Learning Force Control for Contact-Rich Manipulation Tasks With Rigid Position-Controlled Robots

Cristian Camilo Beltran-Hernandez, Damien Petit, Ixchel Georgina Ramirez-Alpizar, Takayuki Nishi, Shinichi Kikuchi, Takamitsu Matsubara, and Kensuke Harada

***Abstract*—Reinforcement Learning (RL) methods have been proven successful in solving manipulation tasks autonomously. However, RL is still not widely adopted on real robotic systems because working with real hardware entails additional challenges, especially when using rigid position-controlled manipulators. These challenges include the need for a robust controller to avoid undesired behavior, that risk damaging the robot and its environment, and constant supervision from a human operator. The main contributions of this work are, first, we proposed a learning-based force control framework combining RL techniques with traditional force control. Within said control scheme, we implemented two different conventional approaches to achieve force control with position-controlled robots; one is a modified parallel position/force control, and the other is an admittance control. Secondly, we empirically study both control schemes when used as the action space of the RL agent. Thirdly, we developed a fail-safe mechanism for safely training an RL agent on manipulation tasks using a real rigid robot manipulator. The proposed methods are validated both on simulation and a real robot with an UR3 e-series robotic arm.**

***Index Terms*—Reinforcement learning, force control, compliance and impedance control, industrial robots.**

I. Introduction

In THE age of the 4th industrial revolution, there is much interest in applying artificial intelligence to automate industrial manufacturing processes. Robotics, in particular, holds the promise of helping to automate processes by performing complex manipulation tasks. Nevertheless, safely solving complex manipulation tasks in an unstructured environment using robots is still an open problem [1].

Reinforcement learning (RL) methods have been proven successful in solving manipulation tasks by learning complex behaviors autonomously in a variety of tasks such as grasping [2], [3], pick-and-place [4], and assembly [5]. While there are some instances of RL research validated on real robotic systems, most works are still confined to simulated environments due to the additional challenges presented by working on real hardware, especially when using rigid position-controlled robots. These challenges include the need for a robust controller to avoid undesired behavior that risk collision with the environment, and constant supervision from a human operator.

So far, when using real robotic systems with RL, there are two common approaches. The first approach consists of learning high-level control policies of the manipulator. Said approach assumes the existence of a low-level controller that can solve the RL agent's high-level commands. Some examples include agents that learn to grasp [2], [3] or to throw objects [6]. In said cases, the agent learns high-level policies, e.g., learns the position of the target object and the grasping pose, while a low-level controller, such as a motion planner, directly controls the manipulator's joints or end-effector position. Nevertheless, the low-level controller is not always available or easy to manually engineer for each task, especially for achieving contact-rich manipulation tasks with a position-controlled robot. The second approach is to learn low-level control policies using soft robots [7]–[9], manipulators with joint torque control or flexible joints, which are considerably safer to work with due to their compliant nature, particularly in the case of allowing an RL agent to explore its surroundings where collisions with the environment may be unavoidable. Our main concern with this approach is that most industrial robot manipulators are, by contrast, rigid robots (position-controlled manipulators). Rigid robots usually run on position control, which works well for contact-free tasks, such as robotic welding, or spray-painting [10]. However, they are inherently unsuitable for contact-rich manipulation tasks since any contact with the environment would be considered as a disturbance by the controller, which would generate a collision with a large contact force. Force control methods [11] can be used to enable the rigid manipulator to perform tasks that

Manuscript received February 24, 2020; accepted July 4, 2020. Date of publication July 21, 2020; date of current version July 28, 2020. This letter was recommended for publication by Associate Editor Z. Doulgeri and Editor P. Rocco upon evaluation of the reviewers' comments. *(Corresponding author: Cristian Camilo Beltran-Hernandez.)*

Cristian Camilo Beltran-Hernandez and Damien Petit are with the Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, Osaka 5600045, Japan (e-mail: cristianbehe@gmail.com; damien.gerard.petit@gmail.com).

Ixchel Georgina Ramirez-Alpizar and Kensuke Harada are with the Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, Osaka 5600045, Japan, also with the Automation Research Team, Industrial CPS Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo 135-0064, Japan (e-mail: ixchel-ramirezalpizar@aist.go.jp; harada@sys.es.osaka-u.ac.jp).

Takayuki Nishi and Shinichi Kikuchi are with the Process Engineering & Technology Center, Research & Development Management Headquarters, FUJIFILM Corporation, Tokyo 107-0052, Japan (e-mail: takayuki.nishi@fujifilm.com; shinichi.kikuchi@fujifilm.com).

Takamitsu Matsubara is with the Robot Learning Laboratory, Institute for Research Initiatives, Nara Institute of Science and Technology (NAIST), Ikoma 630-0192, Japan (e-mail: takam-m@is.naist.jp).

This article has supplementary downloadable material available at https://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2020.3010739

require contact with the environment, though the controller's parameters need to be properly tuned, which is still a challenging task. Therefore, we propose a method to safely learn low-level force control policies with RL on a position-controlled robot manipulator.

This letter presents three main contributions. First, a control framework for learning low-level force control policies combining RL techniques with traditional force control. Within said control scheme, we implemented two different conventional force control approaches with position-controlled robots; one is a modified parallel position/force control, and the other is an admittance control. Secondly, we empirically study both control schemes when used as the action space of the RL agent. Thirdly, we developed a fail-safe mechanism for safely training an RL agent on manipulation tasks using a real rigid robot manipulator. The proposed methods are validated on simulation and real hardware using a UR3 e-series robotic arm.

II. Related Work

A. Force Control

Force control methods address the problem of interaction between a robot manipulator and its environment, even in the presence of some uncertainty (geometric and dynamic constraints) on contact-rich tasks [12], [13]. These methods provide direct control of the interaction through contact force feedback and a set of parameters, which describe the dynamic interaction between the manipulator and the environment. However, prior knowledge of the environment is necessary to properly define the controller's parameters at each phase of the task, such as stiffness. Existing methods address said problem by either scheduling variable gains [14], using adaptive methods for setting the gains [15], or learning the gains from demonstrations [16]. Instead, we propose to directly learn the time-variant force control gains from experience by interacting and observing the environment.

B. Reinforcement Learning and Force Control

Previous research has also studied the use of RL methods to learn force control gains. Buchli *et al.* [17] uses policy improvements with path integrals (PI2) [18] to refine initial motion trajectories and learn variable scheduling for the joint impedance parameters. Similarly, Bogdanovic *et al.* [19], proposed a variable impedance control in joint-space, where the gains are learned with Deep Deterministic Policy Gradient (DDPG) [20]. Likewise, Martín-Martín *et al.* [21], proposed a variable impedance control in end-effector space (VICES).

However, in all these cases, access to the robot manipulator's low-level control of joint torques is assumed, which is not available for most industrial manipulators. Instead, we focus on position-controlled robot manipulators and provide a method to learn manipulation tasks using force feedback control where the controller gains are learned through RL methods. Luo *et al.* [22] propose a method for achieving peg-in-hole tasks on a deformable surface using RL and validated their approach on a position-controlled robot. They propose learning the motion trajectory based on the contact force information. However, the tuning of the compliant controller's parameters is not taken into account. We are proposing a method for learning not only the motion trajectory based on force feedback but simultaneously fine-tuning the compliant controller's parameters.

Additionally, both Bogdanovic [19] and Martín-Martín [21] study the importance of different action representation in RL for contact-rich robot manipulation tasks. We similarly provide an empirical study comparing different choices of action space based on force feedback control methods for rigid robots on contact-rich manipulation tasks.

C. Learning With Real-World Manipulators

Some research projects have explored the capabilities of RL methods on real robots by testing them on a large scale, such as Levine *et al.* [3] and Pinto *et al.* [23], both in which a massive amount of data was collected for learning robotic grasping tasks. However, in both works, a high-level objective, grasp posture, is learned from the experience obtained. In contrast, contact-rich tasks require learning direct low-level control to, for example, reduce contact force for safety reasons. On the other hand, Mahmood *et al.* [24] propose a benchmark for learning policies on real-world robots, so different RL algorithms can be evaluated on a variety of tasks. Nevertheless, the tasks available in [24] are either locomotion tasks with a mobile robot or contact-free tasks with a robot manipulator. In this work, we propose a framework for learning contact-rich manipulation tasks with real-world robot manipulators based on force control methods.

III. Methodology

The present study deals with high precision assembly tasks with a position-controlled industrial robot. Due to the difficulty of obtaining a precise model of the physical interaction between the robot and its environment, RL is used to learn both the motion trajectory and the optimal parameters of a compliant controller. The RL problem is described in Section III-A. The architecture of the system and the interaction control methods considered are explained in Section III-B1, Section III-B2, and Section III-C. Finally, our safety mechanism that allows the robot to learn unsupervised is described in Section III-D.

A. Reinforcement Learning

Robotic reinforcement learning is a control problem where a robot, the agent, acts in a stochastic environment by sequentially choosing actions over a sequence of time steps. The goal is to maximize a cumulative reward. Said problem is modeled as a Markov Decision Process. The environment is described by a state $\mathbf{s} \in \mathcal{S}$. The agent can perform actions $\mathbf{a} \in A$, and perceives the environment through observations $\mathbf{o} \in O$, which may or not be equal to $\mathbf{s}$. We consider an episodic interaction of finite time steps with a limit of T time steps per episode. The agent's goal is to find a policy $\pi(\mathbf{a}(t) \,|\, \mathbf{o}(t))$ that selects actions $\mathbf{a}(t)$ conditioned on the observations $\mathbf{o}(t)$ to control the dynamical system. Given an stochastic dynamics $p(\mathbf{s}(t+1) \,|\, \mathbf{s}(t), \mathbf{a}(t))$ and a reward function $r(\mathbf{s}, \mathbf{a})$, the aim is to find a policy $\pi*$ that maximizes the expected sum of future rewards given by $R(t) = \sum_i^{\infty} \gamma r(s(t), a(t))$ with γ being a discount factor [25].

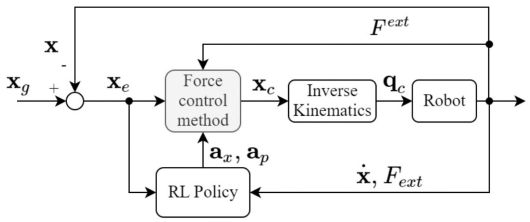

Fig. 1. Proposed learning force control scheme. The input to the system is a goal end-effector pose, $\mathbf{x}_g$. The policy actions are trajectory commands, $\mathbf{a}_x$, and parameters, $\mathbf{a}_p$, of a force controller.

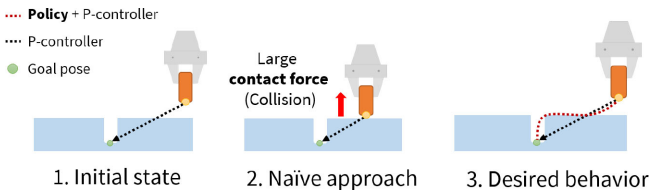

Fig. 2. Proposed approach to solve contact-rich tasks. Assuming knowledge of the goal pose of the robot's end-effector, a simple P-controller can be designed. Our approach aims to combine this knowledge with the policy to generate the motion trajectory.

Soft-Actor-Critic: We use the state-of-the-art model-free RL method called Soft-Actor-Critic (SAC) [26]. SAC is an off-policy actor-critic deep RL algorithm based on the maximum entropy reinforcement learning framework. SAC aims to maximize the expected reward while optimizing a maximum entropy. The SAC agent optimizes a maximum entropy objective, which encourages exploration according to a temperature parameter α. The core idea of this method is to succeed at the task while acting as randomly as possible. Since SAC is an off-policy algorithm, it can use a replay buffer to reuse information from recent rollouts for sample-efficient training. We use the SAC implementation from TF2RL.[1]

B. System Overview

Our proposed method aims to combine a force control with RL to learn contact-rich tasks when using position-controlled robots. Fig. 1 describes the proposed control scheme combining an RL policy and a force control method. We assume knowledge of the goal pose of the robot's end-effector, $\mathbf{x}_g$. Both the policy and the force controller receive as feedback the pose error, $\mathbf{x}_e = \mathbf{x}_g - \mathbf{x}$, and the contact force F_{ext}. The velocity of the end-effector, $\dot{\mathbf{x}}$, is also included in the policy's observations. The F/T sensor signal is filtered using a simple low-pass filter.

The force control method has two internal controllers. First, a PD controller that generates part of the motion trajectory based on the pose error, $\mathbf{x}_e$. Second, a force feedback controller that alters the motion trajectory according to the perceived contact force, F_{ext}.

The RL policy has two objectives. First, to generate a motion trajectory, $\mathbf{a}_x$. Fig. 2, shows how a simple P-controller (from the force control method) would not be enough to solve the task without producing a collision with the environment. For most cases, the P-controller trajectory would just attempt to penetrate the environment, since knowledge of the environment's geometry is not assumed. Nevertheless, the P-controller trajectory is good enough to speed up the agent's learning since it is already driven towards the goal pose. Therefore, to achieve the desired behavior, the nominal trajectory of the robot is the combination of the P-controller trajectory with the policy's trajectory. The second objective of the policy is to fine-tune the force control methods parameters, $\mathbf{a}_p$, to minimize the contact force when it occurs. We defined a collision as exceeding a maximum contact force in any direction. Therefore, contact with the environment is acceptable, but the policy's second goal is to avoid collisions. The policy also controls the P-controller's gains; thus, the policy decides how much to rely on the P-controller trajectory.

1) Pose Control Representation: The pose of the robot's end-effector is given by $\mathbf{x} = [\mathbf{p}, \phi]$, where $\mathbf{p} \in \mathbb{R}^3$ is the position vector and $\phi \in \mathbb{R}^4$ is the orientation vector. The orientation vector is described using Euler parameters (unit quaternions) denoted as $\phi = \{\eta, \varepsilon\}$; where $\eta \in \mathbb{R}$ is the scalar part of the quaternion and $\varepsilon \in \mathbb{R}^3$ the vector part. Using unit quaternions allows the definition of a proper orientation error for control purposes with a fast computation compared to using rotation matrices [27].

The position command from the force controller is $\mathbf{x}_c = [p_t, \phi_t]$, where p_t is the commanded translation, and ϕ_t is the commanded orientation for the time step t. The desired joint configuration for the current time step, $\mathbf{q}_c$, is obtain from an Inverse Kinematics (IK) solver based on $\mathbf{x}_c$.

2) Learning Force Control: Two of the most common force control schemes are considered in these work, parallel position/force control [12] and admittance control [13]. The main drawback of said control schemes is the requirement to tune the parameters for each specific task properly. Changes in the environment (e.g., surface stiffness) may require a new set of parameters. Thus, we propose a self-tuning process using RL method.

The policy actions are $\mathbf{a} = [\mathbf{a}_x, \mathbf{a}_p]$, where $\mathbf{a}_x = [\mathbf{p}, \phi]$ are position/orientation commands, and $\mathbf{a}_p$ are controller's parameters. $\mathbf{a}_p$ is different and specific for each type of controller, see Section III-C1 and Section III-C2 for details. The policy has a control frequency of 20 Hz while the force controller has a control frequency of 500 Hz.

C. Force Control Implementation

1) PID Parallel Position/Force Control: Based on [12], we implemented a PID parallel position/force control with the addition of a selection matrix to define the degree of control of position and force over each direction, as shown in Fig. 3. The control law consists of a PD action on position, a PI action on force, a selection matrix and the policy position action, $\mathbf{a}_x$,

$$u = amp; S(K_p^x \mathbf{x}_e + K_d^x \dot{\mathbf{x}}_e) + \mathbf{a}_x + (I - S)\left(K_p^f F_{ext} + K_i^f \int F_{ext} dt\right) \quad (1)$$

[1]TF2RL: RL library using TensorFlow 2.0. https://github.com/keiohta/tf2rl

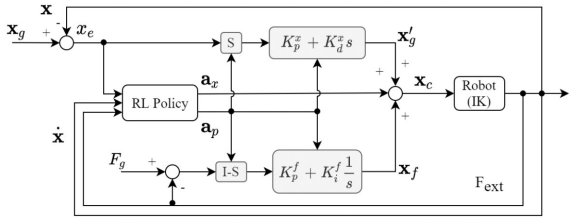

Fig. 3. Proposed scheme for learning PID parallel position/force control. The RL agent controls the controller parameters PD gains, PI gains, and the selection matrix, S.

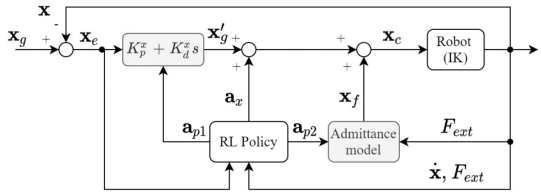

Fig. 4. Proposed scheme for learning admittance control. A PD controller is included to regulate the input reference motion trajectory. The RL agent controls the PD gains, as well as, the admittance model parameters (inertia, damping and stiffness).

where u is the vector of driving generalized forces. The selection matrix is

$$S = diag(s_1, \ldots, s_6), \quad s_j \in [0, 1]$$

where the values correspond to the degree of control that each controller has over a given direction.

Our parallel control scheme has a total of 30 parameters, 12 from the position PD controller's gains, 12 from the force PI controller's (PI) gains, and 6 from the selection matrix S. We reduced the number of controllable parameters to prevent unstable behavior and to reduce the system's complexity. For the PD controller, only the proportional gain, K_p^x, is controllable while the derivative gain, K_d^x, is computed based on the K_p^x. K_d^x is set to have a critically damped relationship as

$$K_d^x = 2\sqrt{K_p^x}$$

Similarly, for the PI controller, only the proportional gain, K_p^f, is controllable, the integral gain K_i^f is computed with respect to K_p^f. In our experiments, K_i^f was set empirically to be 1% of K_p^f. In total, 18 parameters are controllable. In summary, the policy actions regarding the parallel controller's parameters are $\mathbf{a}_p = [K_p^x, K_p^f, S]$.

To narrow the agents choices for the force control parameters, we follow a similar strategy as in [19]. Assuming we have access to some baseline gain values, P_{base}. We then define a range of potential values for each parameter as $[P_{\text{base}} - P_{\text{range}}, P_{\text{base}} + P_{\text{range}}]$ with the constant P_{range} defining the size of the range. We map the agent's actions $\mathbf{a}_p$ from the range $[-1, 1]$ to each parameter's range. P_{base} and P_{range} are hyperparameters of both controllers.

2) Admittance Control: is used to achieve a desired dynamic interaction between the manipulator and its environment. The admittance controller for position-controlled robots implemented is based on [28]. The admittance control is implemented on task-space instead of the robot joint-space. It follows the conventional control law

$$F_{ext} = m_d\ddot{x} + b_d\dot{x} + k_d x \tag{2}$$

where m_d, b_d, and k_d represent the desired inertia, damping, and stiffness matrices respectively. F_{ext} is the actual contact force vector. x, $\dot{x}$, $\ddot{x}$ are the displacement of the manipulator's end-effector, its velocity and acceleration respectively.

The admittance relationship can be expressed in Laplace-domain, adopting conventional expression of a second-order system as

$$\frac{X}{F}(s) = \frac{1/m_d}{s^2 + 2\zeta\omega_n s + \omega_n} \tag{3}$$

where ζ is the damping ratio and ω_n is the natural frequency, and they can be expressed by the admittance parameters as

$$\zeta = \frac{b_d}{2\sqrt{k_d\, m_d}} \qquad \omega_n = \sqrt{\frac{k_d}{m_d}} \tag{4}$$

We are proposing a variable admittance controller, where the inertia, damping, and stiffness parameters are learned by the RL agent. Additionally, a PD controller is included in our admittance control. The PD controller with the policy action, $\mathbf{a}_x$, generates the nominal trajectory as explain in Section III-B. The complete admittance control scheme is depicted in Fig. 4. The PD gains are also controlled by the policy at each time step.

For the admittance control scheme, there are a total of 30 parameters; 12 from the position PD controller's gains and 18 from the inertia, damping, and stiffness parameters. Similarly, as mentioned in Section III-C1, we reduced the number of controllable parameters to prevent unstable behavior of the robot and reduce the system's complexity. Following the same strategy described in Section III-C1, of the PD controller, only the proportional gain, K_p^x, is controllable. Additionally, we considered the inertia parameter for each direction as a constant, 0.1 kg·m^2 in all our experiments as a similar payload is used across tasks. Furthermore, we compute the damping with respect to the inertia parameter and the stiffness parameter by defining a constant damping ratio. From (4) we have that

$$b_d = 2\,\zeta\sqrt{k_d * m_d}$$

Therefore, only the stiffness parameters are controllable. In total, the controllable parameters of the admittance control are reduced to 12 parameters; 6 PD gains and 6 stiffness parameters. In summary, the policy actions regarding the admittance controller's parameters are $\mathbf{a}_p = [K_p^x, k_d]$.

D. Fail-Safe Mechanism

Most modern robot manipulators already include a layer of safety in the form of an emergency stop. Nonetheless, the emergency stop exists at the extreme ends of the robot limits and completely interrupts the entire training session if triggered. To reactivate the robot, a human operator is required. To alleviate this inconvenience, we propose a mechanism that allows the robot to operate within less extreme limits. Thus, training of

Algorithm 1: Safe Manipulation Learning.

Define joint velocity limit $\dot{\mathbf{q}}_{max}$
Define contact force limit F_{max}
Define initial state $\mathbf{x}_0$
Define goal state $\mathbf{x}_g$
for $n = 0, \cdots, N-1$ episodes **do**
  for $t = 0, \cdots, T-1$ steps **do**
    Get current contact force: F_{ext}
    $\mathbf{x}_e = \mathbf{x}_g - \mathbf{x}$
    Get Observation: $\mathbf{o} = [\mathbf{x}_e, \dot{\mathbf{x}}, F_{ext}]$
    Compute policy actions: $\pi_\theta(\mathbf{a}_x, \mathbf{a}_p | \mathbf{o})$
    $\mathbf{x}_c = \textit{control\_method}(\mathbf{x}_e, \mathbf{a}_x, \mathbf{a}_p, F_{ext})$
    $\mathbf{q}_c = \textit{IK\_solver}(\mathbf{x}_c)$
    if $\mathbf{q}_c$ not exists **then continue**
    if $|(\mathbf{q}_t - \mathbf{q}_c)/dt| > \dot{\mathbf{q}}_{max}$ **then continue**
    if $F_{ext} > F_{max}$ **then break**
    Actuate $\mathbf{q}_c$ on robot
  Reset to $\mathbf{x}_0$

an RL agent can be done directly on the position-controlled manipulator with minimal human supervision.

Our system controls the robot as if teleoperating it by providing a real-time stream of task-space motion commands for the robot to follow. Therefore, we added our safety layer between the streamed motion command and the robot's actual actuation. The fail-safe mechanism validates that the intended action is within a defined set of safety constraints. As shown in Algorithm 1, for each action we check whether an IK solution exists for the desired position command, $\mathbf{x}_c$, if so, whether the joint velocity required to achieve the IK solution, $\mathbf{q}_c$, is within the speed limit.

If any of these validations are not satisfied, the intended action is not executed on the robot, and the robot remains in its current state for the present time step. Finally, we check if the contact force at the robot's end-effector is within a defined range limit. If not, the episode ends immediately.

The first two validations are proactive and prevent unstable behaviors of the manipulator before they occur. In contrast, the third validation is reactive, i.e., only after a collision has occurred (the force limit has been violated), the robot is prevented from further actions.

E. Task's Reward Function

For all the manipulation tasks considered, the same reward function was used:

$$r(\mathbf{s}, \mathbf{a}) = w_1 L_m(\|\mathbf{x}_e / \mathbf{x}_{max}\|_{1,2}) + w_2 L_m(\|\mathbf{a} / \mathbf{a}_{max}\|_2) + w_3 L_m(\|F_{ext} / F_{max}\|_2) + w_4 \rho + w_5 \kappa \quad (5)$$

where $\mathbf{x}_{max}$, $\mathbf{a}_{max}$, and F_{max} are defined maximum values. $L_m(y) = y \mapsto x, x \in [1, 0]$ is a linear mapping to the range 1 to 0, thus, the closer to the goal and the lower the contact force, the higher the reward obtained. $||\cdot||_{1,2}$ is L1,2 norm based on [7]. The $\mathbf{x}_e$ is the distance between the manipulator's end-effector and the target goal at time step t. $\mathbf{a}$ is the action taken by the agent. F_{ext} is the contact force. ρ is a penalty given at each time step to encourage a fast completion of the task. κ is a reward defined as follows

$$\kappa = \begin{cases} 200, & \text{Task completed} \\ -10, & \text{Safety violation} \\ 0, & \text{Otherwise} \end{cases} \quad (6)$$

Finally, each component is weighted via w, all w's are hyperparameters.

TABLE I
POLICY MODELS WITH DIFFERENT ACTION SPACES

Control Scheme	Name	Pose	Gains		
			PD	PI / Stiffness	Selection Matrix S
		$\mathbf{a}_x$	$\mathbf{a}_p$		
Parallel	P-9	6	1	1	1
	P-14	6	1	1	6
	P-19	6	6	6	1
	P-24	6	6	6	6
Admittance	A-8	6	1	1	-
	A-13	6	1	6	-
	A-13pd	6	6	1	-
	A-18	6	6	6	-

IV. EXPERIMENTS

We propose a framework for safely learning manipulation tasks with position-controlled manipulators using RL. Two control schemes were implemented. With the following experiments, we seek to answer the following questions: Can a high-dimensional force controller be learned by the agent? Which action space, based on the number of adjustable controller's parameters provides the best learning performance?

A description of the materials used for the experiments is given in Section IV-A. An insertion task was used for evaluating the learning performance of the RL agents with the proposed method on a simulated environment, described in Section IV-B. Finally, the proposed method is validated on a real robot manipulator with high-precision assembly tasks.

A. Technical Details

Experimental validation was performed both in a simulated environment using the Gazebo simulator [29] version 9 and on real hardware using the Universal Robot 3 e-series, with a control frequency of up to 500 Hz. The robotic arm has a Force/Torque sensor mounted at its end-effector and a Robotiq Hand-e gripper. Training was performed on a computer with CPU Intel i9-9900 k, GPU Nvidia RTX-2800 Super.

B. Action Spaces for Learning Force Control

Each control scheme proposed in Section III has a number of controllable parameters. The curse of dimensionality is a well known problem in RL [25]. Controlling few dimensions, number of parameters, makes the task easier to learn at the cost of losing dexterity.

In the following experiment, several policy models were evaluated. Each model has a different action space, i.e., a different number of controllable parameters. We evaluate the learning performance of the models described in Table I, four models per control scheme. Each policy model has the same six parameters to control the position and orientation of the manipulator, $\mathbf{a}_x$, but a different number of parameters to tune the controller's gains,

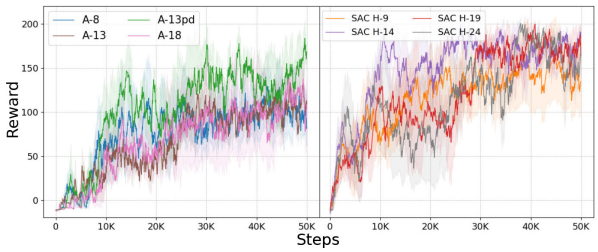

Fig. 5. Learning curve of training session with active penalization of violation of the safety constraints. Peg-insertion scenario on simulation.

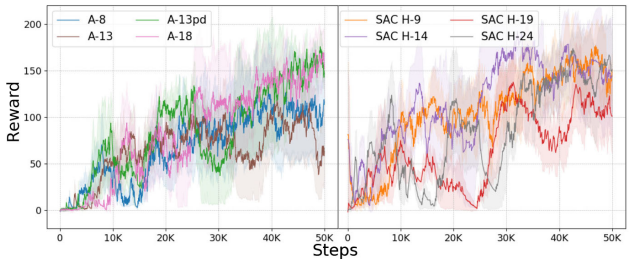

Fig. 6. Learning curve of training without penalizing violation of safety constraints on the reward function. Peg-insertion scenario on simulation.

$\mathbf{a}_p$. From now on, we refer to each model by the name given in Table I.

For a fair comparison, the action spaces were evaluated on a simulated peg-insertion environment so that we could guarantee the exact same initial conditions for each training session. The task is to insert a cube-shaped peg into a task board, where the hole has a clearance of 1 mm. Each policy model was trained for 50.000 (50 k) steps with a maximum of 150 steps per episode. The complete training session was repeated three times per model. Since the policy control frequency was set at 20 Hz, each episode lasts a maximum of 7.5 seconds. The episode ends if 1) the maximum number of time steps is reached, 2) a minimum distance error from the target pose is achieved, 3) or if a collision occurs. In general, a complete training session takes about 50 minutes, including reset times.

Results: The comparison of learning curves for each policy model evaluated is shown in Fig. 5. In the figure, the average cumulative reward per episode across the training sessions (bold line) is displayed along with the standard deviation error (shaded colored area). The results have been smoothed out using the exponential moving averages, with a 0.6 weight, to show the tendency of the learning curves.

From Fig. 5, the overall best performance is achieved with the policy models combined with the parallel control scheme. By the end of the training session, these families of policies can yield higher rewards than the policy models combined with the admittance control scheme.

For the parallel control scheme, the model with the worst performance is P-9; it can be seen that there is not enough control of the controller's parameters to learn a good policy consistently. On the other hand, the model P-24 has the slowest learning rate, but by the end of the training session, it can consistently learn a good policy. The policy model P-14 has the fastest learning rate and overall best performance.

For the admittance control scheme, the models A-13pd and A-18 have the best overall performance, with A-13pd yielding a cumulative reward as high as P-14 by the end of the training session. The model A-8, similar to P-9, has one of the worst performance; again, the lack of controllable parameters seems to have a big impact on learning a successful policy.

It is worth noting that for both control schemes, the models P-14 and A-13pd have the best overall performance. They provide the best trade-off between system complexity and learn-ability. On the other hand, the models with the largest number of parameters P-24 and A-18 can learn successful policies, but they require a longer training time to achieve it.

TABLE II
COLLISION DETECTED DURING TRAINING SESSION

Model	avg. # of collisions across training sessions		
	Penalization	No penalization	Difference
A-8	326	455	-39%
A-13	350	408	-16%
A-13pd	**300**	**462**	**-54%**
A-18	451	457	-1%
P-9	187	369	-98%
P-14	**121**	**206**	**-70%**
P-19	183	392	-115%
P-24	219	337	-43%

The parallel models' learning curve has larger standard deviation. One factor that contributes to these results is the selection matrix S, which highly affects the performance of the controller. Small changes of this parameter can make the behavior completely different. The agent's random exploration of this parameter can result in very different results during the learning phase.

C. Safe Learning

The developed fail-safe mechanism was not only evaluated as a mechanical safety that enables the real robot to explore random action without human supervision. We validate the usefulness of providing information to the robot about the safety constraints violations. Thus, we compare the proposed reward function Equation (5) with a variant that does not provide any punishment when a safety constraint is violated, i.e., κ gives a reward if the task is completed or zero otherwise, see Equation (6). We trained all policy models with this modified reward function.

Results: Fig. 6 shows the comparison of the learning curves of all models with a reward function that does not penalize violation of safety constraints. The results clearly show that the overall performance considerably decreases. The learning speed also decreases, as can be noted by comparing the performance of, for example, the model A-13pd. Learning with active penalization helps the agent learn policies that yield rewards of $+100$ by 12,000 steps while it takes as much as 20,000 steps without penalization to achieve similar performance. Parallel control models show similar results. Moreover, the learning curves are nosier, meaning that the models can not reliably find a successful policy.

Additionally, we counted the average number of collisions detected during training sessions for each policy model. Table II shows the training session results using the proposed reward function with active penalization of the safety constraints and the

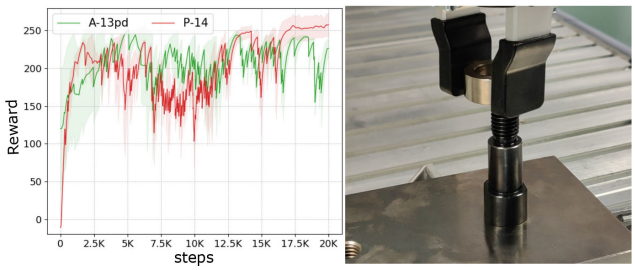

Fig. 7. Ring-insertion task. Hole clearance of 0.2 mm. Cumulative reward per step of 20,000-steps training sessions of A-13pd and P-14 policy models.

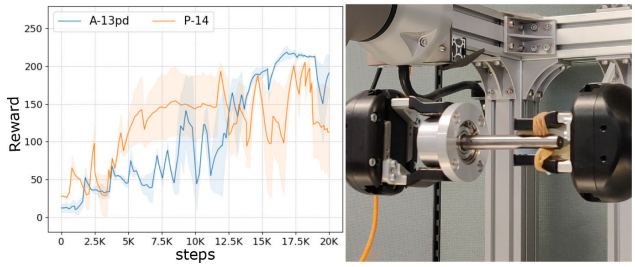

Fig. 8. Peg-insertion task. Hole clearance of 0.05 mm. Cumulative reward per step of 20,000-steps training sessions of A-13pd and P-14 policy models.

reward function without penalization. In all cases, we see a high decrease in the number of collisions when actively penalizing collisions. In other words, the training session can be considered safer when the robot gets feedback on the undesired outcomes, i.e. when safety constraints are violated. Particularly, in the case of the parallel control scheme, the models have difficulty understanding that collisions are a poor behavior; thus, those models keep getting stuck on episodes that finish too soon due to collision. These results also highlight that the models A-13pd and P-14 do not only learn faster than other models but also produce the lowest number of collisions within their family of policies. On the other hand, the policy models with the highest number of parameters, A-18 and P-24, are able to learn successful policies at the cost of producing the highest number of collisions.

D. Real Robot Experiments

Our proposed method was validated on real hardware using two high-precision assembly tasks. The first task involves an insertion task of a metallic ring into a bolt with a clearance of 0.2 mm, as shown in Fig. 7. The second task is a more precise insertion task of the metallic peg into a pulley, with a clearance of 0.05 mm, as shown in Fig. 8. Another robotic arm holds the pulley, and the center of the pulley is slightly flexible, which makes contact less stiff than the ring-insertion task. However, since the clearance is smaller, the peg is likely to get stuck if the peg is not adequately aligned, increasing the difficulty of solving the task. The best policy models from the previous experiment were used for training, P-14, and A-13pd. Both models were trained for 20,000 steps, twice. The episodes have a maximum length of 200 steps, about 10 s.

1) Ring-Insertion Task Results: From Fig. 7, both models A-13pd and P-14 can quickly learn successful policies that solve the task. The high stiffness of the ring and bold makes the task more likely to result in a collision. The model P-14 produced an average of 45 collisions per training session, while A-13pd produced 34. Despite firmly grasping the ring with the robotic gripper, the position/orientation of the ring can still slightly change. These slight changes can explain the drops in performance during the training session. However, the agents can adapt and learn to succeed in the task.

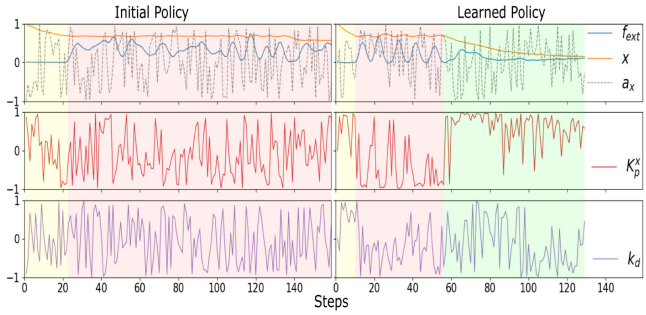

Fig. 9. A-13pd: policy performance evolution on peg-insertion task. On the left, performance of the initial policy tried by agent. On the right, performance of the learned policy after training. All values correspond to the insertion direction only. Only 160 steps are displayed for space constraints. Insertion task divided into three phases: a search phase before contact (Yellow), a search phase after initial contact (Red) and an insertion phase (Green).

2) Peg-Insertion Task Results: From Fig. 8, we can see that it takes a lot more learning time to find a successful policy for both policy models compare to the ring-insertion task. While both policy models find a successful policy after about 13 k steps, A-13pd achieved better consistent performance. As mentioned above, the physical interaction for this task is less stiff; thus, the average collisions per training session were fewer than in the ring-insertion task. For models A-13pd and P-14, the average number of collisions was 4 and 26, respectively.

The evolution of the policy model A-13pd, across a training session, is shown in Fig. 9. The figure displays the observation per time step of only the insertion direction. The actions, $\mathbf{a}_x$ and $\mathbf{a}_p = [K_p^x, k_d]$ are also displayed. Observations and actions have been mapped to a range of $[1, -1]$. The peg-insertion task has three phases. A search phase before contact (Yellow). A search phase after initial contact (Red). An insertion phase (Green). On the left, the initial policy, we can clearly see that the insertion was not successful even after 200 steps, as well as a rather random selection of actions. On the contrary, on the right side, the task is being solved at around 130 steps. On top of that, the controller's parameters k_d and K_p^x have a clear response to the contact force perceived. After the first contact with the surface (Red), k_d and K_p^x are dramatically reduced, as a result, decreasing motion speed and reducing stiffness of the manipulator, which reduces the contact force. Then, when the peg is properly aligned (Green), k_d and K_p^x are increased to apply force to insert the peg -against the friction of the insertion- and to finish the task faster.

V. Discussion

In this work, we have presented a framework for safely learning contact-rich manipulation tasks using reinforcement

learning with a position-controlled robot manipulator. The agent learns a control policy that defines the motion trajectory, as well as fine-tuning the force control parameters of the manipulator's controller. We proposed two learning force control schemes based on two standard force control methods, parallel position/force control, and admittance control. To validate the effectiveness of our framework, we performed experiments in simulation and with a real robot.

First, we empirically study the trade-off between control complexity and learning performance by validating several policy models, each with a different action space, represented by a different number of adjustable force control parameters. Results show that the agent can learn optimal policies with all policy models considered, but the best results are achieved with the models A-13pd and P-14. These models yield the highest reward during training, proving to be the best trade-off between system complexity and learn-ability.

Second, results on a real robot showed the effectiveness of our method to safely learn high-precision assembly tasks on position-controlled robots. The first advantage is that the fail-safe mechanism allows for training with minimal human supervision. The second advantage is that including information about the violation of safety constraints on the reward function helps speed up learning and reduce the overall number of collisions occurred during training.

Finally, in the usual peg insertion task, the motion trajectory is essential when the robot is in the air, while the force control parameters become essential when the peg is in contact with a surface or the hole. Results show that our framework can learn policies that behave accordingly on the different phases of the task. The learned policies can simultaneously define the motion trajectory and fine-tune the compliant controller to succeed in high-precision insertion tasks.

One of the limitations of our proposed method is that the performance is highly dependent on the choice of the controller's hyperparameters, more specifically, the base and range values of the controller's gains. In our experiments, we empirically defined said hyperparameters. However, to address said limitation, an interesting avenue for future research is to obtain these hyperparameters from human demonstrations, and then refine the force control parameters using RL. Additionally, for simplicity, we assume knowledge of the goal pose of the end-effector for each task. However, vision could be used to get a rough estimation of the target pose to perform an end-to-end learning, from vision to low-level control, as proven in previous work [7].

REFERENCES

[1] O. Kroemer, S. Niekum, and G. D. Konidaris, "A review of robot learning for manipulation: Challenges, representations, and algorithms," *CoRR*, 2019, *arXiv:1907.03146*.

[2] D. Kalashnikov *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proc. 2nd Conf. Robot Learn.*, 2018, vol. 87, pp. 651–673.

[3] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *Int. J. Robot. Res.*, vol. 37, no. 4-5, pp. 421–436, 2018.

[4] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 3389–3396.

[5] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from cad," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 1–9.

[6] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Trans. Robot.*, pp. 1–13, 2020.

[7] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[8] G. Schoettler *et al.*, "Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards," in *Proc. Int. Conf. Mach. Learn.*, 2019, *arXiv:1906.05841*.

[9] L. Johannsmeier, M. Gerchow, and S. Haddadin, "A framework for robot manipulation: Skill formalism, meta learning and adaptive control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2019, pp. 5844–5850.

[10] K. Lynch and F. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[11] B. Siciliano and L. Villani, *Robot Force Control*. vol. 540, Berlin, Germany: Springer, 2012.

[12] S. Chiaverini and L. Sciavicco, "The parallel approach to force/position control of robotic manipulators," *IEEE Trans. Robot. Autom.*, vol. 9, no. 4, pp. 361–373, Aug. 1993.

[13] N. Hogan, "Impedance control: An approach to manipulation," in *Proc. Amer. Control Conf.*, Jun. 1984, pp. 304–313.

[14] D. Mitrovic, S. Klanke, and S. Vijayakumar, "Learning impedance control of antagonistic systems based on stochastic optimization principles," *Int. J. Robot. Res.*, vol. 30, no. 5, pp. 556–573, 2011.

[15] M.-C. Chien and A.-C. Huang, "Adaptive impedance control of robot manipulators based on function approximation technique," *Robotica*, vol. 22, no. 4, pp. 395–403, 2004.

[16] M. Racca, J. Pajarinen, A. Montebelli, and V. Kyrki, "Learning in-contact control strategies from demonstration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 688–695.

[17] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 820–833, 2011.

[18] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2010, pp. 2397–2403.

[19] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *CoRR*, 2019, *arXiv:1907.07500*.

[20] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Representation*, 2016, *arXiv:1509.02971*.

[21] R. Martín-Martín, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space. an action space for reinforcement learning in contact rich tasks," in *Proc. Int. Conf. Intell. Robots Syst.*, 2019, pp. 2062–2069.

[22] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino, "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2062–2069.

[23] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 3406–3413.

[24] A. R. Mahmood, D. Korenkevych, G. Vasan, W. Ma, and J. Bergstra, "Benchmarking reinforcement learning algorithms on real-world robots," in *Proc. 2nd Conf. Robot Learn.*, Oct. 2018, vol. 87, pp. 561–591.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[26] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[27] R. Campa and K. Camarillo, "Unit quaternions: A mathematical tool for modeling, path planning and control of robot manipulators," in *Robot Manipulators*, M. Ceccarelli, Ed. Rijeka: IntechOpen, 2008, ch. 2, doi: 10.5772/6197.

[28] D. A. Lawrence, "Impedance control stability properties in common implementations," in *Proc.. IEEE Int. Conf. Robot. Autom.*, 1988, vol. 2, pp. 1185–1190.

[29] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, vol. 3, pp. 2149–2154.