Variable Impedance Control in End-Effector Space:
An Action Space for Reinforcement Learning in Contact-Rich Tasks

Roberto Martín-Martín, Michelle A. Lee, Rachel Gardner, Silvio Savarese, Jeannette Bohg, Animesh Garg

Abstract— Reinforcement Learning (RL) of contact-rich manipulation tasks has yielded impressive results in recent years. While many studies in RL focus on varying the observation space or reward model, few efforts focused on the choice of action space (e.g. joint or end-effector space, position, velocity, etc.). However, studies in robot motion control indicate that choosing an action space that conforms to the characteristics of the task can simplify exploration and improve robustness to disturbances. This paper studies the effect of different action spaces in deep RL and advocates for variable impedance control in end-effector space (VICES) as an advantageous action space for constrained and contact-rich tasks. We evaluate multiple action spaces on three prototypical manipulation tasks: Path Following (task with no contact), Door Opening (task with kinematic constraints), and Surface Wiping (task with continuous contact). We show that VICES improves sample efficiency, maintains low energy consumption, and ensures safety across all three experimental setups. Further, RL policies learned with VICES can transfer across different robot models in simulation, and from simulation to real for the same robot. Further information is available at https://stanfordvl.github.io/vices.

I. INTRODUCTION

Robot manipulation and control tasks are naturally addressed in different action spaces. For a walking robot, it is important to directly control contact interactions to avoid slippage [35]. In contrast, for a tennis swing, it is important to track and control position, velocity, and at times the acceleration of the end-effector [11]. For a surface-to-surface alignment task, minimizing the moment around a contact is important for robustness [15]. Tackling all these tasks requires solving two subproblems: (1) the generation of reference signals (desired contacts, trajectories, moments, etc.), and (2) the tracking of these signals.

Control systems for robots can be structured with two feedback control loops that address the aforementioned subproblems: an outer loop controller that generates a time-varying reference trajectory, and an inner loop controller that tracks this trajectory. Let us refer to the outer loop as a functional map from observations to reference signals $g(o) : \mathcal{O} \to \mathcal{A}$, and the inner loop as a map from reference signals to actuation commands $f(a) : \mathcal{A} \to \mathcal{U}$. The combined control law becomes: $u = f \circ g(o)$, where $o \in \mathcal{O}$ is some observation, $a \in \mathcal{A}$ is an abstract action providing a reference signal in some space, and $u \in \mathcal{U}$ is the control command sent to the robot's actuators to track this reference.

While control theory provides a vast repertoire of strategies to map from reference signals to actuation commands ($f(\cdot)$ implementations), a key problem in robotics is to generate a viable reference in a suitable space given raw sensory
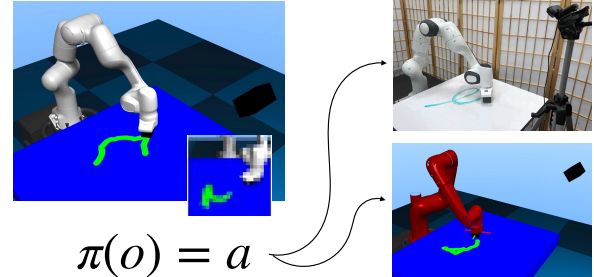
Fig. 1: Depending on the characteristics of a task, different action spaces for policy learning are better suited than others; *Variable impedance control in end-effector space (VICES)* is efficient to learn both free-space motion and contact-rich tasks. Through the compensation of the robot's kinematics and dynamics, the policies learned for one robot in simulation transfer seamlessly to other robot instances and to the real robot.

observations, i.e. modelling the function $g(\cdot)$ and deciding on the interface to $f(\cdot)$. Once the interface has been defined (e.g. forces, positions, contact points, etc.), the generation of reference signals by $g(\cdot)$ can be addressed in multiple ways, ranging from hand-tuned state machines [15, 41], trajectory optimization [14, 45], imitation learning [5, 12, 19, 38], or reinforcement learning (RL) [9, 23].

Recent research in RL has focused on "observations-to-torques" [25], which is akin to merging $f(\cdot)$ and $g(\cdot)$ into a single learned model. Other methods use higher-level action spaces, such as joint space commands (e.g. position or velocity) [7, 8, 46, 48] or task space commands (e.g. end-effector poses, force or fixed impedance) [13, 23]. These works typically focus on the effects of choosing an observation space $\mathcal{O}$ on the learning process and rarely justify the choice of action spaces, $\mathcal{A}$. However, the choice of action space defines the quantity around which the inner control loop is closed, and by extension the space wherein tracking error is minimized. This critically impacts robustness and task performance as well as learning efficiency and exploration.

Moreover, previously proposed action spaces do not necessarily create suitable references for some contact-rich tasks. Consider the task of wiping a board, wherein a robot must control forces in some directions (to keep pressing against the board) and motion in others. The physical constraints of the task dictate in which axes the robot should be stiff and in which it should be compliant. This can often be time-varying. Manual specification of the task constraints is not a scalable solution for the variety of contact-rich tasks the robot may need to perform, and for some tasks, manual specification may be non-trivial.

This paper studies how the selection of an interface between $g(\cdot)$ and $f(\cdot)$, (the space $\mathcal{A}$) affects RL as a method to learn the mapping from observations to reference signals, $g(\cdot)$, and presents the first empirical study comparing the most common choices for the $\mathcal{A}$ in contact-rich manipulation.

We argue that an action space that captures motion and impedance in end-effector space can enable efficient learning of such tasks. We evaluate joint position, joint velocity, joint torque, joint variable impedance, as well as fixed and variable impedance in end-effector space. The choice of action space should be guided not only by the robot model but also by prior knowledge of the task. Hence, we compare action spaces across tasks with varying degrees of task-space constraints, i.e., Path Following with no contact (*path following*), manipulation of constrained mechanisms (*door opening*), and continuous unconstrained contact (*surface wiping*).

Moreover, we introduce *variable impedance control in end-effector space (VICES)* and advocate this action space for Deep RL algorithms applied to contact-rich manipulation. We show that policies defined in VICES improve sample efficiency for exploration in RL, energy efficiency, and reduce peak forces. Thanks to the classical dynamically consistent operational space formalism [17], we observe that policies learned in end-effector space are also more robust to transfer across robots with significant differences in dynamics whether in simulation or the real world.

II. RELATED WORK

Robot Motion Control: Compliant control of a robot manipulator enables adaption to uncertainty in the environment (e.g. exact shape of a surface, or kinematic constraints of mechanisms) during contact-rich tasks. However, certain tasks require direct control of the contact interactions (e.g. don't apply too much force when wiping a window). Previous abstractions have divided dimensions of the task into those that are controlled kinematically (through position and velocity) and dynamically (through force and torques) [17, 20, 29]. However in practice, the hard decoupling requires a level of knowledge about the task that is not always available [3]. Impedance control [10] allows safe robot contact manipulation with an unknown environment by explicitly controlling the amount of force the robot exerts when it deviates from a given kinematic goal. Therefore, it alleviates the need for perfect knowledge or hard separation between the dynamic and kinematic task dimensions.

However, different phases of a manipulation task may require a dynamic balancing between kinematic and dynamic control. Existing methods address it by scheduling variable impedance gains to maintain stability or safety for a given kinematic trajectory [27, 30, 37]. However, these methods assume that a reference trajectory is given.Instead, we propose to directly predict both end-effector displacement (reference) and variable impedance gains based on observations.

Action Spaces in Learning from Demonstrations: LfD derives a task policy based on demonstrations provided by some other agent(s) [2]. If the demonstrations do not perfectly overlap, a possible approach is to derive a policy that imitates the mean motion of the demonstration set, and varies the stiffness according to the coherence of the trials [6, 21] or according to the force sensed during kinesthetic replay [1]. Similar use of variable impedance as an action representation for LfD has been demonstrated to be successful for adaptive grasping [26], manipulation of deformable objects [22], and co-adaptation to human workers [36]. However, the specification of impedance in the demonstration only reflects variability in demonstrations trajectories but not

the underlying task constraints imposed by the environment nor the force profiles required for the task. This approach is also restricted to tasks where expert demonstrations are feasible, and hence is limited in application to kinematic tasks with phases that require different level of precision.

Reinforcement Learning: In the field of model-based reinforcement learning, Kim et al. [18] proposed a method to learn the parameters of a variable impedance position controller in end-effector space based on the equilibrium point formalism. They demonstrated the convergence, robustness and energy efficiency of their method on simulated manipulation task with a two DoF planar arm. However, their method requires an initial trajectory which is not always available.

Similar to our method, Buchli et al. [4] apply policy improvements with path integrals (PI^2) [44] to refine initial trajectories and learn variable scheduling for the joint impedance parameters. They demonstrate that energy consumption can be optimized while achieving a task using variable impedance. However, they use joint space as their action space, which limits the transferability of the learned behaviors to different robots and the optimality of the trajectories in the space of the task. Also, their method requires an initial estimate of the solution to start the iteration.

Rey et al. [34] propose an approach to simultaneously learn kinematic trajectories from demonstrations and variable impedance in task space from exploration. They use Gaussian Mixture Regression as representation for the policy and demonstrate their method in simulation and in one real-world planar task with one single stiffness parameter. [13] proposed a method to refine given trajectories with additional force profiles using PI^2[44] and readings from a force-torque sensor. We aim to achieve dynamic behavior without direct force loop control by using impedance to learn both trajectories and variable stiffness profiles.

It is worth noting that all previous approaches have boostrapped learning with initial demonstrations while we explore learning from scratch to better understand how fast policies converge. Viereck et al. [47] studied how to incorporate control structure to learn hopping policies for one-legged robot with RL. They use an optimal controller for fixed task conditions and learn to imitate its policy with neural networks to generalize to new task conditions. Interestingly, they compare two network architectures outputting signals in different action spaces: directly desired torques or full feedback parameters and desired configuration, which is transformed into torques with an analytic function. Their experiments show that this second action space is best suited for the hopping task with intermittent contact and adds interpretability to the network output. We study a set of analytic functions (controllers) that map policy actions to low level robot commands for robot manipulation in three tasks with different contact properties.

With related motivation to ours, Peng et al. [33] studied the importance of different action representations in RL for the task of locomotion. Similar to us, they aimed to shed light on the best action space to be used, but in their case they focused on imitation learning in bipedal motion of simulation agents. We would like to provide similar insights in the more complex contact-rich robot manipulation domain and include preliminary studies of transfer to real world.

III. REINFORCEMENT LEARNING

The goal in reinforcement learning is to find a policy π, that selects actions based on current observations so as to maximize the expected reward obtained from interactions with the environment [42]. We assume that the underlying problem can be modelled as a discrete-time continuous Markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \rho)$, where $\mathcal{S}$ is a continuous state space, $\mathcal{A}$ is a continuous action space, $\mathcal{P}$ is a Markovian transition model defining the probability of transitioning between states for a given action $P(s'|s,a)$, $\mathcal{R}$ is a reward function $\mathcal{R}(s,a) = r \in \mathbb{R}$, $\gamma \in [0,1)$ is a discount factor (for infinite horizon problems) and ρ is the initial state distribution. When π is probabilistic it represents the probability of the action, a, given the state, s: $\pi(a|s) = p(a|s)$, and $\pi(\cdot|s)$ is the density distribution over $\mathcal{A}$ in state s.

Alternatively, we can assume the state is not directly observable and learn a policy $\pi(a|o)$ conditioned on observations $o \in \mathcal{O}$ instead of latent states $s \in \mathcal{S}$. Herein, the agent following the policy π obtains an observation o_t at time t and performs an action a_t, receiving from the environment an immediate reward r_t and a new observation o_{t+1}. Assuming the policy is parameterized by θ, a policy gradient algorithm optimizes θ to maximize that the expected future return:

$$\theta^* = \underset{\theta}{\arg\max} \, J(\theta, \rho) = \underset{\theta}{\arg\max} \, E\left[\sum_t \gamma^t r_t\right] \quad (1)$$

These algorithms are based on the policy gradient theorem that states: $\frac{\delta J}{\delta \theta} = \sum_s \mu_{\pi_\theta}(s,\rho) \sum_a \frac{\delta \pi_\theta(a|s)}{\delta \theta} Q_{\pi_\theta}(s,a)$, where $\mu_{\pi_\theta}(s,\rho) = \sum_t \gamma^t P(s_t = s|\rho)$ and Q_{π_θ} is the action-value function associated to the current policy π_θ.

There are multiple algorithmic solutions based on the policy gradient theorem that allow us to represent the policy with a deep neural network, e.g. Trust Region Policy Optimization (TRPO) [39], Deep Deterministic Policy Gradients (DDPG) [28], or Advantage Actor-Critic (A2C) [31]. In our evaluation of different action spaces for policies, we will use Proximal Policy Optimization (PPO) [40]. The evaluation of the sensitivity of different algorithms to the action space is deferred to future work.

IV. ACTION SPACES IN RL FOR ROBOT MANIPULATION

Relating the formalisms we introduced in Sec. I and III, π corresponds to $g(\cdot)$, the function that maps observations to reference actions in some space $\mathcal{A}$, assuming that another function $f(\cdot)$ will map these actions to low level control commands, $u \in \mathcal{U}$.

We note that the RL algorithms in Sec. III are agnostic to choice of action space $\mathcal{A}$. In practice, the most common $\mathcal{A}$ in RL for robot manipulation are a) joint torques [25], b) joint velocities [7, 46, 48], c) joint positions [8], and d) end-effector position [23, 43] possibly with orientation. The most common lowest level control commands (and the one we assume for our underlying physical agent) are joint torques, $u = \tau \in \mathcal{T}$. Joint torques are safer than positions and velocities for contact-rich tasks in unstructured environments, because the forces the robot will apply on the environment are limited by the specified desired torques.

Manipulation tasks can seldom be solved solely by only controlling motion since there are tasks that contain contact and force constraints (e.g. the adaptation required to manipulate an articulated object or the minimum force to press
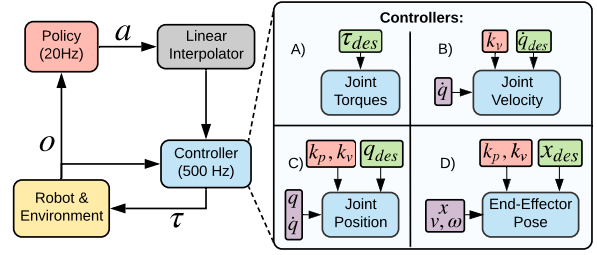

Fig. 2: We trained policies that output actions, a, on different action spaces, $\mathcal{A}$ at 20 Hz; Depending on the action space, the appropriate controller takes in the desired control signal, the control parameters (fixed or given by the policy), and the current state and outputs actuation commands $u \in \mathcal{U}$ at 500 Hz, which for our robots correspond to joint torques τ. We interpolate linearly between consecutive low-frequency policy actions to generate smoother high-frequency controller signals (and parameters, if controlled by policy).

while we wipe a surface) [3, 17, 20, 29]. To succeed in these tasks, the robot needs to dynamically modulate the exerted force on the environment through the torques on each joint.

To map between action space $\mathcal{A}$ and actuation space $\mathcal{U}$, we can define analytic parameterized functions (i.e. *controllers*), $f_\kappa(a, s_{robot}) : \mathcal{A} \times \mathcal{S}_{robot} \to \mathcal{U}$, that transform the output of the policy from the action space to the space of control commands depending on the current state of the robot, $s_{robot} \in \mathcal{S}_{robot}$. The parameters of these functions, κ, can be made part of the policy action space so that the agent has full controllability on the manipulation behavior [4].

In the following, we will introduce the different choices of analytic controllers f_κ, we use to map policy actions from commonly used policy action spaces into joint torques.

Joint Torques: When the policy directly outputs desired joint torques, i.e. $a = \tau_{\text{des}}$, the function that transforms to robot commands is simply (jt = *joint torques*):

$$u = f_\kappa^{\text{jt}}(a = \tau_{\text{des}}) = \tau_{\text{des}} \quad (1)$$

Joint Velocities: When the policy outputs reference joint velocities $a = \dot{q}_{\text{des}}$, the function to map to joint torques is (jv = *joint velocities*):

$$u = f_\kappa^{\text{jv}}(a = \dot{q}_{\text{des}}, s_t = \dot{q}) = k_{\text{v}}(\dot{q}_{\text{des}} - \dot{q}) \quad (2)$$

where we close the loop around $\dot{q}$, the current joint velocity (state), and k_{v} is a vector of proportional gain (parameter κ).

Joint Positions: For policies that output reference joint positions, $a = q_{\text{des}}$, it is most straightforward to use a proportional-derivative (PD) controller that generates torques that increase with the joint position error and decrease with the current joint velocity. We also remove the dynamic effects of the mechanism by scaling the torques with the inertia matrix, M [17]. The function to transform reference joint positions to joint torques is thus (jp = *joint positions*):

$$u = f_\kappa^{\text{jp}}(q_{\text{des}}, q, \dot{q}) = M\left[k_{\text{p}}\Delta q - k_{\text{v}}\dot{q}\right] \quad (3)$$

where $\Delta q = q_{\text{des}} - q$ is the difference between current and desired joint configurations, which can be used as an alternative policy action space. k_{p} and k_{v} are vectors of proportional and derivative gains (parameters κ).

End-Effector Pose: In the cases where the policy outputs the desired 6-D pose $\in \mathbb{SE}(3)$ of the robot in end-effector space, $\mathbf{x}_{\text{des}}$, we can use an impedance-based PD controller to first derive an end-effector space acceleration to move

(a) Path Following   (b) Door Opening   (c) Surface Wiping

Variable Impedance Control in End-Effector Space | High Impedance Control in End-Effector Space | Medium Impedance Control in End-Effector Space | Low Impedance Control in End-Effector Space | Variable Impedance Control in Joint Space | Joint Position | Joint Velocity | Joint Torque
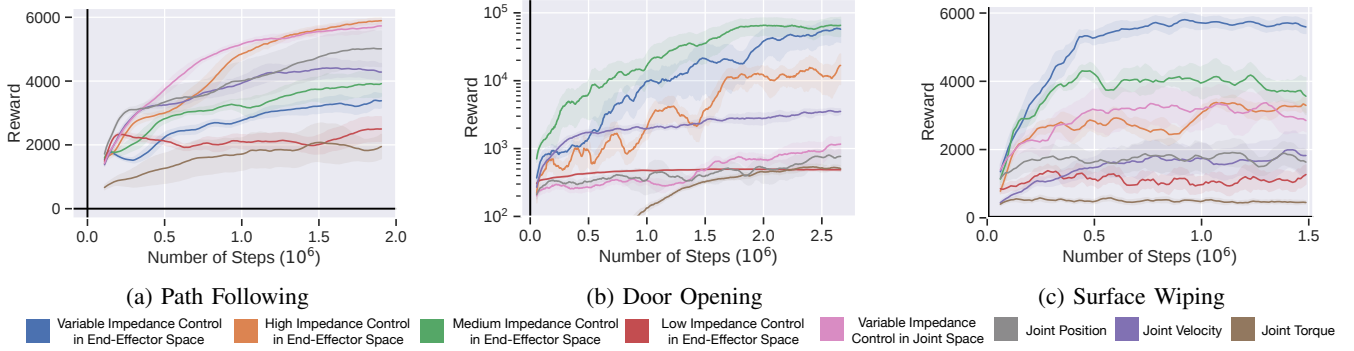
Fig. 3: Training curves for a) Path Following (free space), b) door opening (kinematic constraints), and c) surface wiping (contact rich) tasks; The plots depict mean and standard deviation of five learning processes with different random seeds; Tasks without contact or with kinematic constraints (Path Following and door opening) do not require variable impedance as action space to achieve high reward; In the contact-rich task (surface wiping) the policy using variable impedance in end-effector space achieves higher reward because it learns to adapt correctly the amount of force applied to the tasks constraints

towards the goal. To do that, $\mathbf{x}_{\text{des}}$ can be decomposed into desired position, $p_{\text{des}} \in \mathbb{R}^3$, and desired orientation, $R_{\text{des}} \in \mathbb{SO}(3)$. In the impedance-based PD controller, the end-effector acceleration increases with the difference between desired end-effector pose and current pose, p and R, and decreases with the current end-effector velocity, v and ω.

We then compute the robot actuations (joint torques) to achieve the desired end-effector space accelerations leveraging the kinematic and dynamic models of the robot with the dynamically-consistent operational space formulation [16]. First, we compute the wrenches at the end-effector that correspond to the desired accelerations, $f \in \mathbb{R}^6$. Then, we map the wrenches in end-effector space f to joint torque commands with the end-effector Jacobian at the current joint configuration $J = J(q)$: $u = J^T f$.

Thus, the function that maps end-effector space position and orientation to low level robot commands is (ee = *end-effector space*):

$$
\begin{aligned}
u &= f_\kappa^{\text{ee}}(p_{\text{des}}, R_{\text{des}}, p, R, v, \omega) \\
&= J_{\text{pos}}^T[\Lambda^{\text{pos}}[k_{\text{p}}^{\text{pos}}(p_{\text{des}} - p) - k_{\text{v}}^{\text{pos}}v]] + \\
&\quad J_{\text{ori}}^T[\Lambda^{\text{ori}}[k_{\text{p}}^{\text{ori}}(R_{\text{des}} \ominus R) - k_{\text{v}}^{\text{ori}}\omega]]
\end{aligned}
\tag{4}
$$

where Λ^{pos} and Λ^{ori} are the parts corresponding to position and orientation in $\Lambda \in \mathbb{R}^{6 \times 6}$, the inertial matrix in the end-effector frame that decouples the end-effector motions, J_{pos} and J_{ori} are the position and orientation parts of the end-effector Jacobian, and $\ominus$ corresponds to the subtraction in $\mathbb{SO}(3)$. The difference between current and desired position ($\Delta^{\text{pos}} = p_{\text{des}} - p$) and between current and desired orientation ($\Delta^{\text{ori}} = R_{\text{des}} \ominus R$) can be used as alternative policy action space, $\mathcal{A}$. $k_{\text{p}}^{\text{pos}}$, $k_{\text{v}}^{\text{pos}}$, $k_{\text{p}}^{\text{ori}}$, and $k_{\text{v}}^{\text{ori}}$ are vectors of proportional and derivative gains for position and orientation (parameters κ), respectively.

Variable Impedance End-Effector Space (VICES):

Thus far, we have defined transformations between policy actions a and robot commands u are parameterized with κ. In these cases, parameters κ are manually specified. We observe that it is beneficial to augment the action space with these parameters to give the agent full control of the behavior. As discussed in Sec. II, this idea has been previously explored in joint space for k_{p} and k_{v} [4]. In this paper, we propose to also turn the parameters of the end-effector space function ($k_{\text{p}}^{\text{pos}}$, $k_{\text{v}}^{\text{pos}}$, $k_{\text{p}}^{\text{ori}}$, and $k_{\text{v}}^{\text{ori}}$) into policy outputs. We term this action space as *Variable Impedance End-Effector Space* (VICES).

It enables the policy to learn both to predict the end-effector pose as a trajectory reference as well as to dynamically adapt the impedance gains along each of the six axes (rotation and translation) according to the phase of the task.

V. EXPERIMENTS

We conduct experiments in three application domains: a) free space Path Following [4], b) manipulation of articulated mechanisms [18] and c) surface wiping [24, 32]. These tasks are not only relevant applications in robotics, but also span different levels of task constraints from free motion to highly constrained contact-rich manipulation, which allows us to evaluate and compare the characteristics of the different action spaces for policy learning.

For these three tasks and for each of the evaluated action spaces we aim to answer the following questions: is the action space suitable for model-free RL? Is the learned policy physically efficient? Does the policy learned with a simulated robot transfer to a different simulated robot? Does a policy learned in simulation transfer to a real robot? To answer these questions we will use the following metrics and tests:

1) **Sample efficiency and task completion**: samples required for the policy to succeed in the task and/or converge

2) **Physical efficiency**: energy consumed by the robot when using the trained policy. We assume a proportional relationship between joint torques and electric power

3) **Physical effort**: wrenches applied to the environment by the trained policy during contact-rich manipulation tasks

4) **Transferability between robots**: does a robot achieve the task using a policy trained on a different robot?

5) **Sim-to-real transfer** of contact-rich policies: does a real robot achieve the task using a policy trained in simulation?

Our control framework is outlined in Fig 2. In all experiments our policies output actions ($a \in \mathcal{A}$) at 20 Hz, while we send joint torque commands ($u \in \mathcal{U} = \mathcal{T}$) to the robot at 500 Hz. To generate torque commands at a higher frequency, the controllers use the constant desired goal from the policy while updating the current state of the robot, s_{robot}. In order to ensure smooth robot commands and generated motions, in all of our controllers we interpolate linearly between policy commands at consecutive time steps.

A. Free-Space Motion - Path Following

Setup. In this experiment we aim to measure the properties of different action spaces for tasks that do not involve any

contact with the environment. Agent's goal is to follow a trajectory in free-space passing through four via-points. The via-points are placed on a virtual plane in front of the agent at a constant distance along the x axis. The order and location of the via-points are fixed. We measure success as the fraction of the four via-points that are passed through.

This setup is a more complex version of the one via-point trajectory of Buchli et al. [4]. This task can be solved kinematically without impedance control. However, we found that controlling the compliance of the robot could still offer benefits in this setup.

Reward Model. This task is trained in two phases : a first phase of task completion and a second phase of energy optimization. In the first phase, the agent is rewarded only to complete the task: to pass through the four via-points. In the second phase, the trained models from previous phase are further trained with the additional objective of optimizing their motion to reduce energy consumption.

In the first phase of the experiment, the agent is rewarded when it hits a via-point (it gets closer than $d_{th} = 5$ cm). To help guide exploration, we also provide a small dense reward inversely proportional to the distance to the next via-point in the trajectory. Since the episodes continue after the task is completed, a task-completion bonus proportional to the remaining time steps was introduced to discourage the robot from unnecessarily extending the duration of the task. We train policies with this reward using the different action spaces to evaluate if they can learn to follow the free-space trajectory.

In the second phase of the experiment , we explore if the action spaces can optimize for the additional objective of minimizing energy consumption without decreasing the quality of the first objective (passing through the via-points). We include an energy consumption penalty to the previously defined reward function. To evaluate energy consumption we assume that the torques from the motors are proportional to electric current and the voltage is constant, and thus the amount of electric power scales proportional to the torque and the energy is its time integral.

Observations. We use as observations the pose and velocity of the end-effector in the robot reference frame, as well as the location of the via-points (and whether each one has been checked).

Evaluation. We first evaluate each of the different action spaces in simulation, using a simulated Panda robot agent with five different random seeds. In the first phase of the experiment, we measure sample efficiency (reward as function of the iterations) and level of completion of the task (number of via-points crossed). In the second phase of the experiment, we also measure the total energy consumption and task success. In both phases we evaluate how the original trained policies transfer between robots.

Sample Efficiency and Task Completion. Fig. 3 (*a*) shows the training curves for policies in each of the action spaces. All policies except the ones that output reference joint torques and end-effector poses resolved with a fixed low impedance controller were able to achieve the goal of the task: checking all 4 via-points (see Fig. 5a). For the policies that achieve the task, the differences in reward value after convergence is simply a consequence of the termination bonus: some action spaces (e.g. desired end-effector poses resolved with high fixed impedance) allow for faster motion and thus faster
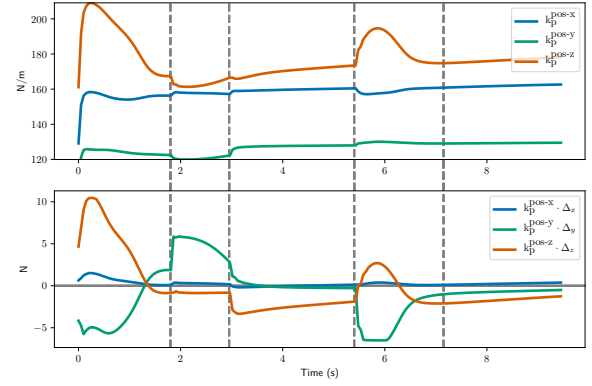

Fig. 4: Free space Path Following: Time evolution for a single episode of a policy using variable impedance in end-effector space; Impedance (stiffness, k_p) changes as via-points are checked as indicated by the vertical dotted lines; The four via-points of the trajectory are aligned with the same x coordinate and thus the policy learns to not exert force in that dimension (blue curves); After checking a via-point, the policy increases stiffness (k_p) and combines it with the right desired displacement ($k_p \Delta x$) to generate motion in the direction to reach the next via-point of the trajectory

completion of the task

We gain insights on how an RL policy exploits VICES for this task by observing the stiffness and damping over the course of an episode. Fig. 4 depicts the commands (the desired stiffness and the product of desired stiffness and delta position) from the policy trained with VICES for one episode after the first stage of training (before applying the energy penalty). The policy exploits the impedance (stiffness) to reach each via-point in the different portions of the trajectory. As it checks each via-point (indicated by the vertical bars in the figure), the impedance changes in the appropriate dimension to move quickly to the next via-point with enough stiffness to avoid overshooting.

Physical Efficiency. We evaluate the physical efficiency of policies in different action spaces by comparing the total energy consumption of the agents at the end of the first phase and of the second phase of our experiment, where we add the energy penalty. We found that the policies using variable impedance in end-effector space as action space were the only end-effector space policies that consistently improved energy efficiency while maintaining task performance. Both the medium and high fixed impedance models became unstable, since the action space does not have sufficient degrees of freedom to optimize the motion to reduce energy consumption while still achieving the trajectory task. Note that since the low impedance model never achieved the task, it was not evaluated with energy penalties.

In joint space, the policies outputting actions in variable impedance space were also able to reduce the energy consumption significantly more than the controllers with fixed impedance, as expected [4]. They also reduced more energy than policies outputting variable impedance in end-effector space. This reflects that the policies outputting joint space commands resulting from the first phase of our experiment solved the task much faster (with higher energy consumption) than their end-effector counterparts and therefore had much more room for improvement when optimizing for energy efficiency. Therefore, the difference in absolute energy optimization between policies in joint and in end-effector space is an artifact of the difference in
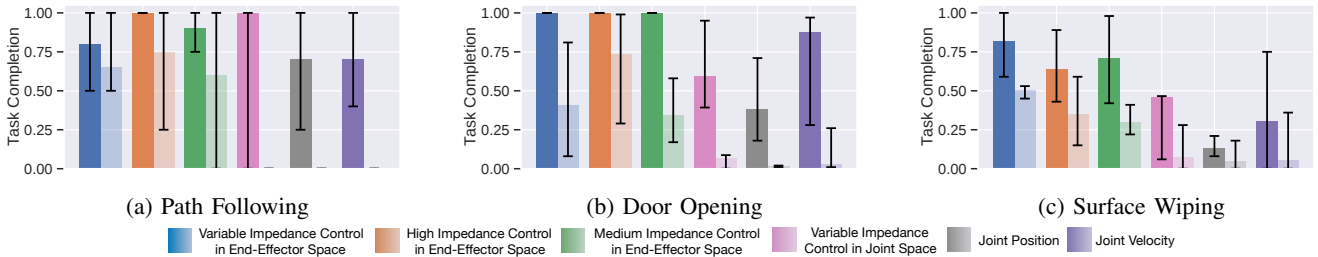
(a) Path Following   (b) Door Opening   (c) Surface Wiping

Variable Impedance Control in End-Effector Space   High Impedance Control in End-Effector Space   Medium Impedance Control in End-Effector Space   Variable Impedance Control in Joint Space   Joint Position   Joint Velocity

Fig. 5: Evaluation of the the task performance and transfer of policies learned on a robotic platform (Panda, dark color bars) to a different platform (Sawyer, lighter color bars) for a) Path Following (free space), b) door opening (kinematic constraints), and c) surface wiping (contact rich) tasks; Transfer is between simulated robots without retraining; The error bars indicate the range of percent task performance across the 5 seeds; Policies trained in the joint torque action space and in the end-effector low impedance action space did not learn to achieve the task in the original embodiment (Panda) and are not part of the transfer evaluation; Policies in joint space are unable to transfer between embodiments while policies in end-effector space transfer better because they are not dependant of the robot dynamics; Policy transfer is harder for tasks with increased contact with the environment, as the agent is more likely to hit joint limits

magnitude between end-effector space delta position limits and joint space delta angle limits (i.e., the joint space agents were originally allowed to move more at each time step).

Transferability. We also evaluate how policies using different action spaces transfer in simulation from one robot to another through zero-shot transfer from the Panda robot to the Sawyer robot. The results are depicted in Fig. 5a. As expected, we observed that after convergence only the policies using fixed and variable impedance in end-effector space could transfer directly between robots. The joint-space policies were not able to transfer due to the very different kinematics and dynamics of the two robot platforms. By using end-effector space control, we factor out the effects of the embodiment from the policy learning problem.

B. Manipulation of Constrained Mechanisms - Door Opening

Setup. In this task, the robot has to learn how to manipulate a one DoF constrained mechanism, a door, to a specific configuration. The agent is equipped with a two-finger gripper it can use to hold the door handle. The door handle is a bar attached vertically on the door leaf. The gripper is closed, leaving a space between the fingers to cage the door handle while still allowing for rotation between handle and gripper.

We ensure that the agent learns to interact in a controlled and safe manner. Hence instead of maximally opening the door, we set the goal to manipulate the door into a desired joint configuration ($\theta_{goal} = 60°$). We measure success as the fraction of the total desired joint state achieved by the robot: $1 - \frac{|\theta_{door} - \theta_{goal}|}{\theta_{goal}}$.

Reward Model. We reward the agent when the door joint gets closer to the desired configuration. We provide additional constant reward if the configuration of the door is very close to the desired value (less than 5°). We penalize forces and torques exerted on the environment that go beyond the physical payload of the robot (40 N). We also penalize the agent for colliding with the environment with links other than the gripper and for going beyond its joint limits. For safety, the episode terminates when joint limits are violated.

Observations. We use as observation the pose and velocity of the robot's end-effector in the robot reference frame, as well as the door's angle and angular velocity.

Evaluation: We evaluate the different action spaces in simulation. We train an agent with a Panda robot embodiment for each action space with five different random seeds.

Sample Efficiency and Task Completion. We first evaluate the different action spaces on their sample efficiency of learning the door-manipulation. The training results are depicted in Fig. 3, middle. The task success results for the door task can be found in Fig. 5b.

We observe that policies that output end-effector space actions (with medium, variable, and high impedance) outperform policies in all other action spaces, in terms of achieving close to 100% task success rate and higher rewards. In end-effector space, the policy resolved with an impedance controller with fixed medium stiffness and damping is able to learn the task at a faster rate than the variable impedance controller, as it is initialized with a suitable impedance to operate the door with the defined friction. However, policies outputting actions in the both aforementioned spaces reach similar rewards and task success rates at the end of training, as the policies that can vary impedance end up learning a suitable impedance for the task.

While the policies resolved with an impedance controller with fixed high impedance parameters also achieve on average 100% task success rate, their rewards are lower because they exert higher forces in the environment that is penalized. The policies resolved with an impedance controller with fixed low impedance parameters are not able to learn the door opening task because they cannot exert high enough forces to overcome the friction of the door and move it. The policies outputting joint velocity actions can reach up to 75% task success, but the rewards are much lower than policies in VICES, as they often reach joint limits while opening the door. Policies outputting other joint-space actions (torques, positions) are unable to learn to exert enough force to open the door without reaching joint limits.

Transferability. We also evaluate the ability of policies in different action spaces to transfer from the Panda robot to the Sawyer robot. The results are shown in Fig. 5b, in lighter colors. Transferring policies for the door opening task is more complex than for the free-space Path Following task because the different robots' kinematics lead to very different task-space limitations, as well as very different joint limit constraints. Similar to results in the other tasks, policies trained in joint space are unable to transfer, since the kinematics of the robots differ substantially. The end-effector space policies are able to transfer much more successfully, as the end-effector space policies are able to abstract away the dynamics and kinematics of each specific robot model.

There is still a performance drop, as the policies in end-effector space do not learn to account for the robots' different kinematic constraints (i.e. joint limits).

C. Contact-Rich Manipulation - Surface Wiping

Setup. In this experiment the goal is to wipe a table whose surface location is unknown. The agents are equipped with a wiping tool, resembling a scrubber or a whiteboard eraser (see Fig. 1). In the simulator, the tool is modeled as a soft material that creates contact forces that increase proportionally to the penetration into the tool's surface. The material to wipe is modeled as a set of small elements of a color different from the table. The elements are placed randomly on the table surface to form a continuous "stain" and are marked initially as *unwiped*. They become *wiped* if the wiping tool passes through them, which also causes them to disappear visually. Note that since the elements are modeled as very thin (1 mm height) cylinders resting on the table's surface, the agent needs to press the tool against the surface so as to be able to wipe elements. Success rate is measured as the fraction of the elements wiped. The sliding coefficient of friction of the table acting along both axes of the tangent plane is sampled uniformly between 1.0 and 0.1. The initial location of the agent above the table is randomized.

Reward Model. The main reward comes from wiping off elements. We also provide additional reward for wiping off all the elements. Additionally, to help during the initial phases of exploration, we give the agent a small reward for maintaining contact with the table. Finally, since we aim to generate safe solutions that can directly be tested on the real robot, we slightly penalize the agent for applying forces over the payload of the real robot (40 N), and harshly penalize the agent for reaching joint limits or colliding with the table with parts other than the wiping tool. If such collisions occur, the episode ends and the tasks restarts.

Observations. There is no straightforward way to represent the state of a wiping task. Instead, we directly use visual observations: $48 \times 48 \times 3$ RGB images of the wiping scene generated in our simulator, and obtained from a camera on our real robot platform for the simulation-to-real transfer experiments. As in previous experiments, we also provide the pose and velocity of the end-effector.

Evaluation: We first evaluate the different action spaces on simulation. We train an agent with a Panda robot embodiment for each action space with five different random seeds.

Sample Efficiency and Task Completion. Fig. 3, right, shows the convergence of the agents with different action spaces. We observe that the agents with variable impedance in end-effector space converge faster and achieve higher reward. The higher reward is obtained thanks to a lower penalty for applying excessive force on the table since the agents can learn to appropriately adapt their stiffness. The mean force applied by the policies with variable impedance in end-effector space is 29 N, less than the robot's payload. Agents using other action spaces apply higher mean force or not enough to wipe the table. In terms of task completion, the results are depicted in Fig. 5c, dark colors. Policies outputting actions in VICES achieve the highest ratio of wiped units.

Transferability. We also evaluate if the policies learned with the Panda robot embodiment transfer directly to the Sawyer robot in simulation. Fig. 5c, depict the results of the policy
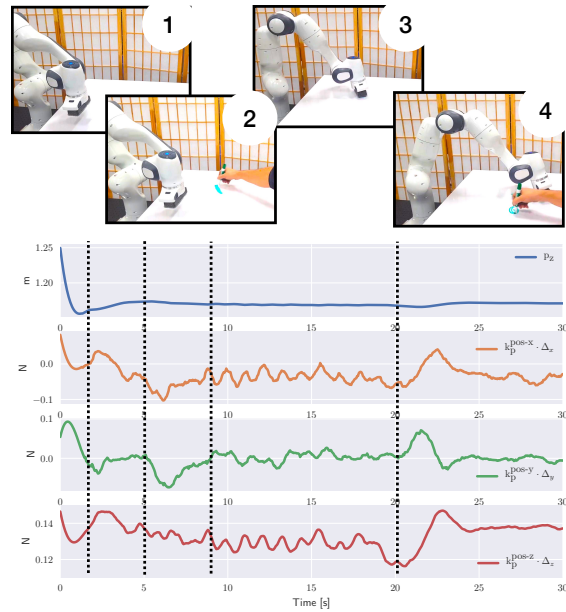

Fig. 6: Sim2Real transfer: we apply directly the policy learned in simulation with action in variable impedance in end-effector space to the real robot; pictures correspond to dotted vertical lines in the plots; 1) the robot pushes towards the table in a compliant manner (height over the table in the top plot, force in *z*, bottom plot); the experimenter marks different parts of the whiteboard, 2) and 4), and the robot reacts 3) moving towards the area following the forces in *x* and *y* (second and third rows)

transfer between robots. Policies trained in variable impedance in end-effector space transfer better than policies in any other space since the policy is independent of the robot embodiment. However, there is a significant drop in performance due to the different forces generated by the different embodiments.

Simulation-to-real transfer. In a final experiment we evaluate if the policies trained in simulation can be used on the real robot without any retraining. We use in our experiment the best performing simulation policy. The goal in the real world is to wipe a whiteboard painted with a marker. Since our focus is on the evaluation of the action space and not on learning a representation of the image, we convert the real images into fake simulated images by superimposing the results of a color segmentation for the colored parts of the table on an image from the simulator where the robot configuration is set to track the real robot. As a safety precaution, we stop the robot if the payload is exceeded. We note that the robot does not use any direct force sensing during the experiments.

We initialize the robot to the same location and run ten trials each with a different part of the whiteboard painted. One example of the run can be seen in Fig. 6 and more runs in the video attachment. We assume a successful trial when the robot wipes more than 3/4 of the painted line. The robot wipes successfully the board in 8 of the 10 trials. In one of the failed trials the robot moved abruptly and triggered the safety mechanism. In another trial the robot did not wipe the mark entirely. These results indicate that the policies trained with VICES can transfer seamlessly to real world by exploiting the knowledge of the dynamics model of the robot.

VI. CONCLUSION

Reinforcement Learning (RL) as a family of algorithms has ushered in impressive results in generalization, yet principled evaluation on how to choose action spaces to learn control

policies is missing. We presented a thorough evaluation of the effect of the choice of action space on learning policies in RL for tasks without contact, with kinematic constraints and contact-rich manipulation tasks. We also presented variable impedance in end-effector space (VICES) as an efficient choice of action space for RL and showed empirically that, even when contact conditions are dynamically variable during the task, this model outperforms other action space choices on sample efficiency, energy consumption, and safety. We also showed that, thanks to the subtraction of the dynamic effects of the embodiment, using variable impedance in end-effector space we can transfer policies learned in simulation to other simulated robots and to a real robot without fine tuning.

REFERENCES

[1] F. J. Abu-Dakka, L. Rozo, and D. G. Caldwell, "Force-based variable impedance learning for robotic manipulation", *Robotics and Autonomous Systems*, vol. 109, pp. 156–167, Nov. 2018.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration", *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, May 2009.

[3] H. Bruyninckx and J. De Schutter, "Specification of force-controlled actions in the "task frame formalism"-a synthesis", *Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 581–589, Aug. 1996.

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

[5] S. Calinon and A. Billard, "A probabilistic programming by demonstration framework handling skill constraints in joint space and task space", in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Nice, France, Sep. 2008, pp. 367–372.

[6] S. Calinon, I. Sardellitti, and D. G. Caldwell, "Learning-based control strategy for safe human-robot interaction exploiting task and robot redundancies", in *IROS*, Oct. 2010, pp. 249–254.

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

[8] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications", *arXiv preprint arXiv:1812.05905*, 2018.

[9] J. Harrison*, A. Garg*, B. Ivanovic, Y. Zhu, S. Savarese, L. Fei-Fei, and M. Pavone (* equal contribution), "Adapt: Zero-shot adaptive policy transfer for stochastic dynamical systems", in *ISRR*, SPRINGER STAR, 2017.

[10] N. Hogan, "Impedance control: An approach to manipulation", *Journal of dynamic systems, measurement, and control*, vol. 107, p. 17, 1985.

[11] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Movement imitation with nonlinear dynamical systems in humanoid robots", in *ICRA*, vol. 2, May 2002, 1398–1403 vol.2.

[12] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors", *Neural Comput.*, vol. 25, no. 2, 2013.

[13] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, "Learning force control policies for compliant manipulation", in *IROS*, 2011.

[14] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. Garcia Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg, "Real-time perception meets reactive motion generation", *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1864–1871, Jul. 2018.

[15] M. Khansari, E. Klingbeil, and O. Khatib, "Adaptive human-inspired compliant contact primitives to perform surface–surface contact under uncertainty", *IJRR*, vol. 35, no. 13, pp. 1651–1675, 2016.

[16] O. Khatib, "Inertial Properties in Robotic Manipulation: An Object-Level Framework", *IJRR*, vol. 14, no. 1, pp. 19–36, 1995.

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

[18] B. Kim, J. Park, S. Park, and S. Kang, "Impedance learning for robotic contact tasks using natural actor-critic algorithm", *Transactions on Systems, Man, and Cybernetics*, vol. 40, no. 2, pp. 433–443, 2010.

[19] O. Kroemer, C. Daniel, G. Neumann, H. Van Hoof, and J. Peters, "Towards learning hierarchical skills for multi-phase manipulation tasks", in *ICRA*, IEEE, 2015.

[20] T. Kröger, B. Finkemeyer, U. Thomas, and F. M. Wahl, "Compliant motion programming: The task frame formalism revisited", *Mechatronics & Robotics, Aachen, Germany*, 2004.

[21] K. Kronander and A. Billard, "Learning compliant manipulation through kinesthetic and tactile human-robot interaction", *Transactions on Haptics*, vol. 7, no. 3, pp. 367–380, Jul. 2014.

[22] A. X. Lee, H. Lu, A. Gupta, S. Levine, and P. Abbeel, "Learning force-based manipulation of deformable objects from multiple demonstrations", in *ICRA*, IEEE, 2015, pp. 177–184.

[23] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks", in *ICRA*, 2019.

[24] D. Leidner, W. Bejjani, A. Albu-Schäffer, and M. Beetz, "Robotic agents representing, reasoning, and executing wiping tasks for daily household chores", in *AAMAS*, 2016.

[25] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies", *JMLR*, vol. 17, no. 1, 2016.

[26] M. Li, H. Yin, K. Tahara, and A. Billard, "Learning object-level impedance control for robust grasping and dexterous manipulation", in *ICRA*, May 2014, pp. 6784–6791.

[27] Y. Li, G. Ganesh, N. Jarrasse, S. Haddadin, A. Albu-Schaeffer, and E. Burdet, "Force, Impedance, and Trajectory Learning for Contact Tooling and Haptic Identification", *Transactions on Robotics*, vol. 34, no. 5, pp. 1170–1182, 2018.

[28] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning", *arXiv preprint arXiv:1509.02971*, 2015.

[29] M. T. Mason, "Compliance and force control for computer controlled manipulators", *Transactions on Systems, Man, and Cybernetics*, vol. 11, no. 6, pp. 418–432, Jun. 1981.

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

[31] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning", in *ICML*, 2016, pp. 1928–1937.

[32] C. Ott, *Cartesian impedance control of redundant and flexible-joint robots*. Springer, 2008.

[33] X. B. Peng and M. van de Panne, "Learning locomotion skills using deeprl: Does the choice of action space matter?", in *SIGGRAPH*, USA: ACM, 2017.

[34] J. Rey, K. Kronander, F. Farshidian, J. Buchli, and ·. A. Billard, "Learning motions from demonstrations and rewards with time-invariant dynamical systems based policies", vol. 42, 2018.

[35] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control", *IJRR*, vol. 32, no. 3, pp. 280–298, 2013.

[36] L. Rozo, S. Calinon, D. G. Caldwell, P. Jimenez, and C. Torras, "Learning physical collaborative robot behaviors from human demonstrations", *Transactions on Robotics*, vol. 32, no. 3, 2016.

[37] E. A. Rückert, G. Neumann, M. Toussaint, and W. Maass, "Learned graphical models for probabilistic planning provide a new class of movement primitives", *Frontiers in computational neuroscience*, vol. 6, p. 97, 2013.

[38] S. Schaal, A. Ijspeert, and A. Billard, "Computational approaches to motor learning by imitation", *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 358, no. 1431, pp. 537–547, 2003.

[39] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization.", in *Icml*, vol. 37, 2015, pp. 1889–1897.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv*, 2017.

[41] S. Sen*, A. Garg*, D. V. Gealy, S. McKinley, Y. Jen, and K. Goldberg (* equal contribution), "Automating multiple-throw multilateral surgical suturing with a mechanical needle guide and sequential convex optimization", in *ICRA*, 2016.

[42] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[43] B. Thananjeyan, A. Garg, S. Krishnan, C. Chen, L. Miller, and K. Goldberg, "Multilateral surgical pattern cutting in 2d orthotropic gauze with deep reinforcement learning policies for tensioning", in *ICRA*, IEEE, 2017, pp. 2371–2378.

[44] E. Theodorou, J. Buchli, and S. Schaal, "A generalized path integral control approach to reinforcement learning", *JMLR*, vol. 11, no. Nov, pp. 3137–3181, 2010.

[45] M. Toussaint, K. R. Allen, K. A. Smith, and J. B. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning", in *RSS*, 2018.

[46] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards", *arXiv preprint arXiv:1707.08817*, 2017.

[47] J. Viereck, J. Kozolinsky, A. Herzog, and L. Righetti, "Learning a Structured Neural Network Policy for a Hopping Task", *RAL*, 2018.

[48] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills", in *RSS*, 2018.