsymbol{\mu}}_q^T) + \frac{\epsilon k_q}{1+\epsilon} \mathbf{e}_{q,k}^T \mathbf{J}_k^T \dot{\mathbf{J}}_k \mathbf{e}_{q,k}. \end{aligned} \quad (35)$$

The terms involving the product $\epsilon \dot{\mathbf{J}}_k$ can be considered small and negligible. Consequently, it leads to $\dot{V} < 0$ due to the properties of the matrices involved. Here is a more explicit breakdown of the reasoning:

- $(-\mathbf{K}_v - 2\epsilon \sqrt{k_q} \mathbf{I}_{6 \times 6})$ is symmetric and negative definite, thus the quadratic form is strictly negative.
- $\frac{1}{1+\epsilon} \dot{\mathbf{q}}_k^T [\mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k]^{-1} (\bar{\boldsymbol{\mu}}_q^T + \underline{\boldsymbol{\mu}}_q^T) \geq 0 \forall \dot{\mathbf{q}}_k, \bar{\boldsymbol{\mu}}_q^T, \underline{\boldsymbol{\mu}}_q^T$.
Indeed, $[\mathbf{J}_k^T \mathbf{w}_1 \mathbf{J}_k]^{-1}$ is symmetric and positive definite, and $\bar{\boldsymbol{\mu}}_q^T = 0, \underline{\boldsymbol{\mu}}_q^T = 0$ whenever $\ddot{\mathbf{q}}_k^*$ is far from the limits. When $\ddot{\mathbf{q}}_k^* = \ddot{\bar{\mathbf{q}}}$, the KKT conditions (25) result in $\bar{\boldsymbol{\mu}}_q^T > 0$. Additionally, with sufficiently small sampling time Δt , the constraints employed lead to $\dot{\mathbf{q}}_k > 0$ [25]. Similarly, when $\ddot{\mathbf{q}}_k^* = \ddot{\underline{\mathbf{q}}}$, $\underline{\boldsymbol{\mu}}_q^T < 0$ and $\dot{\mathbf{q}}_k < 0$.

Thus, we have demonstrated the intrinsic system stability.

IV. ADAPTATION MODULE

This section presents a comprehensive description of the adaptation module implemented to dynamically adjust the behaviour of the robot. As mentioned earlier, this automatic policy directly works on the QP parameters, enabling the robot to adapt to the job requirements by updating the task priorities.

It employs a RL strategy to dynamically select the local weight matrices $w_{1,k}^{\mathcal{L}}, w_{2,k}^{\mathcal{L}}, w_{3,k}^{\mathcal{L}}$ and the most appropriate local contact force $f_{d,k}^{\mathcal{L}}$, based on the continuous observed state $s_k \in \mathbb{R}^6$. The latter is defined as

$$s_k = \{e_{x,k}^{\mathcal{L}} \quad f_{env,k}^{\mathcal{L}} \quad f_{ope,k}^{\mathcal{L}}\}, \quad (36)$$

where, in addition to the previously defined error pose $e_{x,k}$, we have the external forces originating from contact with the environment, denoted as $f_{env,k}^{\mathcal{L}} \in \mathbb{R}^6$, and the external forces exerted by the operator, represented as $f_{ope,k}^{\mathcal{L}} \in \mathbb{R}^6$. The RL algorithm is specialized for a particular axis of the local frame \mathcal{L} . Therefore, the observed state, represented by three 6-dimensional vectors, is processed by six distinct agents, each responsible for handling one direction, corresponding to a set of three scalars. Hence, at each time step k , an agent receives the state

$$s_k = \{e_{x,k}^{\mathcal{L}} \quad f_{env,k}^{\mathcal{L}} \quad f_{ope,k}^{\mathcal{L}}\} \in \mathbb{R}^3, \quad (37)$$

with the elements being normalized to enhance learning stability. On the basis of s_k , the latter determines an action denoted as:

$$a_k = \{\dot{w}_{1,k}^{\mathcal{L}} \quad \dot{w}_{2,k}^{\mathcal{L}} \quad \dot{w}_{3,k}^{\mathcal{L}} \quad \dot{f}_{d,k}^{\mathcal{L}}\} \in \mathbb{R}^4, \quad (38a)$$

$$\dot{w}_{1,k}^{\mathcal{L}}, \dot{w}_{2,k}^{\mathcal{L}}, \dot{w}_{3,k}^{\mathcal{L}}, \dot{f}_{d,k}^{\mathcal{L}} \in [-1, 1], \quad (38b)$$

representing the selected rate of change for task priorities and the desired contact force. This action (38a) is selected according to the agent's policy π , which maps states s_k to actions a_k . In response to the action, the agent receives the subsequent state s_{k+1} and a scalar reward r_k . While the rate of change can be discontinuous, as there are no constraints on the difference between two consecutive actions (e.g., $a_k - a_{k-1}$), the resulting module output, defined as

$$o_k = \{w_{1,k}^{\mathcal{L}} \quad w_{2,k}^{\mathcal{L}} \quad w_{3,k}^{\mathcal{L}} \quad f_{d,k}^{\mathcal{L}}\} \in \mathbb{R}^4, \quad (39a)$$

$$w_{1,k}^{\mathcal{L}}, w_{2,k}^{\mathcal{L}}, w_{3,k}^{\mathcal{L}} \in [0, 1]; \quad (39b)$$

$$w_{1,k}^{\mathcal{L}} + w_{2,k}^{\mathcal{L}} + w_{3,k}^{\mathcal{L}} = 1, \quad (39c)$$

is C^0 , and is calculated using the following equations:

$$\check{w}_{1,k}^{\mathcal{L}} = \check{w}_{1,k-1}^{\mathcal{L}} + \sigma_w \Delta t \dot{w}_{1,k}^{\mathcal{L}} \quad \text{if} \quad -\frac{\sigma_w}{2} \leq \dot{w}_{1,k}^{\mathcal{L}} \leq \frac{\sigma_w}{2}; \quad (40a)$$

$$\check{w}_{2,k}^{\mathcal{L}} = \check{w}_{2,k-1}^{\mathcal{L}} + \sigma_w \Delta t \dot{w}_{2,k}^{\mathcal{L}} \quad \text{if} \quad -\frac{\sigma_w}{2} \leq \dot{w}_{2,k}^{\mathcal{L}} \leq \frac{\sigma_w}{2}; \quad (40b)$$

$$\check{w}_{3,k}^{\mathcal{L}} = \check{w}_{3,k-1}^{\mathcal{L}} + \sigma_w \Delta t \dot{w}_{3,k}^{\mathcal{L}} \quad \text{if} \quad -\frac{\sigma_w}{2} \leq \dot{w}_{3,k}^{\mathcal{L}} \leq \frac{\sigma_w}{2}; \quad (40c)$$

$$\check{f}_{d,k}^{\mathcal{L}} = \check{f}_{d,k-1}^{\mathcal{L}} + \sigma_f \Delta t \dot{f}_{d,k}^{\mathcal{L}} \quad \text{if} \quad 0 \leq \dot{f}_{d,k}^{\mathcal{L}} \leq \dot{f}_d^{\mathcal{L}}; \quad (40d)$$

$$w_{1,k}^{\mathcal{L}} = \frac{\exp \check{w}_{1,k}^{\mathcal{L}}}{\sum_{i=1}^3 \exp \check{w}_{i,k}^{\mathcal{L}}}; \quad w_{2,k}^{\mathcal{L}} = \frac{\exp \check{w}_{2,k}^{\mathcal{L}}}{\sum_{i=1}^3 \exp \check{w}_{i,k}^{\mathcal{L}}};$$

$$w_{3,k}^{\mathcal{L}} = \frac{\exp \check{w}_{3,k}^{\mathcal{L}}}{\sum_{i=1}^3 \exp \check{w}_{i,k}^{\mathcal{L}}}. \quad (40e)$$

In this context, scalar parameters σ_w and σ_f are employed to adjust the rate of change of the variables in o_k . They also define upper and lower limits to bound $\check{w}_{i,k}^{\mathcal{L}}$, which represents the weights before the normalization process, ensuring adherence to the constraints (39b) and (39c).

In the following paragraphs, we will provide a detailed explanation of the RL algorithm used in this study. We will

specify the reward function chosen to guide the agent during the learning process, explain the RL methods employed, and describe the approach used to tune the various hyperparameters of the algorithm.

A. Reward function

The reward plays a vital role in evaluating the desirability of the current behavior of the robot and guiding it towards appropriate actions. In our specific scenario, in cases when there are no external forces acting on the robot, its primary objective is to reach the target position by minimizing the positional error $e_{x,k}$. However, when the robot encounters an external force exerted by an operator $f_{ope,k}$, it should prioritize adhering to the operator's intentions with minimal resistance and maximum compliance. Conversely, if the end effector experiences an environmental force $f_{env,k}$ while maintaining a small positional error, the robot must sustain contact with the desired force $f_{d,k}$. To successfully accomplish these objectives, we define the reward function as a sum of four terms:

$$r_k = \sum_{i=1}^4 \hat{r}_{i,k}; \quad (41a)$$

$$\hat{r}_{1,k} = \alpha_1 \frac{|e_{x,k}^{\mathcal{L}}|}{\bar{e}_x^{\mathcal{L}}} \log \left(\frac{|f_{ope,k}^{\mathcal{L}}|}{\bar{f}_{ope}^{\mathcal{L}}} + p_1 \right); \quad (41b)$$

$$\hat{r}_{3,k} = -\alpha_2 \frac{|f_{ope,k}^{\mathcal{L}}|}{\bar{f}_{ope}^{\mathcal{L}}}; \quad (41c)$$

$$\hat{r}_{4,k} = \alpha_3 \left(1 - \frac{|f_{env,k}^{\mathcal{L}}|}{\bar{f}_{env}^{\mathcal{L}}} \right) \log \left(\frac{|f_{ope,k}^{\mathcal{L}}|}{\bar{f}_{ope}^{\mathcal{L}}} + p_1 \right); \quad (41d)$$

$$\hat{r}_{5,k} = \alpha_4 \left(\frac{f_{d,k}^{\mathcal{L}}}{\bar{f}_{d,k}^{\mathcal{L}}} \right) \log \left(\frac{|f_{env}^{\mathcal{L}}|}{\bar{f}_{env}^{\mathcal{L}}} + p_1 \right). \quad (41e)$$

Here, $\alpha_1, \alpha_2, \alpha_3$, and α_4 represent the positive weights assigned to each term. The scalar parameter p_1 is adjusted to limit the logarithm function's decay to $-\infty$. All the variables have been normalized within the range of $[0, 1]$. In the following list, the purpose of each reward term is described.

- The first term (41b) penalizes large positional errors between the actual manipulator coordinate and the target one. This encourages the robot to appreciate states that are closer to the target and increase $w_{1,k}^{\mathcal{L}}$. The factor $\log \left(\frac{|f_{ope,k}^{\mathcal{L}}|}{\bar{f}_{ope}^{\mathcal{L}}} + p_1 \right)$ is used to adjust the significance of the penalty term based on the presence of the operator force $f_{ope,k}^{\mathcal{L}}$. When the latter is absent, it results in a substantial negative weight. Conversely, when $f_{ope,k}^{\mathcal{L}}$ is present, the associated weight diminishes, reflecting that when following the operator's guidance, we tolerate deviations from the target Cartesian position.
- The second term (41c) penalizes large forces applied by the operator, caused by intense collaboration effort resulting from the reactive response of the robot. This reward term is essential in teaching the robot to activate the AC task by increasing $w_{2,k}^{\mathcal{L}}$ when required.
- The third term (41d) incentivizes the robot to enhance its interaction force with the environment. This helps the

robot to establish contact with the target workpiece or worksurface when required. When the agent detects a small $f_{env,k}^{\mathcal{L}}$, it interprets it as an indication of the nearby workpiece and activates the FC task by increasing $w_{3,k}^{\mathcal{L}}$ to enhance the interaction force and the reward. Once again, the term $\log\left(\frac{|f_{ope,k}^{\mathcal{L}}|}{f_{ope}^{\mathcal{L}}} + p_1\right)$ is included to guide the robot in transitioning from the AC to the FC tasks when contact with the operator is detected.

- The fourth term (41e) penalizes a nonzero desired contact force. The factor $\log\left(\frac{|f_{env,k}^{\mathcal{L}}|}{f_{env}^{\mathcal{L}}} + p_1\right)$ is employed to modulate the importance of this penalty term. When the robot is not in contact, and the associated contact force $f_{env,k}^{\mathcal{L}}$ is zero, a positive desired contact force is heavily discouraged. Conversely, if the robot is in contact, the impact of this penalty term diminishes, allowing for an increase in $f_{d,k}^{\mathcal{L}}$.

By combining these four terms, we create the comprehensive reward function (41a) that guides the robot towards desired behaviors in different scenarios.

B. RL methodology

Given the adeptness of the A3C algorithm in addressing deep learning-supported RL challenges [21], we have chosen it to implement the automatic environment updating module in our control solution. This algorithm approximates both the policy $\pi(a|s; \theta_\pi)$ and the value function $V_\pi(s_k; \theta_v)$ by leveraging parallel agents interacting with separate copies of the environment. In the chosen implementation variant, instead of using two artificial neural networks, both quantities are approximated by a single convolutional neural network, where the learning parameters θ_π and θ_v are partially shared. Consequently, for training the unique network we minimize a global loss function obtained from the sum of three terms:

$$L_k(\theta_\pi, \theta_v) = \sum_{i=1}^3 \hat{L}_{i,k}; \quad (42a)$$

$$\hat{L}_{1,k}(\theta_\pi, \theta_v) = -\log \pi(a_k|s_k; \theta_\pi)(R_k - V(s_k; \theta_v)); \quad (42b)$$

$$\hat{L}_{2,k}(\theta_v) = \lambda_v (R_k - V(s_k; \theta_v))^2; \quad (42c)$$

$$\hat{L}_{3,k}(\theta_\pi) = \lambda_\beta H(\pi(a_k|s_k; \theta_\pi)). \quad (42d)$$

Here, λ_v and λ_β are hyperparameters used to scale the importance of the value loss (42c) and the entropy loss (42d) terms, respectively. The single loss functions are specified in the following list.

- The policy loss term (42b) is derived from the unbiased estimate of the expected return $\mathbb{E}[R_k]$ [21]. Therefore, minimizing this loss translates to maximizing $\mathbb{E}[R_k]$. Subtracting $V(s_k; \theta_v)$ from R_k helps to lower the variance of the estimator [29].
- The value loss term (42c) aims to improve the return prediction capabilities of the value function estimator $V(s_k; \theta_v)$. This estimator learns the return associated with the state s_k following the current policy π . Thus, the difference $R_k - V(s_k; \theta_v)$ represents the advantage

of taking a certain generic action compared to the one suggested by the current policy [20].

- The entropy loss term (42d) is optional but aids in enhancing exploration and prevents premature convergence to a local minimum.

To update the shared common parameters θ_π and θ_v using the computed deltas $\Delta\theta_\pi$ and $\Delta\theta_v$ from each independent agent, we employ the RMSProp algorithm [30]. The update is performed according to the following equations:

$$\begin{aligned} g_\pi &\leftarrow \lambda_\alpha g_\pi + (1 - \lambda_\alpha) \Delta\theta_\pi^2; & \theta_\pi &\leftarrow \theta_\pi - \lambda_\eta \frac{\Delta\theta_\pi}{\sqrt{g_\pi + \lambda_\epsilon}}; \\ g_v &\leftarrow \lambda_\alpha g_v + (1 - \lambda_\alpha) \Delta\theta_v^2; & \theta_v &\leftarrow \theta_v - \lambda_\eta \frac{\Delta\theta_v}{\sqrt{g_v + \lambda_\epsilon}}, \end{aligned} \quad (43)$$

where λ_α is the decay factor, λ_η is the learning rate, and λ_ϵ is a hyperparameter chosen to stabilize square root computation in denominator of RMSProp update. Excessively large computed deltas can have a detrimental effect on the learning process. To mitigate this issue, we apply a clipping operation on $\Delta\theta$ after a specific limit denoted by λ_ζ . This ensures that the updates to the parameter values remain within a reasonable range.

C. Hyperparameters tuning

To effectively employ the A3C algorithm, the identification of an appropriate set of hyperparameters is crucial. We denote this set as

$$\mathcal{H} = \{\lambda_\eta \quad \lambda_\beta \quad n \quad \lambda_\epsilon \quad \lambda_\zeta \quad \lambda_\gamma \quad \lambda_v \quad \lambda_\alpha\}. \quad (44)$$

To tailor this set to our specific problem, we assess the performance reached by the algorithm at the end of the training. We use the mean reward r_k averaged over the last N training episodes as the fitness function. With a training size of T episodes, the fitness function is defined as:

$$F_\lambda = \frac{1}{N} \sum_{k=T-N}^T r_k. \quad (45)$$

To maximize this function, a gradient-free approach is necessary. Thus, we utilize a genetic algorithm, a well-established methodology widely employed for similar problems [31], [32].

V. EXPERIMENT SETUP

In this section, we provide details on the practical implementation of the proposed A3CQP controller and present the setup for the experiments carried out to assess its performance. In the following paragraphs, we will elaborate on the manipulator in use, the sensors and tools employed, and provide specific information about the algorithm parameters.

A. Hardware setup

This controller has been deployed and tested on a Franka EMIKA Panda robot. Two different force and torque sensing methods are used to distinguish between the operator's interactions with the last link of the robot and contact of the end effector with the workspace. An ATI mini45 F/T (force and torque) sensor is mounted between the last link of the

TABLE II: State limits

Limit	Value
$\bar{e}_x^{\mathcal{L}}$	1 <i>m</i>
$\bar{f}_{env}^{\mathcal{L}}$	30 <i>N</i>
$\bar{f}_{ope}^{\mathcal{L}}$	15 <i>N</i>
$\bar{f}_d^{\mathcal{L}}$	20 <i>N</i>

TABLE I: QP control parameters

Parameter	Value
c_p	3
c_v	5
k_p	150
k_θ	20
k_q	10
Λ_d	5
C_d	5
ϵ	1×10^{-4}
μ	0.4

TABLE III: Reward parameters

Parameter	Value
α_1	0.5
α_2	20
α_3	1
α_4	2
p_1	10^{-3}

robot and its end effector to specifically detect any contact with the workpiece or worksurface. This sensor provides the contact force $\mathbf{f}_{env,k}^{\mathcal{L}}$, already oriented in the local frame \mathcal{L} . During collaborative tasks, it's crucial for the operator not to touch the robot below the F/T sensor to prevent the misclassification of his interaction force as the environmental contact force. On the other hand, torque sensors embedded in each joint of the robot measure the external torque $\boldsymbol{\tau}_s$. The total external force applied at the end effector is then computed as $\mathbf{f}_{ext,k} = \mathbf{J}_k^{T\dagger} \boldsymbol{\tau}_s = \mathbf{f}_{env,k} + \mathbf{f}_{ope,k}$. By subtracting the contribution measured by the installed F/T sensor, we can isolate and determine the operator's interaction force alone.

The first experiment entails a collaborative peg-in-hole assembly. A 20 mm diameter peg is directly connected to the F/T sensor, while a 20.5 mm diameter hole is securely clamped to a known position in the worktable using a bench vise. You can see this setup in Figure 3a. The second experiment centers around collaborative wooden board polishing. For this purpose, we design a flange to connect an Einhell drill to the F/T sensor. The drill is equipped with a polishing disc, and we control its motor speed using an Arduino¹ controller, as shown in Figure 3b.

B. Software implementation

The QP controller operates at a frequency of 1 kHz, is integrated with the ROS framework, and is based on the Franka Control Interface². The control parameters of the QP have been carefully tuned to achieve a reactive behavior, allowing the robot to track the desired trajectory accurately with smooth control torque profiles and gentle movements. The final values of these control parameters are described in Table I. The wrench limits are defined as $\pm 20N$ for the forces and $\pm 5 Nm$ for the torques. The position, velocity and acceleration boundaries are directly retrieved by the robot specifications. The adaptation module has been implemented as a standalone Python ROS node, running at a frequency of 100 Hz. The elements of the input state s_k are normalized to fall within the range of $[0, 1]$ before being fed into the neural network. This normalization enhances learning stability and is also applied when processing these elements and the desired contact force

TABLE IV: A3C hyperparameters

Hyperparameter	Value
λ_η (Learning Rate)	$4.25339846 \times 10^{-6}$
λ_β (Entropy Loss Term Weight)	$3.371768978 \times 10^{-4}$
n (Number of Steps)	4
λ_ϵ (RMSProp Parameter)	$3.938214948 \times 10^{-5}$
λ_ζ (Clipping Threshold)	4.24498881
λ_γ (Discount Coefficient)	0.99
λ_v (Value Loss Term Weight)	0.5
λ_α (RMSProp Decay Factor)	0.99

TABLE V: Genetic Algorithm Settings

Setting	Value
Number of Generations	50
Number of Solutions to be Selected as Parents	4
Number of Solutions within the Population	8
Number of Genes in the Solution/Chromosome	5
Parent Selection Type	Steady-State
Number of Parents to Keep in the Next Population	1
Type of Crossover Operation	Single Point
Mutation Type	Random
Percentage of Genes to Mutate	10

$f_d^{\mathcal{L}}$ in the reward function. During the normalization process, each element is divided by its empirically determined upper bound, which is set at reasonably high values to prevent excessive values under typical conditions. You can find these parameters listed in Table II. The reward parameters have been manually tuned to describe the desired behavior of the agent in the environment. Details of these parameters can be found in Table III. The rate scaling parameters have been set to $\sigma_w = 6$ and $\sigma_f = 10$ in order to achieve task transitions from complete deactivation to complete activation within 1 or 2 seconds. The A3C algorithm has been implemented using the RL library Stable-Baselines3³. The hyperparameters of the algorithm, resulting from the tuning procedure, are summarized in Table IV. The genetic algorithm used for tuning the RL hyperparameters has been implemented using the library pygad⁴. The specific settings applied in this implementation are presented in Table V. The training of the A3C algorithm has been performed in a virtual Gym⁵ environment. The robot dynamics are simplified by modeling them as a one-degree-of-freedom point mass system. In this system, we omit the QP controller for the sake of simplicity. The control force applied is a weighted sum of several components: a PD position control contribution for inverse kinematic tracking, an admittance control force designed to achieve desired virtual dynamics, and a constant desired contact force. The dynamic equation for the simplified system is as follows:

$$\Lambda \ddot{x}_k = f_{ext,k}^{\mathcal{L}} + w_1 \Lambda (K_p e_{x,k} - K_v \dot{x}_k + \ddot{x}_{pd}) + w_2 \left(\frac{\Lambda}{\Lambda_d} - 1 \right) f_{ext,k}^{\mathcal{L}} - w_2 C_d \dot{x} + w_3 f_d^{\mathcal{L}}, \quad (46)$$

with $\Lambda = 10$.

³<https://stable-baselines3.readthedocs.io/en/master/modules/a2c.html>

⁴<https://pygad.readthedocs.io/en/latest/>

⁵<https://www.gymnasium.dev/index.html>

¹<https://www.arduino.cc>

²<https://frankaemika.github.io/docs>

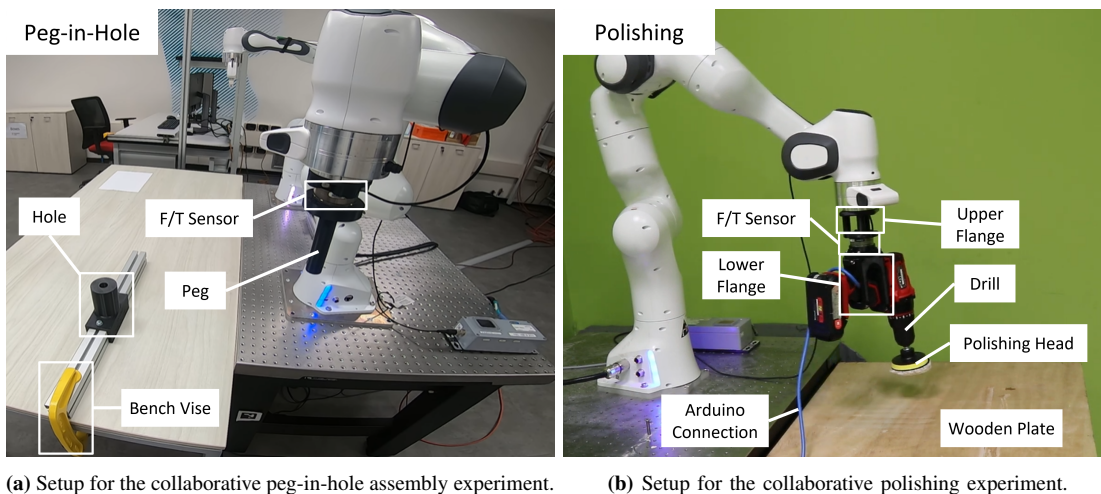


Fig. 3: The experimental setups. It's important to note that the operator must interact with the robot above the F/T sensor to prevent the misclassification of his interaction force \mathbf{f}_{ope} as the environment contact force \mathbf{f}_{env} .

VI. RESULTS

In this section, we begin by presenting and discussing the results of training the A3C algorithm. Subsequently, we provide an overview of the two experimental tests that were carried out to validate the entire pipeline. The first experiment centers around a collaborative peg-in-hole assembly, while the second one is focused on collaboratively polishing a wooden board. Both of these experiments are showcased in the accompanying video⁶ associated with this paper.

A. Training results

In order to train the A3C algorithm, which is utilized by the adaptation module to dynamically adjust QP priorities and select the desired contact force in real-time, we conducted simulations over a total of $1e6$ time steps using four parallel virtual environments. We followed the approach described in the algorithm version mentioned in [22], where the four parallel agents each complete their own segments of experience before updating the global network. The training process took approximately one hour, and the resulting learning curve, which represents the reward averaged over the last 100 time steps across all four agents, is depicted in Figure 4. This curve demonstrates the convergence of the algorithm after an initial exploration phase. Figure 5 presents the action a_k returned by the algorithm in response to the input state s_k at the end of the training. Each subplot visualizes one component of the action within the state space. The combination of these components describes the behavior of the adaptation module in various situations. In the edge characterized by a large operator force $f_{ope}^{\mathcal{L}}$ and a small contact force with the fixed environment $f_{env}^{\mathcal{L}}$, the AC task priority $w_2^{\mathcal{L}}$ tends to increase, while the priorities of other tasks decrease. The influence of the position error $e_x^{\mathcal{L}}$ on this transition is relatively minor. Conversely, in the opposite edge, where $f_{ope}^{\mathcal{L}}$ is low, and $f_{env}^{\mathcal{L}}$ is high, the priority for the FC control task and the value of the desired contact

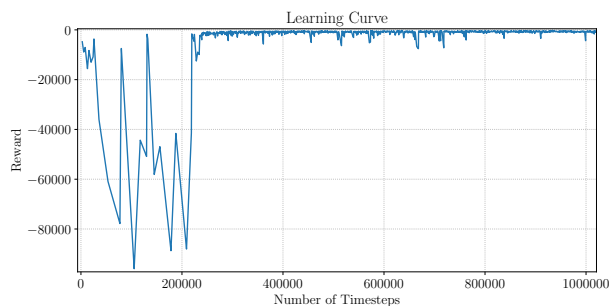


Fig. 4: Learning curve displaying the mean reward over the last 100 time steps during the training of the A3C agent.

force $f_d^{\mathcal{L}}$ increase. The IK task priority only increases when both $f_{ope}^{\mathcal{L}}$ and $f_{env}^{\mathcal{L}}$ are absent.

B. Experimental validation

The performance of the proposed A3CQP controller has been evaluated in two collaborative industrial scenarios: a peg-in-hole assembly and the polishing of a wooden plate. These comprehensive jobs engage all the functionalities the controller can manage. They involve stages where the robot needs to precisely follow a predefined Cartesian position $\mathbf{x}_{d,k}$ generated by a trajectory planner, phases where the robot must establish a robust physical connection with the workpiece or the worksurface to accomplish the task effectively, and intervals where the operator directly interacts with the robot to physically guide its movements in the workspace.

1) *Peg-in-hole experiment:* The robot starts from an initial *home* position, and the controller guides it to an *approach* pose, aligning with the hole. When the alignment is achieved within a specified tolerance, the reference position $\mathbf{x}_{d,k}$ is adjusted again to begin the peg insertion going to the *contact* pose. The activity of the adaptation module and the dynamics of the peg-hole misalignment are illustrated in Figure 6. Three peg insertions were performed during the experiment. Owing to the relatively lenient alignment tolerances enforced during

⁶<https://youtu.be/2w-ZQ8-GUtc>

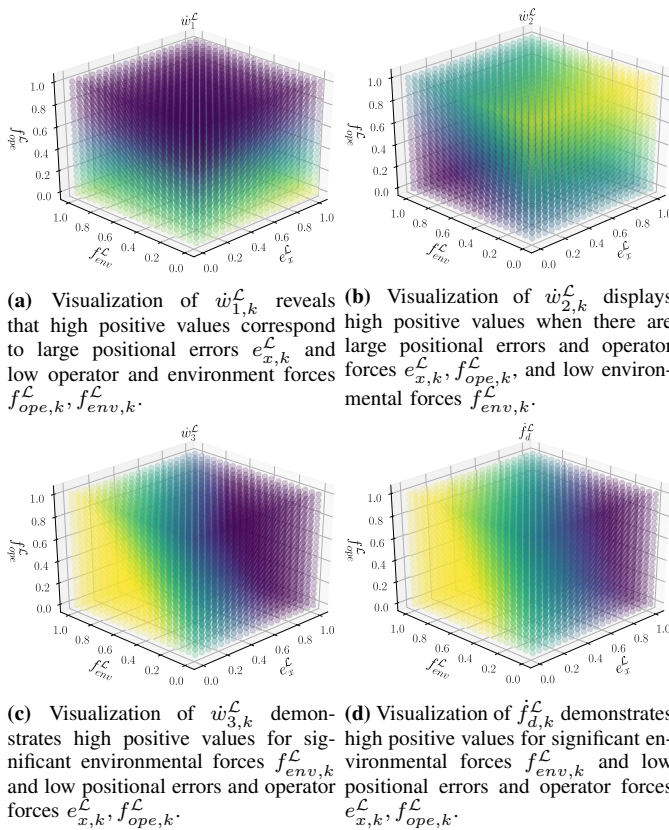


Fig. 5: Plot of the four action components $a_k = \{\dot{w}_{1,k}^{\mathcal{L}}, \dot{w}_{2,k}^{\mathcal{L}}, \dot{w}_{3,k}^{\mathcal{L}}, \dot{f}_{d,k}^{\mathcal{L}}\}$ according to the three component of the state $s_k = \{e_{x,k}^{\mathcal{L}}, f_{env,k}^{\mathcal{L}}, f_{ope,k}^{\mathcal{L}}\}$. In this visualization, the color yellow indicates high positive rate of change values near 1, while the dark blue color represents high negative rate of change values near -1.

the IK task phase, the peg typically initially enters into the FC task along the z-axis while in contact with the external hole border. These moments correspond to time instants at 7.7s, 41.0s, and 96.9s, during which the misalignment error primarily manifests as translation error. These moments are depicted in snapshot (A) of Figure 6. The transition from IK to the FC task along the z-axis is triggered by the detection of an initial minor environmental force. This detection prompts the adaptation module to simultaneously increase both $w_3^{\mathcal{L},z}$ and $f_d^{\mathcal{L},z}$ while reducing $w_1^{\mathcal{L},z}$. These dynamics are visually represented in Figures 5c, 5d, and 5a. This phase results in a sudden, slight deviation in $e_x^{\mathcal{L},z}$ as the system ceases to track the Cartesian reference in that particular direction. Notably, following the initial minor force $f_{env}^{\mathcal{L},z}$ measured after the first contact, there is a significant increase in $f_{env}^{\mathcal{L},z}$ to match the desired $f_d^{\mathcal{L},z}$. This leads to noticeable steps in the plot. At this point, the synergy between the FC task in the z-direction, exerting downward pressure on the peg, and the IK task in the x and y directions, which continue to guide the peg toward the desired position in the x-y plane, allows the peg to slip into the hole when it intercepts it during the approach maneuvers. These maneuvers are accompanied by force shocks measured in both $f_{env}^{\mathcal{L},z}$ and $f_{ope}^{\mathcal{L},z}$ but do not result in any perceivable change in the task weights, indicating that the controller

dynamics remain robust against these disturbances. When the peg intercepts the hole at time instants 12.2s, 45.5s, and 101.5s, there is an increase in misalignment due to the rotation caused by the reaction forces of the hole walls. You can see this particular moment in snapshot (B) of Figure 6. In this situation, the downward force exerted by the FC task guides the peg within the hole. As a result, at time instants 15.9s, 49.3s, and 105.9s, the misalignment is nullified, demonstrating the successful insertion of the peg. This moment is captured in snapshot (C). The large $e_x^{\mathcal{L},z}$ value registered at 26.2s is a result of the robot return to the *home* position after the initial insertion. During the subsequent two insertions, the operator manually disengages the peg from the hole at 54.4s and 112.6s. This leads to a noticeable presence of a consistent operator force $f_{ope}^{\mathcal{L},z}$, causing a subsequent increase in $w_2^{\mathcal{L},z}$ in line with the expected agent behavior represented in Figure 5b. This transition, represented by snapshot (D), results in a sharp increase in misalignment and position error $e_x^{\mathcal{L},z}$ since the reference \mathbf{x}_d and the hole placement remain unchanged. Following the decrease in $f_{ope}^{\mathcal{L},z}$, $w_1^{\mathcal{L},z}$, driven by the objective to reduce $e_x^{\mathcal{L},z}$, takes the lead again at time instants 75.1s and 124.4s, allowing the manipulator to smoothly return to the *home* position. This moment is depicted in snapshot (E). Thus, the adoption of the proposed A3CQP controller in the chosen peg-in-hole assembly facilitates the successful and autonomous execution of the job. This effectiveness holds true even when dealing with uncertainties in the hole position, thanks to the controller adept utilization of the FC task, which helps eliminate peg-hole misalignment. An essential feature of this system is its adaptability to operator intervention at any point during the operation. The controller seamlessly transitions to accommodate the operator's commands, even when the robot is already in contact with the workpiece.

2) *Polishing experiment:* The robot starts from an initial *rest* position, with the tool not yet in contact with the wooden plate intended for polishing. After a brief interval in which the system velocity is maintained null, the A3CQP controller directs the robot to make contact with the work surface, moving towards the *polishing* pose. There, the planar translation components of the desired Cartesian position \mathbf{x}_d , specifically x_d^x and x_d^y , are adjusted to trace a circular trajectory on the work surface. Simultaneously, the Arduino controller activates the drill motor to spin at 50 rpm. The actions of the adaptation module and the behavior of the desired reference position are illustrated in Figure 7. The robot makes contact with the wooden plate three times to polish specific areas until the operator intervenes to change the working zone. As shown earlier, when the agent senses the initial contact force with the plate, it triggers the FC task in the z Cartesian direction. This results in three distinct steps in the environment contact force $f_{env}^{\mathcal{L},z}$ at time instants of 13.6s, 43.3s, and 79.2s. In this phase, the desired reference position is continuously adjusted to follow a circular trajectory, corresponding to sinusoidal references for the coordinates x_d^x and x_d^y . Unlike before, due to the periodic motion in the x-y plane, the $f_{env}^{\mathcal{L},z}$ no longer remains mainly flat during the contact phase. Instead, it exhibits sinusoidal oscillations, which are also visible in the position error $e_x^{\mathcal{L},z}$ and introduce measurement noise in the

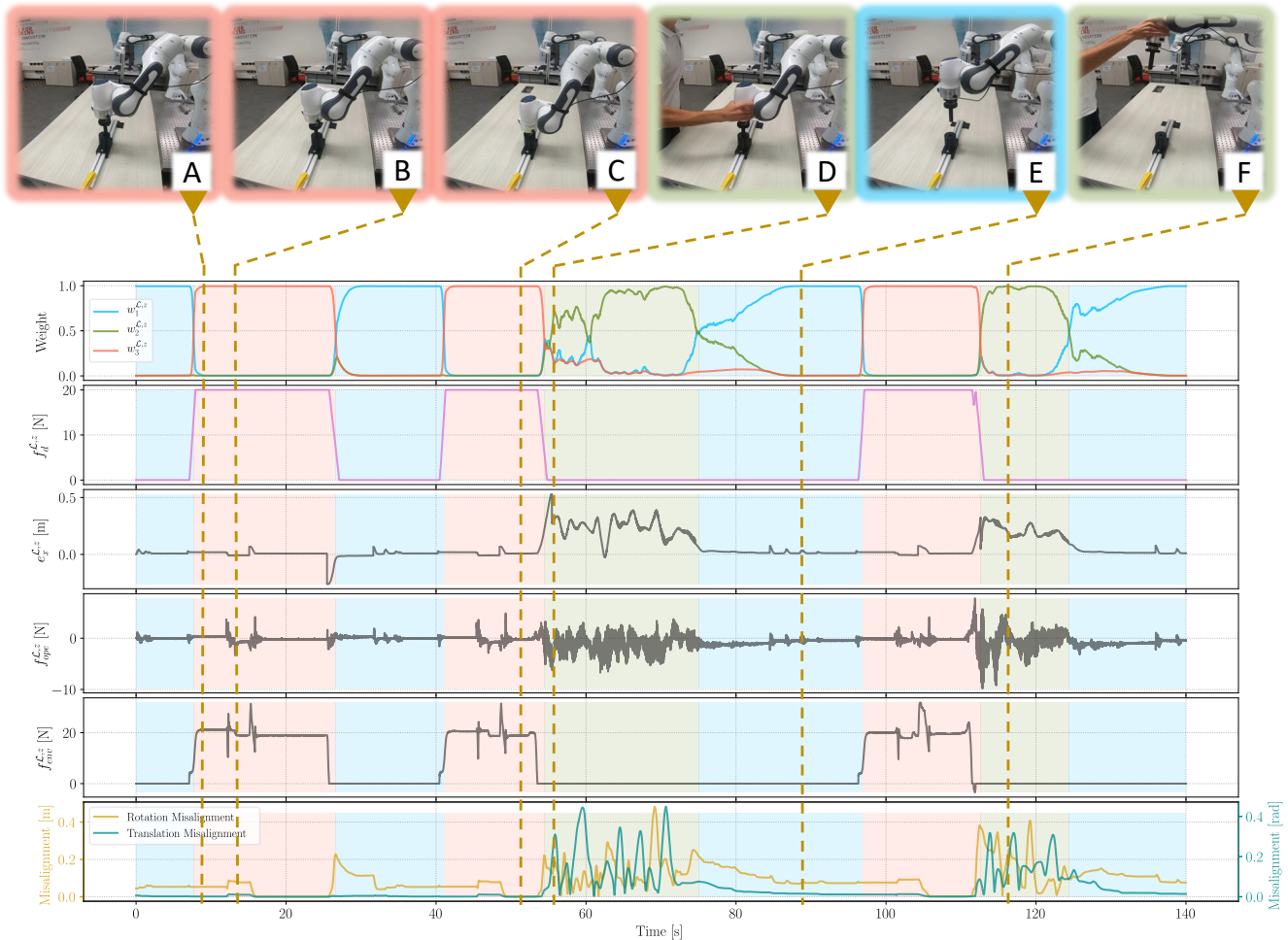


Fig. 6: Peg-in-hole experiment results. The top two plots display the action a^z determined by the adaptation module in the z Cartesian direction during the peg-in-hole experiment. The latter is based on the input state s^z , which is depicted in the subsequent three plots. The bottom plot illustrates the dynamics of the peg-hole translation and rotation misalignment. Color-coded patches are employed to highlight the primary controller task: blue signifies the IK task phases, red corresponds to the FC task phases, and green represents the AC task phases. Towards the top of the image, you'll find six snapshots captured during the experiment. The first three snapshots capture the different phases of the FC task. In (A), you can see the initial contact of the peg with the external hole border. Moving on to (B), it illustrates the peg successfully intercepting the hole while maintaining contact. Finally, (C) displays the peg being inserted into the hole. Snapshot (D) documents the transition from the FC task to the AC task as the operator manually disengages the peg-hole assembly. After the interaction with the operator in (E), the robot returns to its "approach" pose. Snapshot (F) represents a moment of inspection captured during the final collaborative interaction.

operator interaction force $f_{ope}^{L,z}$. However, despite these movements, the manipulator, leveraging the FC task, maintains the polishing head aligned with the plate and exerts the necessary force for the polishing task. When the operator intervenes, indicated by the sudden rise in $f_{ope}^{L,z}$ at time instants 29.7s, 62.2s, and 101.5s, the transition to the AC task phase occurs. In these moments, the operator grabs the manipulator end effector, physically guides it to a new working region, and then releases it. During this interval, the planar references x_d^x and x_d^y change to align with the new positions the robot assumes under the operator's guidance. Once released, at time instants 37.2s, 68.5s, and 107.3s, the robot reactivates the IK task to stabilize the last position taught by the operator. After a brief period of maintaining this position at zero velocity, the robot is prepared to return to the wooden plate and recommence

the polishing activity from the new position. The snapshots of the experiment, as illustrated in Figure 7, document the various phases of the polishing process. At first, the robot concentrates on the plate's central area, as captured in (B). It is then repositioned to the right by the operator, depicted in snapshots (C) and (D). Subsequently, the robot is shifted to the left, as seen in snapshots (E) and (F), where it concludes the polishing task. In conclusion, the presented A3CQP controller isn't limited to a single category of jobs. Instead, it equips the robot with the flexibility to adapt to a diverse range of activities. These activities may require autonomous tasks involving interaction with the environment, and the operator is free to step in and collaborate with the robot. In this case, the operator indicates which areas of the plate require attention.

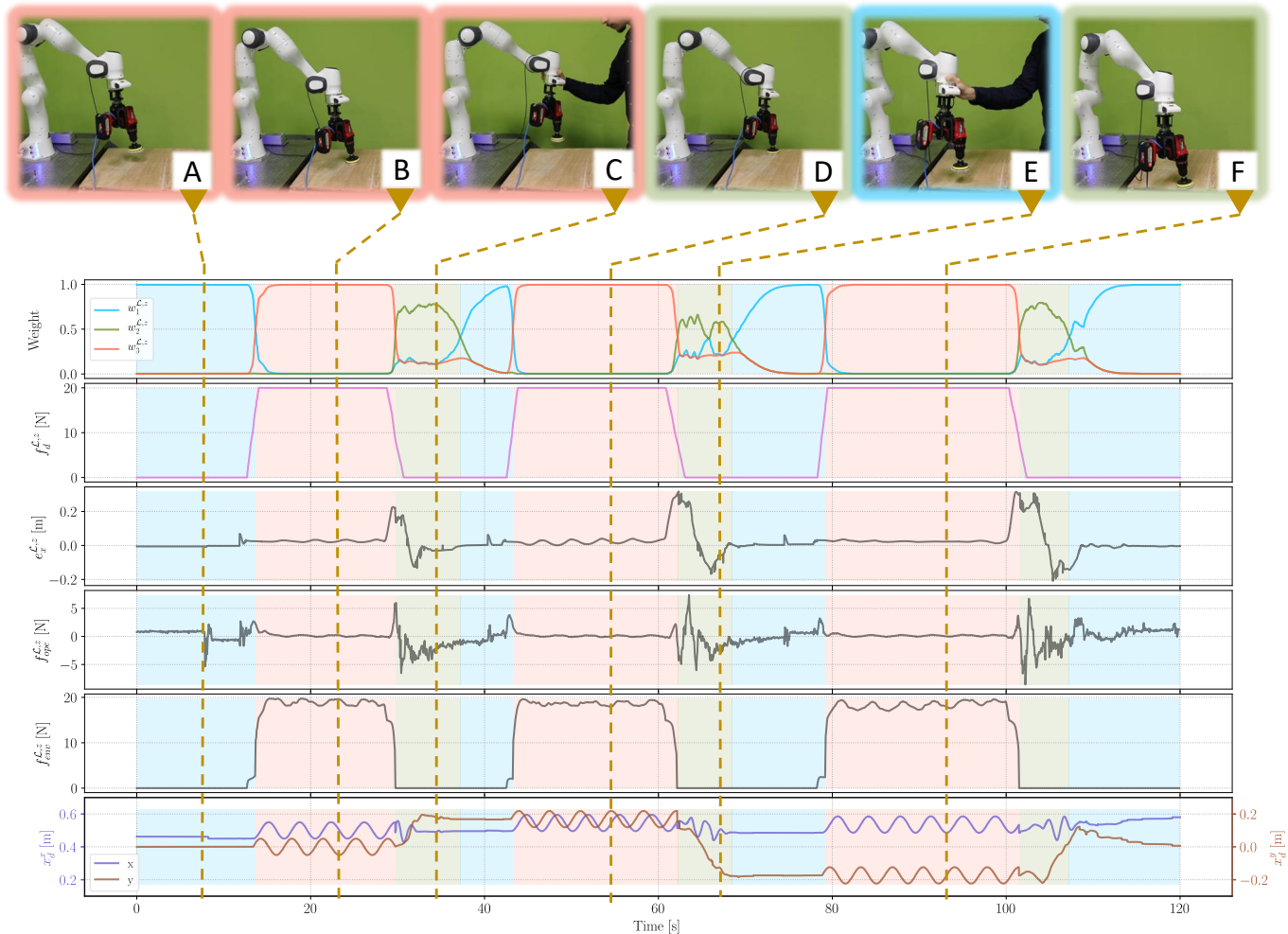


Fig. 7: Polishing experiment results. The top two plots showcase the action a^z determined by the adaptation module in the z Cartesian direction throughout the polishing experiment. This action is based on the input state s^z , visualized in the following three plots. The bottom plot illustrates the behavior of the desired reference position in the Cartesian x - y plane, specifically x_d^x and x_d^y . These reference positions describe circles whose centers are modified in response to the operator's intervention. To highlight the prevailing controller task at any given time, color-coded patches are utilized. Just as before, the color blue denotes the IK task phases, red represents the FC task phases, and green corresponds to the AC task phases. In the upper part of the image, you'll find six snapshots captured during the experiment. The first snapshot (A) portrays the robot in its initial *rest* position before making contact with the plate. The second snapshot (B) illustrates the first phase of polishing, with the robot concentrating on the central part of the plate. Following some moments of work, (C) shows the operator shifting the end effector to the right, with the robot continuing its work in position (D). In (E), the operator opts to move the robot to the left, allowing it to complete the polishing on that side of the plate (F).

VII. CONCLUSION

In this work, we have devised an innovative A3CQP controller aimed at enhancing the robot adaptability when tackling complex, unstructured tasks. This controller is grounded in a non-strict hierarchical Quadratic Programming algorithm, seamlessly incorporating both motion and interaction capabilities. It ensures that control inputs remain stable, consistent, and predictable while adhering to specific constraints. What distinguishes this controller is its ability to dynamically and autonomously adjust task priorities. It employs a Reinforcement Learning-based adaptation module, which leverages the Asynchronous Advantage Actor-Critic algorithm. This approach ensures swift convergence and steady training within the continuous action and observation spaces. This adaptation

module plays a pivotal role in endowing the robot with the ability to learn from its interactions with the environment. It empowers the robot to smoothly transition between three essential tasks: the Cartesian Inverse Kinematic Task, Cartesian Admittance Control Task, and Cartesian Force Control Task. These tasks span a wide range of behaviors, from precise position tracking for manipulation to exerting specific forces during physical interactions and following operator guidance for co-manipulation. To assess the validity of our proposed solution, we conducted a series of experiments. These tests encompassed both collaborative peg-in-hole assembly and the cooperative polishing of a wooden plate, employing the Franka Emika Panda robot. The outcomes of these evaluations demonstrate the robot adeptness in combining a variety of tasks to fulfill the specific needs of different job phases.

These activities necessitate autonomous operations involving interactions with the environment, and the operator retains the flexibility to intervene and collaborate with the robot as needed.

REFERENCES

- [1] 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.
- [2] Y. Abe, M. Da Silva, and J. Popović, "Multiobjective control with frictional contacts," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2007, pp. 249–258.
- [3] G. Raiola, C. A. Cardenas, T. S. Tadele, T. De Vries, and S. Stramigioli, "Development of a safety-and energy-aware impedance controller for collaborative robots," *IEEE Robotics and automation letters*, vol. 3, no. 2, pp. 1237–1244, 2018.
- [4] J. Salini, V. Padois, and P. Bidaud, "Synthesis of complex humanoid whole-body behavior: A focus on sequencing and tasks transitions," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1283–1290.
- [5] M. Liu, Y. Tan, and V. Padois, "Generalized hierarchical control," *Autonomous Robots*, vol. 40, pp. 17–31, 2016.
- [6] V. Modugno, G. Neumann, E. Rueckert, G. Oriolo, J. Peters, and S. Ivaldi, "Learning soft task priorities for control of redundant robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 221–226.
- [7] L. Penco, E. M. Hoffman, V. Modugno, W. Gomes, J.-B. Mouret, and S. Ivaldi, "Learning robust task priorities and gains for control of redundant robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2626–2633, 2020.
- [8] J. Silvério, S. Calinon, L. Rozo, and D. G. Caldwell, "Learning task priorities from demonstrations," *IEEE Transactions on Robotics*, vol. 35, no. 1, pp. 78–94, 2018.
- [9] L. Roveda, A. Testa, A. A. Shahid, F. Braghin, and D. Piga, "Q-learning-based model predictive variable impedance control for physical human-robot collaboration," *Artificial Intelligence*, vol. 312, p. 103771, 2022.
- [10] N. Dehio, R. F. Reinhart, and J. J. Steil, "Multiple task optimization with a mixture of controllers for motion generation," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 6416–6421.
- [11] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.
- [12] A. Ajoudani, A. M. Zanchettin, S. Ivaldi, A. Albu-Schäffer, K. Kosuge, and O. Khatib, "Progress and prospects of the human–robot collaboration," *Autonomous Robots*, vol. 42, pp. 957–975, 2018.
- [13] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.
- [14] E. Mingo Hoffman, A. Laurenzi, L. Muratore, N. G. Tsagarakis, and D. G. Caldwell, "Multi-priority cartesian impedance control based on quadratic programming optimization," in *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 2018, pp. 309–315.
- [15] N. Dehio and J. J. Steil, "Dynamically-consistent generalized hierarchical control," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1141–1147.
- [16] F. Tassi and A. Ajoudani, "Multi-modal and adaptive control of human-robot interaction through hierarchical quadratic programming," 2023.
- [17] R. Lober, V. Padois, and O. Sigaud, "Variance modulated task prioritization in whole-body control," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 3944–3949.
- [18] S. Nambiar, A. Wiberg, and M. Tarkian, "Automation of unstructured production environment by applying reinforcement learning," *Frontiers in Manufacturing Technology*, vol. 3, p. 1154263, 2023.
- [19] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.
- [20] T. Degris, P. M. Pilarski, and R. S. Sutton, "Model-free reinforcement learning with continuous action in practice," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 2177–2182.
- [21] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.
- [22] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763*, 2016.
- [23] B. Siciliano, O. Khatib, and T. Kröger, *Springer handbook of robotics*. Springer, 2008, vol. 200.
- [24] S. Caron, Q.-C. Pham, and Y. Nakamura, "Stability of surface contacts for humanoid robots: Closed-form formulae of the contact wrench cone for rectangular support areas," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5107–5112.
- [25] A. Testa, L. Raiano, M. Laghi, A. Ajoudani, and E. M. Hoffman, "Joint position bounds in resolved-acceleration control: a comparison," in *Workshop on Human-Friendly Robotics (HFR 2023)*, 2023.
- [26] J. Sola, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018.
- [27] 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, 1987.
- [28] H. W. Kuhn and A. W. Tucker, "Nonlinear programming," in *Traces and emergence of nonlinear programming*. Springer, 2013, pp. 247–258.
- [29] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
- [30] T. Tieleman, G. Hinton, *et al.*, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
- [31] N. M. Aszemi and P. Dominic, "Hyperparameter optimization in convolutional neural network using genetic algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 6, 2019.
- [32] H. Alibrahim and S. A. Ludwig, "Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 1551–1559.