AFORCE: A Bio-Inspired Action Space for Multimodal Manipulation Learning and Adaption

Anonymous Author(s)

Abstract: Intelligent agents must be able to think fast and slow to perform elaborate manipulation tasks. Reinforcement Learning (RL) has led to many promising results on a range of challenging decision-making tasks. However, in real-world robotics, these methods still struggle, as they require large amounts of expensive interactions and have slow feedback loops. On the other hand, fast human-like adaptive control methods can optimize complex robotic interactions, yet fail to integrate multimodal feedback needed for unstructured tasks. In this work, we propose to factor the learning problem in a hierarchical learning and adaption architecture to get the best of both worlds. The framework consists of two components, a slow reinforcement learning policy optimizing the task strategy given multimodal observations, and a fast, real-time adaptive control policy continuously optimizing the motion, stability, and effort of the manipulator. We combine these components through a bio-inspired action space that we call AFORCE. We demonstrate the new action space on a contact-rich manipulation task on real hardware and evaluate its performance on three simulated manipulation tasks. Our experiments show that AFORCE drastically improves sample efficiency while reducing energy consumption and improving safety.

Keywords: Reinforcement Learning, Adaptive Control, Manipulation

1 Introduction

Deep reinforcement learning (RL) is a promising approach to solve complex robotic manipulation tasks by enabling seamless integration of perception, decision-making, and control. Such methods can learn mappings from a variety of sensors to actions but may require millions of interactions to converge, especially in the real world [1]. This sample inefficiency could be attributed to the inherent properties of the action and observation space of many real-world tasks.

Real-world action spaces are inherently continuous. Yet, finding the correct action only solves part of the problem. To achieve the desired outcome of an action, the agent must control the interaction appropriately. In the context of manipulation, the agent must be able to compensate for any forces arising from physical interaction. Moreover, it must have the ability to continuously adapt the mechanical impedance at contact points to stabilize the manipulation [2, 3]. Such real-time adaption is important, however, most modern deep RL methods fail to achieve it due to feedback loops being slower than the robot control frequency.

Real-world observation spaces are inherently multimodal. In addition, different sensor signals must be evaluated at different time intervals. Visual information often changes slowly over time and requires deliberate reasoning to process. In contrast, proprioceptive measurements need to be evaluated in real-time for control and sometimes require an unconscious, intuitive response to establish reactive and safe behavior [4]. In recent work, RL has shown to allow elegant integration of multiple modalities [5, 6], yet, it remains challenging to account for temporal context.

In this work, we propose to combine bio-inspired adaptive control and reinforcement learning. Inspired by studies in neuroscience [7, 8, 9], we model the solution of learning robotic manipulation as a hierarchical control policy. An outer high-level policy – or task planner – produces actions given...
Figure 1: (left) Intelligent agents have to think fast and slow to solve complex manipulation tasks. AFORCE allows combining slow reinforcement learning with fast adaptive control to get the best of both worlds. (right) AFORCE in action on a contact-rich manipulation task. After locating the wipe spot through visual feedback, (A) the expert policy commands to approach the spot, (B) apply a specific force to the surface and (C) execute a wiping motion.

observations of the environment. It is optimized in an RL framework through the maximization of future reward [10]. Additionally, an inner low-level policy computes robot commands given high-level actions and plant measurements. It optimizes the interaction between the manipulator and its environment by minimizing instability, motion error, and effort [11].

The key idea of this work is to enable modular integration of perception modules and modern control paradigms, yet reducing the computational complexity of the individual components. Intuitively, the RL policy is not required to optimize the interaction with the environment through tactile and proprioceptive sensing, hence we keep the RL observation and action space small.

To combine these methods, we propose Cartesian Adaptive Force-Impedance Control (AFORCE) as a novel action space for learning robotic manipulation. AFORCE allows continuous adaption of impedance and force, both critical to contact stability, disturbance compensation, and energy efficiency. We show that AFORCE significantly increases the sample efficiency and stability of RL agents learning manipulation tasks. In addition, we demonstrate the efficacy of the action space on real hardware by performing a contact-rich manipulation task and comparing its energy efficiency to other compliant action spaces. In this work, we focus on validating the action space and leave vision-based RL experiments for future work.

Our primary contributions are: (i) AFORCE, a bio-inspired action space implementing a hierarchical control policy for learning contact-rich manipulation tasks. (ii) A perspective on human motor control and its capacity to improve RL policies. (iii) Evaluation of the learning efficiency and comparison to baseline action spaces in robotic manipulation tasks. (iv) Demonstration of the action space efficacy in real-world robot experiments.

2 Related Work

Adaptive Impedance Control is a paradigm that takes inspiration from human motor control by adapting endpoint force and impedance to compensate for environment forces and instability [12, 13]. The dynamic properties of such controllers were previously analyzed and demonstrated in [11].

In [14] the paradigm is extended to iteratively adjust a preplanned trajectory in the presence of obstacles. Furthermore, Li et al. [15] introduce a trajectory adaption law, with the assumption that the desired contact force is maintained and a reference trajectory exists. Johannsmeier et al. [16] propose a framework to optimize adaptive parameters, given a trajectory that can solve the task. Our method builds on the control laws introduced and analyzed in [14, 11]. However, in contrast to prior work, we don’t optimize the task trajectory in the same process as force and impedance or assume the trajectory to be given. Instead, we propose to factor the system in a hierarchical control structure such that the task trajectory can be learned with reinforcement learning.

Hierarchical Reinforcement Learning (HRL) [7, 17] is widely used for robotic tasks, such as manipulation [18], navigation [19], and locomotion [20]. Many HRL algorithms have been proposed, such as Hierarchies of Machines [21], MAXQ [22], Hierarchical Policy Search [23], and more recently Layered Direct Policy Search [24]. Such a structure allows temporal and state abstractions which reduces the computational complexity of the individual components. In this work, we use this concept with a model-free high-level RL policy and a traditional model-based low-level policy, both
containing learnable parameters. However, in this work, we refrain from learning the parameters of
the low-level policy and keep them fixed for each task.

Action Spaces for Robot Learning. Robot learning research has a rich history in combining well-
established control methods – often in the form of an action space – with machine learning [25].
This hybrid approach can result in an improvement in sample efficiency, robustness, and quality of
the learned policies [26, 27]. In [28], a dual-arm system learns to juggle by learning a world model
and an optimal control policy. Peters and Schaal [29] use an Operational Space based action space
to learn an RL policy that follows a reference acceleration. Another popular action representation
is dynamical movement primitives (DMPs) [30] which utilize a low-dimensional parameterization
of a dynamical system to generate smooth trajectories. DMPs are often used with policy search
methods [31] and HRL [32, 24]. Buchli et al. [33] propose to learn a variable impedance controller
in joint space and DMPs to optimize energy consumption while refining an initial trajectory. Similar
to our approach, Martín-Martín et al. [34] uses a hierarchical structure to combine a model-free
high-level policy with an impedance controller. In contrast, we use a bio-inspired adaptive low-level
policy which independently optimizes trajectory deviation, stability, and energy consumption. For
manipulation, the exerted force of the robot to the environment is essential and some action spaces
include explicit force control [35, 36] or implicit modulation based on impedance control [34, 37].
In this work, the agent is equipped with direct means to apply a wrench to the environment.

3 Preliminaries

Compliant Robot Control in RL Action Spaces. Compliant robot behavior is a classical problem
in robotics [38, 39]. It is most relevant when the goal of the robot is to manipulate its environment
or the environment is only partially known [40]. In cartesian impedance control, the inputs to the
controller are typically the desired end-effector position \( x_d \) and velocity \( \dot{x}_d \), from which the con-
troller computes joint torques \( \tau_u \). The torques are computed to realize a desired, compliant dynamic
relationship between the robot motion and external forces [40].

Consider the dynamical model of the robot

\[
M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau_u + \tau_{ext},
\]

where \( M(q) \) is the symmetric, positive definite inertia matrix, \( C(q, \dot{q}) \) is the Coriolis/centrifugal
matrix, \( g(q) \) is the vector of gravity torques, \( \tau_u \) is the vector of joint torques and \( \tau_{ext} \) is the vector
of external torques. Following the operational space formulation [39], the control law of the impedance
controller in task space coordinates is

\[
\tau_u(t) = J(q)^T F_u(t),
\]

\[
= J(q)^T (-K\dot{e}(t) - D\dot{\dot{e}}(t)),
\]

with input wrench \( F_u(t) \), Jacobian matrix \( J(q) \), pose error \( e(t) = x(t) - x_d(t) \) and velocity
error \( \dot{e}(t) = \dot{x}(t) - \dot{x}_d(t) \). The stiffness \( K \) and damping \( D \) matrices are both positive definite,
symmetric matrices. For clarity, we drop the vector of gravity torques \( g(q) \). A full derivation can be
found in [40].

This controller can be used as an action space by tracking a reference signal generated by an RL
agent. The action space \( A \) consists of absolute or relative values \( x_d(t) \in SE(3) \) and needs a fixed
mechanical impedance \( K, D \) to be implemented. By abuse of notation, we write \( u(t) = f(\alpha = x_d) \)
to indicate that the action generated by the agent is a time depended pose \( x_d(t) \).

A reasonable extension to the fixed impedance action space is to equip the agent with the means
to vary the impedance by extending the action space [34]. This variable impedance action space
consists of tuples \( u(t) = f(\alpha = (x_d, K_d, D_d)) \) and is implemented in a similar fashion.

Reinforcement Learning. In this work, we construct the high-level policy by maximization of fu-
ture task reward. The goal is to learn a policy to perform tasks in continuous action and state spaces.
Therefore, the tasks are modeled as finite-horizon, discounted Markov Decision Processes (MDP)
defined by the tuple \((S, A, P, r, \gamma, \rho, T)\). Here, \( S \) is a continuous state space, \( A \) is a continuous
action space, \( P : S \times A \rightarrow S \) are the transition dynamics of transitioning between states given an
action, \( r : S \times A \rightarrow \mathbb{R} \) is a scalar field quantifying the reward, \( \gamma \in [0, 1) \) a discount factor, \( \rho \) an
initial state distribution and horizon \( T \). The goal of the learner is to find a policy \( \pi : S \rightarrow \mathcal{P}(A) \)
which selects actions to maximize the expected future reward \[10\]. Assuming, that the policy is a function parameterized by \( \theta \), the policy is optimized such that

\[
\theta^* = \arg\max_{\theta} J(\theta) = \arg\max_{\theta} E_{\pi} \left[ \sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right].
\]

4 Learning Hierarchical Control for Robotic Manipulation

Consider the problem of learning a manipulation task with a robotic arm from observations and measurements. To succeed, the control policy needs to generate the correct high-level actions, which are relevant sub-goals required to complete the task. Additionally, it needs to translate these high-level actions to the actuation of the manipulator such that the sub-goals are achieved. To implement this behavior, we design an action space and policy according to the following principles:

- **Multimodal Observations** are at the core of most manipulation tasks. Visual, tactile, and proprioceptive sensing vary drastically in nature and temporal context. A control policy must process different modalities at different frequencies to make the most out of available data without unnecessarily increasing the complexity of decision-making and actuation.

- **System Stability** is required at all times while performing manipulation. Studies have shown that slow feedback delays can lead to undesirable dynamic behavior and in the worst case to unstable dynamics \[38, 2, 11\]. Thus, we require the policy to be able to continuously adapt the interaction to stabilize the system dynamics.

- **Physical Efficiency** is usually not an explicit goal of reinforcement learning agents. One could add an energy-specific term to the reward function of the agent, however, increasing the number of factors in the signal might require careful tuning of the coefficients and can in some cases deteriorate the learning performance \[41\]. Hence, the policy should optimize its energy consumption independent of the reward signal.

- **Safety** is one of the most critical attributes of any intelligent agent. While negative reward is important to learn safe interaction, we want the robot to be as compliant as possible unless the task requires otherwise. This can prevent harming the system if unexpected collisions occur.

- **Learning Efficiency** is a central struggle in RL. Especially in robotics, the state and action spaces are continuous and can have large dimensions. The complexity increases exponentially with each extra dimension according to the curse of dimensionality \[25\]. To reduce the complexity of the learning problem, it is important to keep the dimensionality of the action space as low as possible.

4.1 Hierarchical Learning and Adaption

Based on these principles and inspired by studies in neuroscience \[7, 17, 9\], we propose to model the solution of a general manipulation task as a hierarchical control system that is capable of thinking fast and slow. A similar concept has been explored in previous work in robotics and RL \[34, 42\].

We factor the policy into two components. A slow outer loop is modeled as a policy \( \pi(a) : \mathcal{O} \to \mathcal{A} \) which maps environment observations to actions. Its goal is to plan the correct actions to solve a given task and it is parameterized by \( \theta \). We consider this loop to be analogous to slow neural feedback loops found in human and animal behavior used for deliberate decision-making \[43, 17\].

In contrast to prior work \[34, 44\], the fast inner loop of the system is a second optimization process. In this work, we model it as a second policy that maps actions and plant measurements to robot control commands \( f(a, y) : \mathcal{A} \times \mathcal{Y} \to \mathcal{U} \).\(^1\) The inner policy continuously adapts the interaction and motion to minimize the trajectory error and consumed energy of the system. It is parameterized by its own set of parameters \( \psi \) determining the physical behavior. The goal of the inner loop is to optimize the action commanded by the high-level policy. An illustration can be found in Figure 1.

The central goal of this architecture is to incorporate fast and slow learning processes to reduce the computational complexity of the individual components. Intuitively, the high-level policy \( \pi \) is not required to learn and adapt high-frequency physical properties of the interaction, such as contact stability or energy efficiency, consequently reducing the computational effort.

\(^1\)In this work, we consider measurements to be real-time sensor data which is required for robot control and observations a compilation of this data and other exteroceptive sensor modalities, such as image information.
4.2 Cartesian Adaptive Force-Impedance Action Space

The goal of the low-level control policy is to compute the joint torques $\tau_u$ to establish efficient, safe manipulation of the environment, such that the commanded actions of the high-level policy are successful. For the environments we are interested in, we require low target error, stable interaction during contact-rich actions, and explicit force control for environments where it is necessary.

In this work, the low-level policy is a bio-inspired adaptive force-impedance controller [11]. Instead of varying the impedance through the neural feedback of the task planner, we adapt it through the minimization of instability, motion error, and effort [45, 13].

Following the formulation in [14, 11], we model the input wrench $F_u(t)$ as a composition of a restoring feedback term $r(t)$ (see (3)) and feedforward term $v(t) = -F_{ff}(t) - F_d(t)$ [16], such that

$$
\tau_u(t) = J(q)^T F_u(t), \\
= J(q)^T \left( -F_{ff}(t) - F_d(t) - K(t)e(t) - D(t)e(t) \right). 
$$

Here, $F_{ff}(t)$ denotes an adaptive feedforward wrench that optimizes the motion of the manipulator [13]. An optional desired wrench profile $F_d(t)$ can be added for environments that require specific force control, such as wiping or many other tool applications.

We adapt the impedance and feedforward wrench according to

$$
\dot{K}(t) = \beta e(t) - \gamma I \tag{7}
$$

$$
\dot{F}_{ff}(t) = \alpha e(t) - \mu F_{ff}(t) \tag{8}
$$

with feedback error $e(t) = e(t) - \delta \dot{e}(t)$ and $\delta > 0$. In addition, $\alpha, \beta, \gamma, \mu$ are constant symmetric positive-definite matrices that determine the adaption behavior of the control policy. $\alpha$ and $\beta$ are considered to be stiffness and feedforward adaption rates that determine the speed at which the controller can adapt to errors. In contrast, $\gamma$ and $\mu$ are relaxation factors that decrease the exerted energy of the system.

Intuitively, the first term of equation (7) adapt the mechanical stiffness proportionally to the absolute error, reducing the tracking deviation. Simultaneously, the second term $-\gamma I$ constantly decreases the stiffness, thus relaxing the arm and reducing the energy consumption along dimensions where the stiffness is currently not required. Hence, the impedance adaption law allows concurrent minimization of the energy and tracking error. A similar intuition can be employed for (8).

The resulting impedance and feedforward wrench at time $t$ is thus $K(t) = \int_0^t \dot{K}(t) \, dt + K(0)$, $D(t) = 2\sqrt{K(t)}$ and $F_{ff}(t) = \int_0^t \dot{F}_{ff}(t) \, dt$, respectively.

For clarity, we compile all adaptive parameters $\psi = (\alpha, \beta, \gamma, \mu)$ and define the Cartesian Adaptive Force-Impedance (AFORCE) action space as

$$
\alpha(t) = f_\psi(a = (x_d, F_d)), \tag{9}
$$

with desired pose $x_d(t)$ and optional wrench $F_d(t)$. The values of $\psi$ can be chosen by a designer or learned. In this work, we choose them experimentally and leave learning them in a complete RL framework for future work.

AFORCE is agnostic to the choice of a task planner. A suitable policy can either be directly programmed through expert knowledge, learned from demonstrations or through reinforcement.

5 Experiments

We evaluate the performance of AFORCE in both simulation and on real hardware. In simulation, we assess the learning performance on three challenging tasks using the rebruite simulator [46]. On real hardware, we demonstrate the efficacy of AFORCE on a contact-rich manipulation task. In the experiments, we aim to answer the following questions: (i) Is the action space suitable for learning robotic manipulation tasks on real hardware? (ii) How does the action space perform compared to baselines for contact-rich tasks? (iii) Is AFORCE suitable for model-free reinforcement learning? (iv) Does the action space enable safe and energy-efficient learning?
5.1 Real-World Experiments

In all experiments, the high-level control policy outputs actions at 20 Hz, and the low-level policy runs at 1 kHz. We use a jerk-limited online trajectory generator to generate smooth trajectories from the actions [47]. We also show qualitative results of the robot in action in the supplementary video.

**Contact-Rich Manipulation.** In this experiment, we demonstrate that the hierarchy and action space are suitable to solve contact-rich manipulation tasks and investigate the adaptive properties of the policy. To this end, we use an expert high-level policy to solve a wiping task. To highlight the multimodality of the task, we briefly cover its measurement and observation space. The measurement space at 1 kHz for the experiment consists of robot joint positions and velocities, end-effector pose and velocity, and an estimated external wrench, such that the agent can control the force. The observation space at 20 Hz consists of the current pose, recent external wrench measurements, and the centroid of the wipe spot, that can be detected with a visual sensor.

The evolution of the experiment is shown in Figure 2. After detecting the wipe spot, the policy computes actions to approach the centroid and establishes contact (first dashed vertical line). Next, it commands to apply a force of 10 N to the surface. Once the desired force is applied (second dashed vertical line), the policy commands a circular motion perpendicular to the surface via the action space while trying to maintain the same amount of force. The adaption of the policy to optimize the motion can be seen in (b), (d), and (e). During the time in which the policy applies the specified force, the mechanical impedance of the system decreased due to a low cartesian error. However, once the circular motion commences, the current impedance is not sufficient to achieve a low tracking error. Thus, it increases the stiffness periodically to adapt to the new motion. The result of the adaption to decrease the total cartesian tracking error is shown in (d) and (e).

**Energy Efficiency and Tracking Error.** In addition, we investigate how AFORCE performs in contrast to other compliant action spaces in a wiping task. In particular, we compare AFORCE to three fixed impedance action spaces with a low, mid, and high impedance configuration. All spaces use the same force controller. All impedance values of this experiment can be found in Table 2.

We employ the same task plan as in the previous experiment but extend the wiping phase to wipe a total of eight full rotations. To acquire representative results, we conduct the same experiment for multiple trials. The efficiency of the different action spaces is evaluated by comparing the total energy consumption of the manipulator and the total tracking error$^2$ along the episode (see Figure 2).

All action spaces except low managed to wipe off all the paint. We found that AFORCE only consumes about half the energy of high while maintaining a competitive total error. In our experiments, low did not manage to maintain full contact in its trials, thus leaving paint unwiped.

$^2$sum of absolute position error and quaternion difference
5.2 Simulated Experiments

In simulation, we conduct experiments on three manipulation tasks using the robosuite simulator [46]: Door, Lift and Wipe. The chosen tasks aim to cover a wide range of common manipulation challenges, such as contact dynamics, and kinematic constraints. We use Soft Actor Critic (SAC) [48, 49], as our RL policy, which is a model-free, off-policy deep RL algorithm. The actor and critic both consist of two layers with 1024 units each and a complete list of the used hyperparameters can be found in Appendix D. To evaluate the performance of AFORCE, we train multiple different seeds for each task and action space. At the beginning of each seed, we pretrain the models with 5 thousand samples collected by rolling out random actions. Then, we evaluate the model every 10 thousand steps for 10 episodes and report the average reward.

Tasks. In Door, the agent has to learn to open a door by pushing down a spring-loaded handle and pulling the door open. We penalize excessive contact forces over 60 N. In Lift, the goal of the agent is to grasp an object and lift it off the table. Lastly, in Wipe, the agent has to learn to wipe markers off a table using a wiping tool and applying a force above 15 N. We add the same penalty as in Door and only remove markers off the table if the force is below the force penalty threshold. Additionally, we vary the task and robot initialization on each task reset. The joints of the robot are initialized with Gaussian noise, in Lift the object position is varied, in Door the position and orientation of the door is varied and in Wipe the wipe spots are randomized.

Low-level Control Policies. In all simulated experiments, the high-level control policy outputs actions at 20 Hz. The low-level control policies run at 500 Hz, using a linear interpolator to ensure smooth trajectories generated from the high-level actions. For the comparison, we are using the variable_jp implementation of robosuite with \( u(t) = f^{\text{variable}}(a = (x_d, K_d)) \) which is a variable impedance action space. The environments Door and Lift do not require the explicit application of force, thus the AFORCE action space becomes \( u(t) = f^{\psi}(a = x_d) \).

For the Wipe environment, we extend AFORCE by a desired wrench that is applied to the environment if contact is established. The policy is \( u(t) = f^{\psi}(a = (x_d, F_d)) \) and we implement the desired wrench using a PID controller (see Appendix A). The controller can apply the desired wrench up to 50 N but not enough to cause a penalty. To gain insights on the importance of a force controller, we also evaluate the performance of \( \text{variable + force} \) \( u(t) = f^{\text{variable}}(a = (x_d, K_d, F_d)) \). Similarly, we evaluate \( \text{low + force} \), a low impedance action space with force control \( u(t) = f^{\text{low}}(a = (x_d, F_d)) \).

More details on tasks and control parameters can be found in Appendix C.

Results: Learning Performance. The results of the simulated experiments are shown in Figure 3. In our comparison against baselines, AFORCE outperforms all alternatives on all tasks. For all three tasks, it converges faster and reaches a higher maximum reward than the other action spaces.

In Door, AFORCE achieves a higher maximum reward and lower standard deviation than \textit{variable}. We attribute that to the lower average stiffness of AFORCE (see Appendix C). Hence, the collision penalties are smaller than of an agent which explores its stiffness. Additional support for this hypothesis can be found in Table 4. The percent of episodes with a penalty is twice as high for \textit{variable} than for AFORCE and the average penalty is smaller. However, these values depend drastically on the success of the learning process, thus they should be merely seen as an indicator.

In Lift, we observe a similar result. \textit{variable} requires around 300 thousand steps before it starts to converge. Since there are no negative rewards in this environment, a contributing factor may be
the increased size of the action space which needs to be explored. Furthermore, AFORCE achieves a very stable reward during evaluation across multiple seeds. Once the task is learned, the reward variance of AFORCE in Lift becomes small and all policies succeed in most evaluation episodes.

In Wipe, we compare AFORCE to three action spaces, with and without force control. The task demands precise force application to wipe the dirt spots. We observe that the performance of variable is almost identical with explicit force control. During the training period, the stiffness of variable is higher than that of the other action spaces. The high stiffness resulted in high external forces when the manipulator made contact. This made it impossible for the force controller to regulate the desired force. Even when the variable + force agent chose the right desired force, the controller was unable to stabilize the interaction. The low + force space performs better than the variable spaces. It wipes on average more markers but cannot consistently remove all. After inspecting training videos, it is often seen sliding on the table without making full contact. We attribute this to the fact that the force controller is engaged when the tool makes contact, but the agent is unable to align the tool with the table and stabilize the interaction. At last, AFORCE reaches its maximum reward consistently after 500 thousand steps. We observe that AFORCE manages to wipe off all markers after only a few training episodes (see Figure 5). Our action space can align the tool with the surface to use the force controller and simultaneously stabilize the interaction at the contact points.

Results: Energy Efficiency and Safety. First, we compare the average energy one action consumed during training (see Table 1). We found that variable spaces consumed more energy in all three environments. In the Wipe environment, the low + force action space needed the least amount. We observe that variable + force used less energy on average than variable. This could mean that the force controller attempts to regulate the measured external forces to the desired, safe value but fails. However, to definitively answer this question, more experiments are required.

Finally, we investigate how safe the learning of the different policies is. Hence, we evaluate two metrics: (a) the percentage of training episodes with a penalty (see Table 1), and (b) the average penalty per episode (see Table 4). We observe that no action space was completely safe. AFORCE performs best in terms of metric (a), yet received a penalty in 12% of episodes in Wipe. On average, the force penalty of the low + force was the lowest. In our experiments, the variable spaces did not converge to a physically feasible solution. They are penalized in more than three-quarters of the episodes and collect a significant amount of negative reward which is scaled by the amount of force above the penalty threshold.

### 6 Conclusion and Future Work

We presented AFORCE, an action space for learning contact-rich manipulation tasks. It draws inspiration from neuroscience and biomechanics to allow concurrent optimization of the task strategy, physical interaction, and motion. We demonstrated our approach on real hardware, showed its ability to solve a complex, dynamic manipulation task, and compared its energy efficiency to other compliant action spaces. We also showed that our method outperforms action space alternatives in simulation in terms of sample efficiency, learning stability, energy consumption, and safety. The main limitation of this work is that the adaptive parameters have to be tuned manually. While there is a large set of valid parameters, an invalid set can have a drastic impact on the learning performance. Additionally, the adaption of the interaction is reactive and the agent has no means to adapt it proactively. For future work, we plan to address these problems by learning the adaptive parameters and high-level policy concurrently and investigate methods that allow the agent to interfere with the adaption behavior.
References


The appendix consists of additional details on cartesian force control, control parameters for real-world experiments and descriptions of tasks, evaluation metrics, observation space, and control parameters of the simulated environments.

**Appendix A  Cartesian Force Control**

As introduced in in section 4, AFORCE allows to add a desired wrench profile $F_d(t)$ to the control law to equip the agent with explicit force control. This paradigm allows to apply an accurate contact force in contrast to imprecise impedance control force regulation. An overview and comparison of different force control methods can be found in [50]. In this work, we implement the force control using a simple PID controller, as proposed in [51]. We use the same approach for simulation and real world experiments.

Consider equation (6), the term $F_d(t)$ is implemented as

$$F_d^*(t) = K_p (F_d(t) - F_{\text{ext}}(t)) + K_d (\dot{F}_d(t) - \dot{F}_{\text{ext}}(t)) + K_i \int_0^t (F_d(\sigma) - F_{\text{ext}}(\sigma)) \, d\sigma \quad (10)$$

where $K_p$, $K_i$, and $K_d$ are constant, diagonal, positive definite matrices for proportional, derivative and integral gains. $F_d(t)$ is the desired wrench which gets specified by the agent and $F_d^*(t)$ the value which we implement in the control law. The external wrench $F_{\text{ext}}(t)$ can be obtained via a force/torque sensor or estimated through proprioceptive sensors [52]. In this work, we use the latter approach. Additionally, we increase $F_d(t)$ smoothly once contact has occurred.

For both simulation and real experiments, we used $K_p = 1_{6 \times 6}$, $K_d = 0 \cdot 1_{6 \times 6}$ and $K_p = 2 \cdot 1_{6 \times 6}$.

**Appendix B  Additional Details for the Real-world Experiments**

The hardware setup for the real-world wiping task can be seen in Figure 2. For our experiments, we used one Franka Emika (FE) Panda robot, two Linux PCs, and one depth camera. The setup requires two computers because the PC which controls the robot requires a real-time kernel patch. For this patched machine, there are no NVIDIA drivers available which are required for visual data processing and machine learning libraries. We used the standard FE gripper with a 3d printed adapter to attach a rubber polishing utensil with velcro fabric. To the velcro fabric, we attached a generic whiteboard eraser made from felt. Furthermore, we use an Intel RealSense camera with depth for wipe spot detection and localization, for which we calibrated the camera to the robot coordinate frame.

To calibrate the camera, we attached QR code tags to the end-effector of the robot and captured multiple images in different end-effector poses. From that, we compute the transformation matrix from the camera frame to the robot base frame.

To control the robot, the low-level policy is implemented in C++, whereas the high-level policy is implemented in python. To establish efficient communication between the two, we used pybind11.

For networking between the two machines python sockets were used.
The parameters which define the low-level control policy are shown in Table 2. For the experiment, we get $x_a(t)$ and $F_a(t)$ of the expert task planner. We used $\Delta x_a(t)$ actions in the range of $[-1, 1]$ for the experiments. The policy first translates the high-level policy command range to the control command range of $[-0.05, 0.05]$ and clipped it to the maximum and minimum range if the action is out of bounds as defined in the table. For the wiping experiment, we commanded no additional rotation. The adaptive parameters were found experimentally within a few trials. However, it should be noted that these should not be considered the optimal parameters, as indicated by the isotropic values. These values are merely an example supposed to highlight the advantages and disadvantages of the adaptive controller. We leave finding the optimal parameters for a specific task in a full reinforcement learning framework for future work.

Appendix C  Additional Details for the Simulated Experiments

C.1 Tasks

**Door.** In Door, the agent has to learn to open a door by pushing down a spring-loaded handle and pulling the door open. We keep the implementation of the task identical to the robosuite [46](version 1.2.2) implementation, except for one detail. We penalize excess contact forces over 60 N to incentivize safe interaction. Hence, the shaped reward is computed according to the distance of the gripper to the handle and the rotation of the door. From that reward, we subtract a penalty, scaled by the excessive value, if it occurs. As the default initialization in robosuite, at the beginning of each episode the position of the door uniformly sampled in $x \in [0.07, 0.09]$, $y \in [-0.01, 0.01]$ and an orientation of $\alpha \in [-\frac{\pi}{2}, -0.25, -\frac{\pi}{2}]$ around the z-axis.

The observation space that the RL agent observes at 20 Hz consists of proprioceptive and exteroceptive information. The proprioceptive observations are joint positions, joint velocities, end-effector pose, gripper position and gripper velocity. The exteroceptive observations are the door position, door handle position, relative position of end-effector to door, relative position of end-effector to handle, joint position of the door handle and door hinge. Furthermore, the measurement space of the adaptive low-level policy at 500 Hz consists of only proprioceptive information required for impedance control: joint positions, joint velocities, end-effector pose, and end-effector velocity.

**Lift.** In Lift, the goal of the agent is to reach an object, grasp it and lift it off the table while holding it. We did not make any changes to the task from the default robosuite implementation. Thus, the agent gets rewarded when the manipulator gets closer to the object, when it is grasping the object and finally if the object is lifted. As the default initialization in robosuite, at the beginning of each episode the position of the object uniformly sampled in $x, y \in [-0.03, 0.03]$.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input min/max</td>
<td>$[-1, 1]$</td>
</tr>
<tr>
<td>Output min/max [m]</td>
<td>$[-0.05, 0.05]$</td>
</tr>
<tr>
<td>rel. max. dynamics</td>
<td>0.7</td>
</tr>
<tr>
<td>$x_{\min}$ [m]</td>
<td>$[0.3, -0.35, -0.1]$</td>
</tr>
<tr>
<td>$x_{\max}$ [m]</td>
<td>$[0.65, 0.35, 0.3]$</td>
</tr>
<tr>
<td>$K_{\text{initial}}$ [N/m, Nm/rad]</td>
<td>$\text{diag}{550, 550, 550, 50, 50, 50}$</td>
</tr>
<tr>
<td>$K_{\text{min}}$ [N/m, Nm/rad]</td>
<td>$\text{diag}{50, 50, 50, 10, 10, 10}$</td>
</tr>
<tr>
<td>$K_{\text{max}}$ [N/m, Nm/rad]</td>
<td>$\text{diag}{1500, 1500, 1500, 100, 100, 100}$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>$\text{diag}{0.001, 0.001, 0.001, 5.0, 5.0, 5.0}$</td>
</tr>
<tr>
<td>$\mu$</td>
<td>$\text{diag}{0.5, 0.5, 0.5, 0.4, 0.4, 0.4}$</td>
</tr>
<tr>
<td>$\beta$</td>
<td>$\text{diag}{10.0, 10.0, 10.0, 1.0, 1.0, 1.0}$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>$\text{diag}{0.075, 0.075, 0.075, 0.0085, 0.0085, 0.0085}$</td>
</tr>
</tbody>
</table>

| $K_{\text{low}}$ [N/m, Nm/rad] | $\text{diag}\{200, 200, 200, 20, 20, 20\}$ |
| $K_{\text{mid}}$ [N/m, Nm/rad] | $\text{diag}\{550, 550, 550, 30, 30, 30\}$ |
| $K_{\text{high}}$ [N/m, Nm/rad] | $\text{diag}\{1300, 1300, 1300, 50, 50, 50\}$ |

Table 2: Control parameters of the real-world robot experiments. The upper part are the values for the adaptive control policy and the lower part are the low, mid, and high fixed impedance action space parameters.
The observation space of the RL policy at 20 Hz consists again of proprioceptive and exteroceptive information. The proprioceptive observations are identical to the Door task: joint positions, joint velocities, end-effector pose, gripper position, and gripper velocity. However, the exteroceptive consist of the object position, orientation of relative position to the gripper. The measurement space of the adaptive policy at 500 Hz is the same as in Door: joint positions, joint velocities, end-effector pose, and end-effector velocity.

Wipe. In Wipe, the agent has to learn to wipe markers off a planar table using a wiping tool. In order to make the task more realistic, we made changes to the default implementation of the environment. Instead of removing a marker off the table once contact has occurred, we only remove them once a specific force threshold has been met of 15 N. This is similar to real wiping experiments, in which without enough force applied the tool only smears the paint or leaves paint behind. Additionally, to discourage potentially dangerous interaction we only remove markers off the table if the force is below the force penalty threshold. Although it is not very realistic, it hinders the agent to learn to just hit the surface with large amounts of force to remove the markers because of reward scaling. In initial experiments, we found this to be the case. To this end, we reward the agent to reach and touch the markers with the tool and apply a force to the marker to wipe it off the table. Similarly to Door, we subtract a scaled force penalty if the threshold is exceeded. The agent is rewarded for reaching and contacting the table, wiping off markers, and if the entire table has been cleaned. As the default initialization in robosuite, at the beginning of each episode the position of 100 wipe spots is randomized in one connected cluster.

The observation and measurement space for the wiping task look slightly different than for the other two tasks due to the need for force control. The proprioceptive information in the observation space gets extended by a boolean contact field and the gripper information is removed. Thus, the observation space consists of joint positions, joint velocities, end-effector pose, and the contact flag. The exteroceptive information entails the wipe spot centroid, wipe radius, relative position of the end-effector to the centroid, and to the next wipe spot. The adaptive low-level policy requires joint positions, joint velocities, end-effector pose, end-effector velocity, and external wrench at the end-effector for force control.

C.2 Control Parameters

For all experiments, independent of the task or the controller we used a very basic controller configuration that defines the minimum and maximum impedance as well as position limits. The configuration can be found in Table 3. The top section of the table shows the default configuration of all controllers. The three sections below show the adaptive parameters that we used for the simulated tasks. We used the same feedforward wrench parameters in all adaptive controllers. For the feedback adaption, we changed the parameters based on the task. It should be noted that these parameters should not be considered to be the optimal parameters for the task.

C.3 Additional Analysis of Learning Performance and External Forces

In this subsection, we add some additional analysis of the results on learning performance and safety. The results in Section 4 suggest that AFORCE accelerates and stabilizes the learning of the three
about the same stiffness in all three rotational directions, AFORCE keeps the orientation around z
task shows the strength of the bio-inspired design of AFORCE. While stiffness is higher than of AFORCE which correlates with the higher contact forces as seen in column

\[ \text{low + force} \]

Figure 5 (a) shows the average force penalty (over 10 evaluation episodes) during the evaluation of

\[ \text{learning a task with a scalar reward signal, they can make the learning of the task more complex.} \]

stability, and smaller action space dimensionality.

Negative rewards can be important in environments where safe interaction is critical, however, while

\[ \text{learning a task with a scalar reward signal, they can make the learning of the task more complex.} \]

Figure 5 (a) shows the average force penalty (over 10 evaluation episodes) during the evaluation of

\[ \text{the wiping task. The data indicates that the variable action spaces generated substantial negative} \]

reward, even after 2 million training steps. It can be seen that the penalty is decreasing over time,

\[ \text{yet it remains a significant portion of the total accumulated reward. A driving factor for this fact} \]

is the high impedance of the variable action spaces, as can be seen in Figure 6. The wiping task

\[ \text{average translational stiffness, orientational stiffness, and dangerous contact forces can be seen in} \]

the third row. The impedance of translational impedance of AFORCE is significantly lower than of the variable action spaces. The result are seen in (c), the contact forces of AFORCE stay below 75 N, while the variable action spaces in some cases generate contact forces of above 500 N. Surprisingly, low + force has more dangerous contact forces than AFORCE. This might stem from the fact that it

\[ \text{has to hit the table with a higher velocity to align the tool with the table, however, more experiments} \]

are required for validation. We believe that a similar argument can be made for the other two

\[ \text{environments (see in the first and second row of Figure 6). The translational stiffness of the variable} \]

stiffness is higher than of AFORCE which correlates with the higher contact forces as seen in column

\[ \text{(c).} \]

Considering contact stability, the orientational stiffness in the wiping (see Figure 6 (b) bottom row)
task shows the strength of the bio-inspired design of AFORCE. While variable action spaces have

\[ \text{about the same stiffness in all three rotational directions, AFORCE keeps the orientation around z} \]

\[ \text{Table 3: Control parameters of the low-level adaptive policy for simulated robot experiments.} \]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input min/max</td>
<td>([-1, 1])</td>
</tr>
<tr>
<td>Output min/max ([\text{m}])</td>
<td>([-0.05, 0.05])</td>
</tr>
<tr>
<td>(x_{\text{min}} \text{[m]})</td>
<td>([-0.3, -0.35, 0.9])</td>
</tr>
<tr>
<td>(x_{\text{max}} \text{[m]})</td>
<td>([0.3, 0.35, 1.1])</td>
</tr>
<tr>
<td>(K_{\text{initial}} \text{[N/m, Nm/rad]})</td>
<td>(\text{diag}{100, 100, 100, 20, 20, 20})</td>
</tr>
<tr>
<td>(K_{\text{min}} \text{[N/m, Nm/rad]})</td>
<td>(\text{diag}{50, 50, 50, 10, 10, 10})</td>
</tr>
<tr>
<td>(K_{\text{max}} \text{[N/m, Nm/rad]})</td>
<td>(\text{diag}{500, 500, 500, 100, 100, 100})</td>
</tr>
<tr>
<td>(\alpha_{\text{Door}})</td>
<td>(\text{diag}{10.0, 10.0, 10.0, 10.0, 10.0, 10.0, 10.0})</td>
</tr>
<tr>
<td>(\mu_{\text{Door}})</td>
<td>(\text{diag}{0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2})</td>
</tr>
<tr>
<td>(\beta_{\text{Door}})</td>
<td>(\text{diag}{0.68, 0.68, 0.68, 2.5, 2.5, 2.5})</td>
</tr>
<tr>
<td>(\gamma_{\text{Door}})</td>
<td>(\text{diag}{0.009, 0.009, 0.009, 0.003, 0.003, 0.003})</td>
</tr>
<tr>
<td>(\alpha_{\text{Lift}})</td>
<td>(\text{diag}{10.0, 10.0, 10.0, 10.0, 10.0, 10.0})</td>
</tr>
<tr>
<td>(\mu_{\text{Lift}})</td>
<td>(\text{diag}{0.2, 0.2, 0.2, 0.2, 0.2, 0.2})</td>
</tr>
<tr>
<td>(\beta_{\text{Lift}})</td>
<td>(\text{diag}{2.35, 2.35, 2.35, 2.84, 2.84, 2.84})</td>
</tr>
<tr>
<td>(\gamma_{\text{Lift}})</td>
<td>(\text{diag}{0.045, 0.045, 0.045, 0.0028, 0.0028, 0.0028})</td>
</tr>
<tr>
<td>(\alpha_{\text{Wipe}})</td>
<td>(\text{diag}{10.0, 10.0, 10.0, 10.0, 10.0, 10.0})</td>
</tr>
<tr>
<td>(\mu_{\text{Wipe}})</td>
<td>(\text{diag}{0.2, 0.2, 0.2, 0.2, 0.2, 0.2})</td>
</tr>
<tr>
<td>(\beta_{\text{Wipe}})</td>
<td>(\text{diag}{1.2, 1.2, 1.2, 2.5, 2.5, 2.5})</td>
</tr>
<tr>
<td>(\gamma_{\text{Wipe}})</td>
<td>(\text{diag}{0.024, 0.024, 0.024, 0.003125, 0.003125, 0.003125})</td>
</tr>
</tbody>
</table>

\[ \text{Table 4: Force penalties during training of the simulated tasks.} \]

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Door</td>
<td>Wipe</td>
</tr>
<tr>
<td>AFORCE</td>
<td>25.13 17.76</td>
</tr>
<tr>
<td>variable</td>
<td>149.13 460.72</td>
</tr>
<tr>
<td>variable + force</td>
<td>– 449.88</td>
</tr>
<tr>
<td>low + force</td>
<td>– 17.16</td>
</tr>
</tbody>
</table>
Figure 6: Additional analysis of impedance and contact forces in training episodes of three manipulation tasks. (a) The average translational of the manipulator during training episodes. (b) The average orientational stiffness during training episodes. (c) Contact forces arising during training episodes above 50 Newton.

significantly lower than the other action spaces. The rotation around the z-axis is less important for the task and the error is generally smaller. In contrast, for wiping motions in x direction, the stiffness $K_{ay}$ is especially important to keep full contact with the table. Due to the higher error, AFORCE adapts the stiffness in that direction to higher values to align the tool with the table. Due to the higher error, AFORCE adapts the stiffness in that direction to higher values to align the tool with the table. By inspecting Figure 5 (c), we see that the action space $low + force$ achieves maximum episodic reward during evaluation after around 75 hundred thousand steps. However, it does not manage to achieve such a reward consistently over all 10 evaluation episodes per evaluation cycle. This is evident from Figure 5 (b), where it can be seen that the average wiped markers at that point is still around 75.

Appendix D  High-level Policy Hyperparameters

Table 5 lists all hyperparameters of the SAC [48] high-level policy, implemented in pytorch [49], we used for the simulated tasks.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batch size</td>
<td>1024</td>
</tr>
<tr>
<td>Replay buffer capacity</td>
<td>$1 \times 10^6$</td>
</tr>
<tr>
<td>Discount $\gamma$</td>
<td>0.99</td>
</tr>
<tr>
<td>Hidden dimension</td>
<td>1024</td>
</tr>
<tr>
<td>Critic learning rate</td>
<td>$1 \times 10^{-4}$</td>
</tr>
<tr>
<td>Critic target update frequency</td>
<td>2</td>
</tr>
<tr>
<td>Critic soft target update rate $\tau$</td>
<td>0.005</td>
</tr>
<tr>
<td>Critic hidden layers</td>
<td>2</td>
</tr>
<tr>
<td>Critic hidden units</td>
<td>1024</td>
</tr>
<tr>
<td>Actor learning rate</td>
<td>$1 \times 10^{-4}$</td>
</tr>
<tr>
<td>Actor update frequency</td>
<td>1</td>
</tr>
<tr>
<td>Actor log std bounds</td>
<td>[-10, 2]</td>
</tr>
<tr>
<td>Actor hidden layers</td>
<td>2</td>
</tr>
<tr>
<td>Actor hidden units</td>
<td>1024</td>
</tr>
<tr>
<td>Temperature learning rate</td>
<td>$1 \times 10^{-4}$</td>
</tr>
<tr>
<td>Init temperature</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 5: The hyperparameters of SAC we used as the high-level policy of the simulated tasks.