Adapting Object-Centric Probabilistic Movement Primitives with Residual Reinforcement Learning

João Carvalho[1], Dorothea Koert[1,4], Marek Daniv[1], Jan Peters[1,2,3,4]

***Abstract*— *Abstract*— It is desirable for future robots to quickly learn new tasks and adapt learned skills to constantly changing environments. To this end, Probabilistic Movement Primitives (ProMPs) have shown to be a promising framework to learn generalizable trajectory generators from distributions over demonstrated trajectories. However, in practical applications that require high precision in the manipulation of objects, the accuracy of ProMPs is often insufficient, in particular when they are learned in cartesian space from external observations and executed with limited controller gains. Therefore, we propose to combine ProMPs with the Residual Reinforcement Learning (RRL) framework, to account for both, corrections in position and orientation during task execution. In particular, we learn a residual on top of a nominal ProMP trajectory with Soft Actor-Critic and incorporate the variability in the demonstrations as a decision variable to reduce the search space for RRL. As a proof of concept, we evaluate our proposed method on a 3D block insertion task with a 7-DoF Franka Emika Panda robot. Experimental results show that the robot successfully learns to complete the insertion, which was not possible before with using basic ProMPs.**

I. Introduction

The ability to learn new tasks from non-robotic experts would be of great benefit for future robots in industry, as well as everyday applications. To this end, Learning from Demonstration (LfD) [1], and particularly movement primitives, offers a way to learn generalizable trajectory generators, from a few demonstrated trajectories. However, in precise or contact-rich manipulation tasks, often pure imitation of demonstrated behaviors does not work well [2]. Additionally, in real robotic applications controller accuracy might differ between robots or tasks, and lower controller gains, desirable e.g. to ensure safety in close cooperation with humans, could lead to tracking inaccuracies of trajectories learned from demonstrations.

The recently introduced Residual Reinforcement Learning (RRL) [3], [4], [5] approach has shown success in combining classical controllers with residual policies learned through direct interactions with simulated or real environments. In this paper, we explore combining RRL with movement primitives to use advantages of both in a symbiotic way. This has already been investigated for the cases of Dynamic Movement Primitives (DMPs) [6] and Gaussian Mixture Model (GMM) based Primitives [7]. Inspired by these works,

This work was funded by the German Federal Ministry of Education and Research (project IKIDA, 01IS20045).

[1] TU Darmstadt (TUDa), Institute for Intelligent Autonomous Systems, Computer Science Department [2] German Research Center for AI (DFKI), Research Department: Systems AI for Robot Learning [3] Hessian.AI [4] TUDa Centre for Cognitive Science – Email: {joao,doro,jan}@robot-learning.de, {marek.daniv}@stud.tu-darmstadt.de

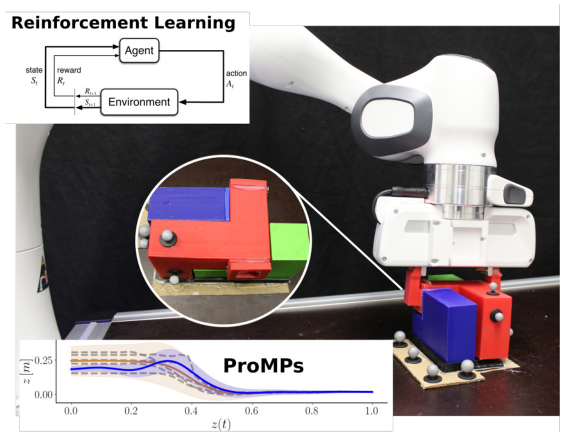

Fig. 1: The robot learns to insert the red block from the Ubongo3D game with ProMPs adapted with residual learning in cartesian space. The tolerance for insertion is $\sim$ 3mm.

we are investigating here the benefit of RRL for another movement primitive representation, namely Probabilistic Movement Primitives (ProMPs) [8], [9]. ProMPs encode demonstrated trajectories as distributions over weighted basis functions and allow to capture variability and correlations within demonstrated trajectories. Probabilistic operators such as conditioning on different goals and start points can then be used to generalize from demonstrations to unseen situations. While this works well for tasks demonstrated with joint-space kinesthetic teaching, e.g. assisting in coffee making [10], these conditioning operations often are too imprecise for insertion tasks, in particular when executed on a real robot with limited controller gains.

To overcome these limitations, we propose to combine ProMPs with RRL, such that during task execution a robot can refine and iteratively improve trajectories. In particular, we investigate in an experimental study whether ProMPs, which were learned from external observations, can be used in combination with RRL to solve a precision 3D block insertion task. The main contributions of this paper are hereby the following:

- On top of a nominal trajectory generated with a ProMP we propose to learn a residual to account for both corrections in position and orientation with Soft Actor-Critic (SAC) [11] in a real robotic system.
- We use the variability in the demonstrations as a decision variable to reduce the search space for RRL and compare this approach to a distance-based strategy for weighting nominal and residual policy.
- We evaluate the proposed method on a 3D block in-

sertion task on a 7-DoF Franka Emika Panda robotic arm. Contrary to a standard peg-in-hole, this task is not invariant to rotations around all axes.

The rest of this paper is structured as follows. In Section II we discuss related work for RRL with classical controllers and LfD. In Section III we introduce our approach to combine RRL with an object-centric formulation of ProMPs. Here we also discuss three different options for the combination of the residual and nominal policies, which we then evaluate on a 3D block insertion task with a real robot in Section IV. In Section V we draw conclusions, discuss current limitations of the proposed approach and give an outlook on future work.

II. Related Work

Residual reinforcement learning [3], [4], [5] has been proposed as a way to solve challenging robotic manipulation tasks by adapting control actions from a conventional (model-based) controller with a policy learned with model-free RL, which significantly reduces the search space and thus improves sample efficiency. [3] introduced RRL as learning a residual on top of an initial controller to improve non-differentiable policies using model-free deep RL. [4] uses a residual policy and a hand-engineered base controller in a robotic task to insert a block between two others that can topple, using a dense reward and a fine-tuned vision setup. [12] extends RRL to learn from sparse rewards and visual input demonstrations as image sequences in simulation. Through behavioral cloning, they first learn task-relevant visual features. Afterwards, the RL policy is optimized using the pretrained state features. [5] applies deep RRL to industrial tasks in the real world, uses feature state from images, a sparse reward, and a hand-designed P-controller as a nominal policy. The cartesian actions are transformed to joint-space via inverse kinematics. [13] learns assembly tasks with a real robot in a few minutes, combining cartesian impedance control with a recurrent policy version of TD3 [14] to learn in the presence of position uncertainties. Variable Impedance End-Effector Space (VICES) [15] studied the effects of different action spaces, and argued for impedance control in end-effector space. Besides learning a policy to learn small changes in pose, they also learn state-dependent gains for the impedance controller. Older work [16] learned impedance gains of a robotic arm based on the equilibrium point control theory and the natural actor-critic algorithm [17] for contact tasks. [18] learns force control for rigid position-controlled robots and its follow up work [19] solves peg-in-hole tasks with hole-position uncertainty with an off-policy, model-free RL method using several sim2real techniques, such as domain randomization. [20] imitate human assembly skills through hybrid trajectory and force learning with hierarchical imitation learning. A scheme to learn an optimal force control policy with goal conditioned imitation learning is presented in [21] and closely connect to [22]. Guided Uncertainty-Aware Policy Optimization (GUAPO) [23] quantifies uncertainty in pose estimation to define a binary switching strategy that determines where to use model-based or RL policies. [24] evaluates Qgraph-bounded DDPG for improving model-free RL to solve a peg-in-hole with a force-torque action space. [25] proposes a hybrid RRL to modify the signals used by the RL policy to prevent internal feedback signals of the low-level controller limiting the RL agent to adequately improve its policy and thus harming learning. Their approach is shown in a contact-rich peg-insertion task. [26] combines visual servoing-based LfD and force-based Learning by Exploration (LbE), and proposes region-limited residual RL (RRRL) policy that acts only on a region close to the goal, determined by the euclidean distance. [27] builds upon Deep Deterministic Policy Gradients (DDPG) to incorporate existing base controllers into stages of exploration, value learning, and policy update, and present a straightforward way of synthesizing different base controllers to integrate their strengths. [7] proposes Soft-Actor Critic Gaussian Mixture Model (SAC-GMM), which is a hybrid approach that learns robot skills through a dynamical system modeled in state-space with GMMs and adapts the learned skills through interactions with the environment using RRL. They present results in simulation for peg-insertion and power-lever-sliding skills, and real-world results for a door-opening task, using a camera image as a policy input. [28] propose the InsertionNet, which uses visual and wrench inputs to learn a residual policy in position and orientation. Demonstrations are provided with a carefully designed procedure with backwards learning by first physically moving the robot to the final pose, e.g. the hole in the peg-in-hole task, and then generating collisions in order to collect data. Data augmentation of images is used for robustness. Having this dataset the insertion problem resorts to learning function parameters in a regression task. To reach the insertion area they use a PD controller to follow a pre-computed trajectory.

The closest work to ours is the combination of Dynamic Movement Primitives (DMPs) with residual learning as done in [6], where it was shown that RRL is better than learning the DMP parameters with RL. Additionally, it is stated that learning orientation is important for reliable insertion tasks (several other works learn only the deviation in position). Our work differs from [6] in that we make use of the variance in the demonstrations to check if the current position is inside the confidence interval and decide if another demonstration is needed, and when to adapt the nominal controller, while [6] specifies a time period after which the RL policy contributes to solving the task. In [29] insertion tasks have been learned with DMPs by tuning the parameters with episodic RL using Policy learning by Weighting Exploration with the Returns (PoWER) [30], but the trajectories are demonstrated with kinesthetic teaching, which facilitates learning in joint space. On the contrary, we work in cartesian space, which is inherently more difficult.

III. Residual RL for Object-Centric ProMPs

In this section, we explain the different components of our proposed approach to combine ProMPs and RRL. An overview of the resulting method is shown in Fig. 2. We

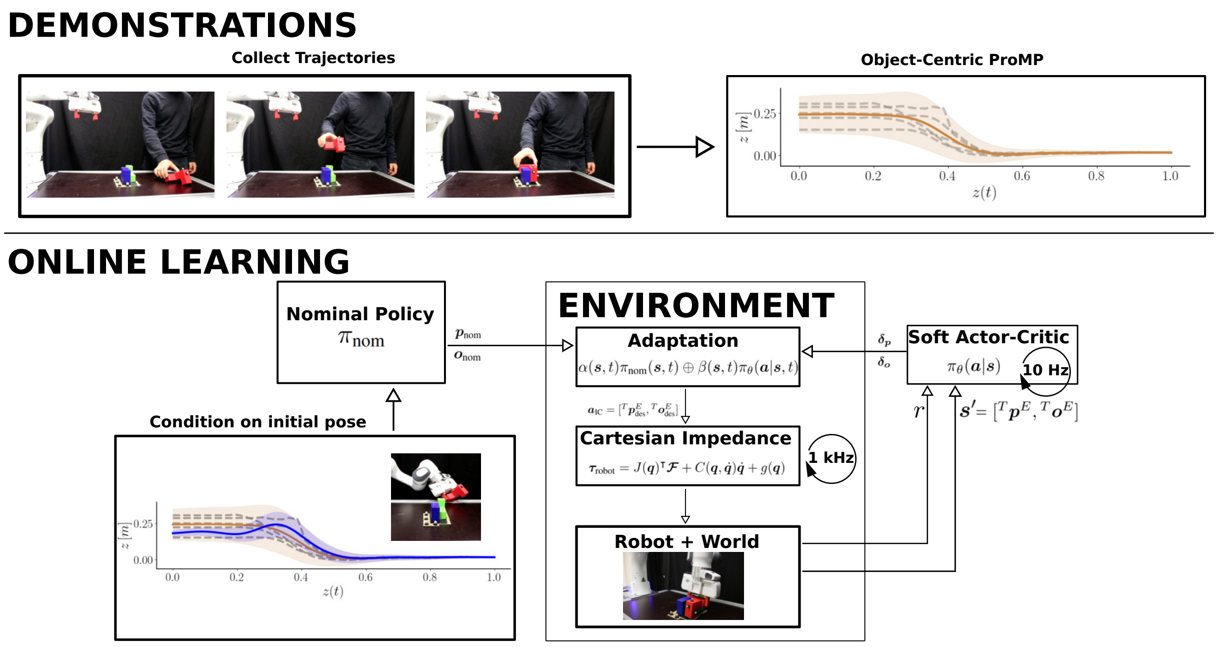
DEMONSTRATIONS
Collect Trajectories
Object-Centric ProMP
$z\,[m]$
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
$z(t)$
ONLINE LEARNING
Nominal Policy
π_{nom}
$\boldsymbol{p}_{\text{nom}}$
$\boldsymbol{o}_{\text{nom}}$
Condition on initial pose
ENVIRONMENT
Adaptation
$\alpha(\boldsymbol{s},t)\pi_{\text{nom}}(\boldsymbol{s},t) \oplus \beta(\boldsymbol{s},t)\pi_\theta(\boldsymbol{a}|\boldsymbol{s},t)$
$\boldsymbol{a}_{\text{IC}} = [{}^T\boldsymbol{p}^E_{\text{des}}, {}^T\boldsymbol{o}^E_{\text{des}}]$
Cartesian Impedance
$\tau_{\text{robot}} = J(\boldsymbol{q})^\intercal \boldsymbol{\mathcal{F}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q})$
1 kHz
Robot + World
Soft Actor-Critic
$\pi_\theta(\boldsymbol{a}|\boldsymbol{s})$
10 Hz
δ_p
δ_o
r
$\boldsymbol{s}' = [{}^T\boldsymbol{p}^E, {}^T\boldsymbol{o}^E]$

Fig. 2: Our approach uses ProMPs to learn an object-centric trajectory generator from few demonstrations. At a new position within the confidence interval of the demonstrated trajectory distribution, the ProMP is conditioned and the mean trajectory is used to guide the robot to the goal area. A residual policy learns deltas in position and orientation with SAC and an adaptation policy combines the actions from the nominal and learned policies. A low-level cartesian controller implements the interface to the robot torques.

discuss the used underlying control structure in Section III-A and provide a short recap on ProMPs in Section III-B and on RL and SAC in Section III-C. Afterwards, we introduce our approach for adapting Cartesian ProMPs with Residual Robot Learning in Section III-D and explain the used action space and policy parametrization in Section III-E.

A. Cartesian Impedance Control

When performing contact-rich tasks, such as an insertion, cartesian impedance control [31] is an appropriate choice, because it allows to specify the desired compliant behavior of the robot in presence of external forces, e.g. collisions. This aspect is particularly relevant not only to prevent damaging the robot but also in human-robot collaboration. Given a desired set-point (with zero velocity) of the end-effector pose $\mathbf{X}^E_{\text{des}} \in \mathbb{R}^7$, with position $\boldsymbol{p}_{\text{des}} \in \mathbb{R}^3$ and orientation represented as quaternion $\boldsymbol{o}_{\text{des}} \in \mathbb{R}^4$, the torques applied at the robot joints are computed as

$$\begin{aligned}\boldsymbol{\tau}_{\text{robot}} &= J(\boldsymbol{q})^\intercal \boldsymbol{\mathcal{F}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + g(\boldsymbol{q})\\ \boldsymbol{\mathcal{F}} &= \operatorname{diag}(\mathbf{K}_p(\boldsymbol{p}_{\text{des}} - \boldsymbol{p}), \mathbf{K}_o(\boldsymbol{o}_{\text{des}} \ominus \boldsymbol{o})) + \mathbf{D}_d J(\boldsymbol{q})\dot{\boldsymbol{q}},\end{aligned}$$

where $\boldsymbol{q}, \dot{\boldsymbol{q}} \in \mathbb{R}^n$ are the current robot joint positions and velocities, $J(\boldsymbol{q}) \in \mathbb{R}^{6\times n}$ is the Jacobian matrix, $C(\boldsymbol{q},\dot{\boldsymbol{q}}) \in \mathbb{R}^{n\times n}$ and $g(\boldsymbol{q}) \in R^n$ the coriolis forces and gravity compensation terms, $\boldsymbol{\mathcal{F}} \in \mathbb{R}^6$ is the simulated external wrench, $\mathbf{K}_p \in \mathbb{R}^{3\times 3}$ and $\mathbf{K}_o \in \mathbb{R}^{3\times 3}$ the position and orientation stiffness matrices, $\mathbf{D}_d$ the damping matrix and $\ominus$ denotes a difference in quaternion space and translated to axis-angle. Lower gains $\mathbf{K}$ and $\mathbf{D}$ allows for safer robot interaction and exploration, but result in larger tracking errors.

B. Probabilistic Movement Primitives

Movement Primitives (MPs) are a convenient way to represent time-based smooth robot and object movements [32]. In particular, probabilistic formulations allow to also capture variance in the demonstrations [33], [34], [8]. Here, we use ProMPs that are able to construct distributions conditioned on arbitrary time-steps and points inside a confidence interval of the demonstrations while relying on a small amount of training data, when compared to other state-based representations [35].

Formally, a ProMP is a compact representation of a trajectory, where a point $\boldsymbol{y}_z \in \mathbb{R}^d$ in the trajectory (e.g. robot joint values) is assumed to be a linear combination of N basis functions $\boldsymbol{y}_z = \boldsymbol{\Phi}_z^\intercal \boldsymbol{\omega}$, with $\boldsymbol{\Phi} \in \mathbb{R}^N$ a basis function matrix, $\boldsymbol{\omega} \in \mathbb{R}^{N\times d}$ the learnable weights and $z(t) \in [0,1]$ a phase-variable. A distribution $p(\boldsymbol{\omega})$ over the weights is learned from multiple demonstrations. Assuming the weights are Gaussian distributed $p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \boldsymbol{\mu}_{\boldsymbol{\omega}}, \boldsymbol{\Sigma}_{\boldsymbol{\omega}})$, the mean and covariance matrix are obtained via maximum likelihood estimation. For more details on the exact training procedure, we refer the reader to [8] and [10]. Let $\overline{\boldsymbol{y}}_z$ be a point to reach at step $z(t)$ with covariance $\overline{\boldsymbol{\Sigma}}_z$. The conditional distribution over weights is computed with Bayes' rule for Gaussian distributions as $p(\boldsymbol{\omega}|\overline{\boldsymbol{y}}_z, \overline{\boldsymbol{\Sigma}}_y) \propto p(\overline{\boldsymbol{y}}_z|\boldsymbol{\omega}, \overline{\boldsymbol{\Sigma}}_y)p(\boldsymbol{\omega}) = \mathcal{N}(\boldsymbol{\omega}; \overline{\boldsymbol{\mu}}_{\boldsymbol{\omega}}, \overline{\boldsymbol{\Sigma}}_{\boldsymbol{\omega}})$, and the resulting trajectory distribution $p(\overline{\boldsymbol{\tau}}) = \mathcal{N}(\overline{\boldsymbol{\tau}}; \boldsymbol{\Phi}^\intercal \overline{\boldsymbol{\mu}}_{\boldsymbol{\omega}}, \boldsymbol{\Phi}^\intercal \overline{\boldsymbol{\Sigma}}_{\boldsymbol{\omega}} \boldsymbol{\Phi})$, with

$$\begin{aligned}\overline{\boldsymbol{\mu}}_{\boldsymbol{\omega}} &= \boldsymbol{\mu}^*_{\boldsymbol{\omega}} + \mathbf{K}\left(\overline{\boldsymbol{y}}_z - \boldsymbol{\phi}_z^\intercal \boldsymbol{\mu}^*_{\boldsymbol{\omega}}\right), \quad \overline{\boldsymbol{\Sigma}}_{\boldsymbol{\omega}} = \boldsymbol{\Sigma}^*_{\boldsymbol{\omega}} - \mathbf{K}\boldsymbol{\phi}_z^\intercal \boldsymbol{\Sigma}^*_{\boldsymbol{\omega}}\\ \mathbf{K} &= \boldsymbol{\Sigma}^*_{\boldsymbol{\omega}} \boldsymbol{\phi}_z \left(\overline{\boldsymbol{\Sigma}}_y + \boldsymbol{\phi}_z^\intercal \boldsymbol{\Sigma}^*_{\boldsymbol{\omega}} \boldsymbol{\phi}_z\right)^{-1}, \quad \boldsymbol{\Phi}_z = \mathbf{I}_d \otimes \boldsymbol{\phi}_z.\end{aligned}$$

C. Reinforcement Learning and Soft Actor-Critic

Let a Markov Decision Process (MDP) be defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma, \mu_0)$, where $\mathcal{S}$ is a continuous state space $\boldsymbol{s} \in \mathcal{S}$, $\mathcal{A}$ is a continuous action space $\boldsymbol{a} \in \mathcal{A}$, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a transition probability function, with $\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$ the density of landing in state $\boldsymbol{s}'$ when taking action $\boldsymbol{a}$ in state $\boldsymbol{s}$, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function, $\gamma \in [0, 1)$ is a discount factor, and $\mu_0 : \mathcal{S} \to \mathbb{R}$ the initial state distribution. A policy $\pi(\boldsymbol{a}|\boldsymbol{s})$ is a (stochastic) mapping from states to actions. The state-action value function – Q-function – is the discounted sum of rewards collected from a given state-action pair following the policy π, $Q^\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_{\pi,\mathcal{P}}\left[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) | \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right]$. In general, the goal of a Reinforcement Learning (RL) agent is to maximize the expected sum of discounted rewards $J(\pi) = \mathbb{E}_{\tau \sim \mu_0, \pi, \mathcal{P}}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. In high-dimensional and continuous action spaces, typically a policy with parameters $\boldsymbol{\theta}$ is updated iteratively with a gradient ascent step on J, using a variation of the policy gradient theorem [36]. The Soft Actor-Critic (SAC) algorithm is a sample-efficient method to compute an off-policy gradient estimate of $\nabla_{\boldsymbol{\theta}} J(\pi_{\boldsymbol{\theta}})$ [37], [11], with several improvements over previous approaches, namely the entropy regularization and the squashed Gaussian policy. The entropy term encourages exploration by preventing the policy from becoming too deterministic during learning. The surrogate objective optimized by SAC is $J(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{s} \sim d^\beta, \boldsymbol{a} \sim \pi_\theta(\cdot|\boldsymbol{s})}\left[Q^\pi_{\boldsymbol{\phi}}(\boldsymbol{s}, \boldsymbol{a}) - \alpha \log \pi_\theta(\boldsymbol{a}|\boldsymbol{s})\right]$, where d^β is an off-policy state distribution, the Q-function is a neural network parameterized by $\boldsymbol{\phi}$ and α weighs the entropy regularization term. The (unbiased) policy gradient of J is computed by sampling from a replay buffer containing off-policy samples and using the reparametrization trick [38] to differentiate the expectation over actions.

D. Adapting Cartesian ProMPs with Residual Robot Learning

Residual learning is commonly formulated as a combination of policies, which can be both time and state dependent as

$$\begin{aligned}\pi(\boldsymbol{a}|\boldsymbol{s}, t) &= \Psi\left(\pi_{\text{nom}}, \pi_\theta, \boldsymbol{s}, \boldsymbol{a}, t\right) \\ &= \alpha(\boldsymbol{s}, t)\pi_{\text{nom}}(\boldsymbol{s}, t) \oplus \beta(\boldsymbol{s}, t)\pi_\theta(\boldsymbol{a}|\boldsymbol{s}, t),\end{aligned}$$

where π_{nom} is a nominal (or model-based) policy, π_θ is a learnable policy, and α and β we call adaptation parameters. The operation $\oplus$ is dependent on the action space. If $\boldsymbol{a}$ represents a translation, then it can be an addition, but if it is an orientation it can be a quaternion multiplication. In [3], the authors assume $\pi(\boldsymbol{s}) = \pi_{\text{nom}}(\boldsymbol{s}) + \pi_\theta(\boldsymbol{s})$, and hence $\nabla_{\boldsymbol{\theta}} \pi(\boldsymbol{s}) = \nabla_{\boldsymbol{\theta}} \pi_\theta(\boldsymbol{s})$, meaning one can use the policy gradient to optimize $\boldsymbol{\theta}$ without knowing the gradient of π_{nom}. However, this is equivalent to writing the transition function of the residual MDP with the policy transformation $\mathcal{T}^{\pi_{\boldsymbol{\theta}}}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') = \mathcal{T}(\boldsymbol{s}, \Psi\left(\pi_{\text{nom}}, \pi_\theta, \boldsymbol{s}, \boldsymbol{a}, t\right), \boldsymbol{s}')$. Because the transformation is now part of the environment, the agent is unaware of it. Note that in the original formulation of [3] the policy is only state-dependent, and not time-dependent. We augment this definition by using a time dependency, to include policies that result from time-dependent movement primitives such as ProMPs. This does not break the MDP assumption, since t can be seen as part of the state (in an episodic task).

In our approach, we learn ProMPs from external observation of demonstrated object trajectories in cartesian space. For experiments in this paper, we used a motion capturing system and markers on the objects to obtain the demonstrated trajectories. An example of such a demonstration can be seen in Fig. 2 on the top left. We assume that for an insertion task we collect a set of N trajectories $\{\boldsymbol{\tau}_i\}_{i=1,\ldots,N}$ of the pose of an *interest* object I (red object in Fig. 1) in the reference frame of a *target* object T (blue object), where $\boldsymbol{\tau}_i = \left[{}^T\mathbf{X}^I_0, \ldots, {}^T\mathbf{X}^I_{H_i-1}\right]$, with H_i the i-th trajectory length, and ${}^T\mathbf{X}^I = (\boldsymbol{p}, \boldsymbol{o})$, with $\boldsymbol{p} \in \mathbb{R}^3$ is the position and $\boldsymbol{o} \in \mathbb{R}^4$ the orientation (as quaternion). Representing the trajectory in the target frame allows to get the intention of the demonstration, i.e. classify if it is an insertion task, and additionally, if the target or interest objects move to a different pose, their relation is maintained. For a compact representation of the trajectory, we encode the position with an object-centric ProMP in cartesian space. The orientation representation is the average over trajectories. Learning ProMPs for orientation spaces in an ongoing topic of research [39], and we leave this for future work.

A key advantage of movement primitives is their ability to generalize from demonstrated trajectories to new situations. In particular, here ProMPs can be used to compute nominal trajectories for varying start positions of the object. In particular, they can also distinguish between starting points covered by the provided demonstrations and starting points outside this region. We focus here on generalization to starting points within the demonstrated distribution over trajectories only. However, as a future direction it would be also possible to include active requests for additional demonstrations or multi-modal ProMPs using incremental Gaussian Mixture Models [40].

For our approach we compute the nominal trajectory by conditioning the ProMP on the initial time-step $z = 0$ and initial pose $\overline{\boldsymbol{y}}_0 \equiv {}^T\mathbf{X}^I$ with a small covariance $\overline{\boldsymbol{\Sigma}}_0$ and compute the resulting mean trajectory $\overline{\boldsymbol{\tau}} = \boldsymbol{\Phi}^\intercal \overline{\boldsymbol{\mu}}_{\boldsymbol{\omega}}$. Afterwards, the desired trajectory $\overline{\boldsymbol{\tau}}$ is translated to the end-effector in the target frame with ${}^T\mathbf{X}^E = {}^T\mathbf{X}^{II}\mathbf{X}^E$, where a transformation from the end-effector to the interest object is given via a grasping pose.

To follow $\overline{\boldsymbol{\tau}}$ we could compute the inverse kinematics and track it with an inverse dynamics controller in joint space. However, this strategy would require a large gain to ensure a low tracking error, necessary for an insertion task with limited tolerance. These large gains could damage the robot and the environment in case of interaction. For this reason, a low gain controller in cartesian space is better suited for this task. However, due to the low gains, velocity constraints, among others, the controller will not perfectly follow the desired trajectory and thus cannot complete the task, as depicted in the experiment of Fig. 4.

An important decision is where/when to activate the residual part of the policy. In the original formulation of residual RL $\alpha = \beta = 1$, which for longer trajectories can lead to exploring in free-space regions far away from the insertion goal. To prevent this unnecessary exploration, [23] and [26] set $\alpha(\boldsymbol{s}, t) = 0$, $\beta(\boldsymbol{s}, t) = 1$ if $\boldsymbol{s} \in \mathcal{S}_u$, a region in the vicinity of the goal, and $\alpha = 1$, $\beta = 0$ otherwise. [23] defines $\mathcal{S}_u$ based on an uncertainty quantification in pose estimation, and [26] defines $\mathcal{S}_u$ as the distance to the goal, both being hyperparameters. [13] uses a time-based weighting as $\alpha(t) = 1 - \beta(t)$, with $\beta(t) = \max(0, (T - t)/T)$, with T the settling time of the nominal controller, which in practice means that the learned policy only acts if the nominal controller fails, and the residual part of the policy acts alone in the environment. In [6] $\alpha = \beta = 1$ after executing the nominal controller for a certain amount of time (in their task 3.9 sec), and otherwise $(\alpha, \beta) = (1, 0)$.

While conducting experiments, we noticed that, particularly for insertion tasks, it is more beneficial for the agent to explore closer to the insertion point, which has lower entropy (less variance), as can be seen from the last time steps of the learned ProMPs in Fig. 3. With this in mind, we make use of the covariance over originally demonstrated trajectories computed with $\boldsymbol{\Phi}^\intercal \boldsymbol{\Sigma}_{\boldsymbol{\omega}}$ as a guiding signal for exploration. We propose a simple ***variance-based adaptation*** scheme with $\alpha = 1$ and

$$\beta(\boldsymbol{s}, t) = \begin{cases} 1 & \text{if } \exists i : \min_i \sigma_i(t) \leq \varepsilon \\ 0 & \text{otherwise} \end{cases}, \qquad (1)$$

where $\sigma_i(t)$ is the standard deviation of the i-th dimension of the original ProMP at time step t, and ε an hyperparameter.

E. Action Space and Policy Parametrization

For the RL agent, the cartesian impedance controller is part of the environment and takes as input a desired pose $\boldsymbol{a}_{\text{IC}} = [{}^T\boldsymbol{p}^E_{\text{des}}, {}^T\boldsymbol{o}^E_{\text{des}}]$, computed with the adaptation scheme. The policy π_θ is learned with SAC, and encodes the mean and variance of a (factored) Gaussian distribution over positions and orientations $\pi_\theta(\boldsymbol{a}|\boldsymbol{s}) = \mathcal{N}(\boldsymbol{\delta_p}; \mu^{\boldsymbol{p}}_{\boldsymbol{\theta}}(\boldsymbol{s}), \Sigma^{\boldsymbol{p}}_{\boldsymbol{\theta}}(\boldsymbol{s}))\, \mathcal{N}(\boldsymbol{\delta_\nu}; \mu^{\boldsymbol{\nu}}_{\boldsymbol{\theta}}(\boldsymbol{s}), \Sigma^{\boldsymbol{\nu}}_{\boldsymbol{\theta}}(\boldsymbol{s}))$. The functions encoding the mean and covariance are neural networks that share the same features up to the last linear layer. The delta in orientations $\boldsymbol{\delta_\nu} \in \mathbb{R}^3$ parametrizes the coordinates of an axis-angle representation. Positions and orientations are restricted to the cartesian controller limits with a hyperbolic tangent. The nominal and learned policies are combined as follows. For the position it is a simple addition ${}^T\boldsymbol{p}^E_{\text{des}} = \boldsymbol{p}_{\text{nom}} + \boldsymbol{\delta_p}$. For orientations, we compute the quaternion representation of $\boldsymbol{\delta_\nu}$ as $\boldsymbol{\delta_o} = [\cos(\|\boldsymbol{\delta_\nu}\|/2), \boldsymbol{\delta_\nu}/\|\boldsymbol{\delta_\nu}\| \sin(\|\boldsymbol{\delta_\nu}\|/2)]$, and afterwards apply a quaternion multiplication to obtain the desired rotation ${}^T\boldsymbol{o}^E_{\text{des}} = \boldsymbol{\delta_o} \circ \boldsymbol{o}_{\text{nom}}$.

IV. Experimental Results

As a proof of concept, we evaluate our proposed method in a $3D$ block insertion task with a 7-DoF Franka Panda Robot. In these experiments, we investigate if the task, which we could not solve with basic ProMPs before, benefits from the combination of ProMPs and RRL and compare different ways of combining nominal and residual policies to assess which one works best in the real system.

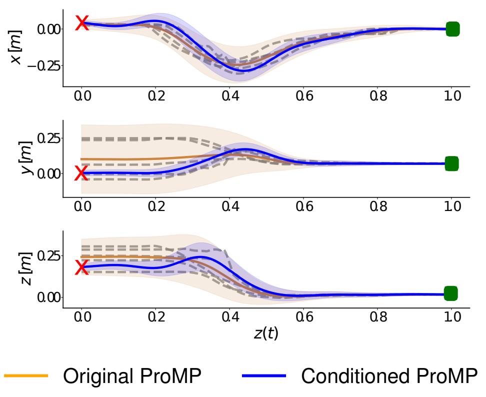

Fig. 3: Human demonstrated trajectories (gray lines) and ProMPs of interest object position in the target object frame. The overall ProMP learned from demonstrations is shown in orange and the result after conditioning is shown in blue. Solid lines represent the mean and the shaded area two times standard deviations of the ProMPs. The red cross indicates the conditioned observation at time $z = 0$, and the green square the trajectory goal at $z = 1$.

A. Experiment Setup

We evaluate the proposed method on an approximation of the Ubongo3D game [41], which consists of different shapes that have to be assembled together in a limited space. For the experiments in this paper, we use 3 different shapes - the red, green and blue elements in Fig. 1 - and a base plate (in black). The tolerance for insertion is approximately 3mm. Each shape is a custom 3D printed structure, consisting of cubes with 3.5cm sides. The goal is to build a *a priori* unknown structure with a height of two cubes and such that the base plate is covered. The game has a planning and a manipulation part. Solving the planning problem involves deciding the pose of each shape and could e.g. be done using Mixed-Integer Programming [42]. On the other hand, stacking/inserting the shapes together can be seen as a fine manipulation task, which is knowingly difficult for robots [2].

We see the Ubongo 3D task as a proxy for more complex assembly scenarios with small tolerances. Due to the involvement of multiple points of contact for the insertion, we also consider it a particularly suitable task for model-free RL approaches, since they do not rely on accurate models which would be hard to obtain in practice. Additionally, the insertion is not invariant to changes in orientation and therefore requires learning orientations as part of the residual policy.

For the experiments in this paper, we assume the blue and green shapes are already placed and learn how to insert the red shape from human demonstrated trajectories. Therefore, demonstrations are recorded starting from different initial

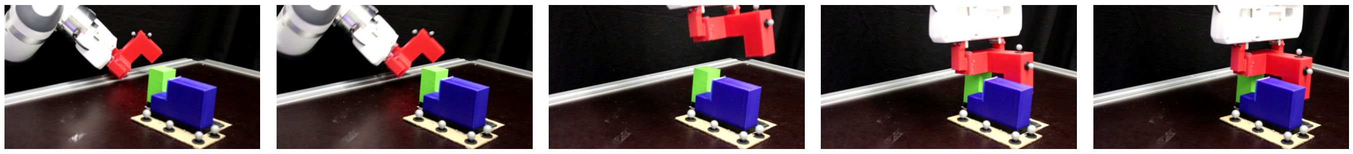
Fig. 4: Snapshot an episode using the nominal controller only. This policy fails to precisely track the trajectory, due to the low gains used to ensure safe interaction.

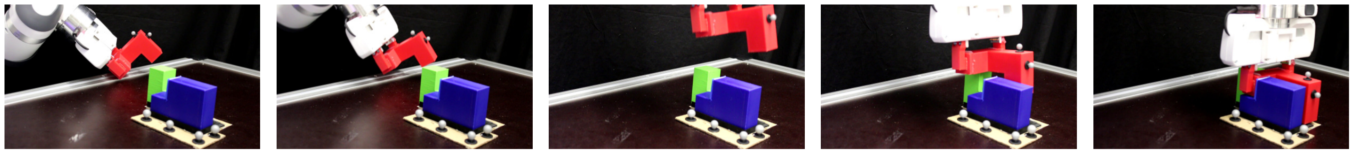
Fig. 5: Snapshot of an episode using the learned policy of Residual RL with variance-based adaptation. By using the residual in low entropy regions of the state-space, such as ones close to the insertion point, the RL agent only needs to learn to adapt closer to the goal. Notice how the 4th image closely matches the one from Figure 4, showing that the agent follows the nominal policy and only adapts near the goal.

configurations, using a motion capturing system and markers attached to the objects. We record the positions and orientations of the interest object pose (the red shape) in the reference frame of a target object (the blue shape). It is notable here that these demonstrations are recorded from external observations in Cartesian space, i.e. a human demonstrator moving the objects and not with kinesthetic teaching on the robot. We believe this is an important aspect when learning from humans demonstrators, especially non technical ones, since operating a robot with kinesthetic teaching can easily lead to kinematic singularities. Fig. 3 shows the 5 recorded trajectories, the resulting learned ProMP (orange), and an example for conditioning on a new (not initially demonstrated) initial position (blue). For learning the ProMP we decided to use 10 basis function after comparing the data log-likelihood in a grid search.

The cartesian impedance controller runs at 1kHz and the learnable policy at 10Hz. The task is executed episodically with 100 steps, amounting to approximately 15 seconds per episode, and a learning trial took approximately 15 minutes. The episode terminates if the interest object (red shape) is at a distance of the goal position and orientation less than 5mm and $5°$, respectively. The state is the position and orientation of the end-effector in the target object frame $\boldsymbol{s} = [{}^{T}\boldsymbol{p}^{E}, {}^{T}\boldsymbol{o}^{E}] \in \mathbb{R}^{7}$. While recent works [13] also use the external wrench expressed in the target frame as part of the state feedback, we found in our experiments that the measurements provided by the Panda robot were too unreliable and not useful for learning. Additionally, force torque sensors are still costly and thus developing methods that work without them are also desirable.

We experimented using a sparse reward, but found that the task was too difficult to learn without a reward signal. We hypothesize that this was due to the termination condition being too strict. Quite often the red shape was already fairly in place, but not enough to get to terminate and obtain a reward. For a peg-in-hole task, many works use a sparse reward, because once the peg is inside the hole the region of exploration is lower, especially with a fixed orientation, and the agent simply has to push down the peg. In our case, we have an insertion that includes both precise position and orientation, and since an approximate final pose is known, we decided to use instead a per-step dense reward function that weighs the absolute distances of the error in position and orientation $r(\boldsymbol{s}, \boldsymbol{a}) = -\sum_{i=1}^{3} |{}^{T}\boldsymbol{p}^{E}_{\text{des}} - {}^{T}\boldsymbol{p}^{E}_{i}| - 50\|({}^{T}\boldsymbol{o}^{E}_{\text{des}} \ominus {}^{T}\boldsymbol{o}^{E})\|$. Using dense rewards is also common in other works that learn in the real system [4]. The goal pose $[{}^{T}\boldsymbol{p}^{E}_{\text{des}}, {}^{T}\boldsymbol{o}^{E}_{\text{des}}]$ is known to the agent, because it is given by the last point in the trajectory extracted from the demonstrations encoded in the ProMP.

The RL module includes a policy and Q-functions as two-layer neural networks with 128 hidden neurons and ReLU activations, and the entropy regularization weight in SAC (α) is also learned. All parameters are optimized with ADAM [43] and a learning rate $3 \cdot 10^{-4}$. The initial replay size and the number of samples before policy updates is 100, as well as the batch size for actor and critic learning. In practice, we did not find the need to use a recurrent policy as in [13], [6]. The discount factor was $\gamma = 0.99$. The RL agent was implemented using Mushroom-RL [44], and an interface to real robot uses ROS [45] for data processing and computing the low-level cartesian impedance controller.

B. Results

We report the results of executing the nominal controller and three adaptation strategies:

- *Residual RL* - the residual policy is always active
- *Residual RL with variance-based adaptation* (ours) - detailed in Eq. 1
- *Residual RL with distance-based adaptation* - if the distance to the goal is less than a threshold, use only the residual policy as in [26]. The threshold was chosen empirically such that it occurs almost before insertion as in the 4-th picture in Fig. 4.

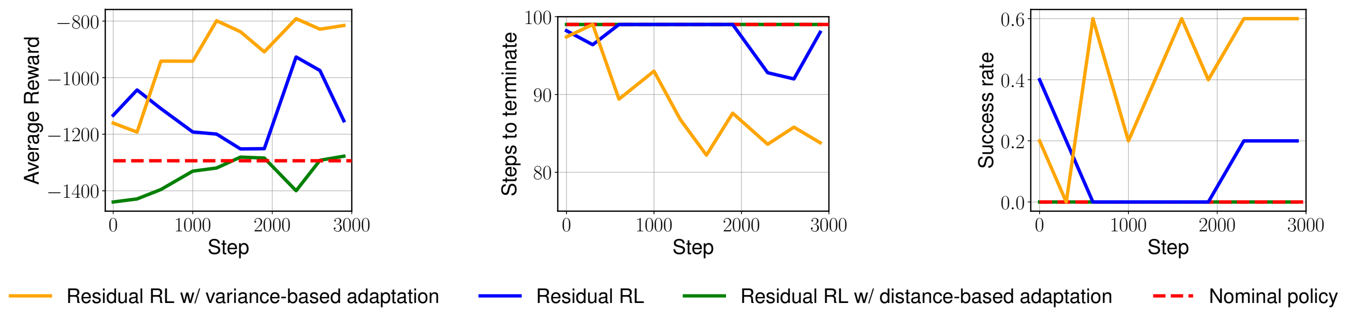

Fig. 6: Results of learning in the real robot. In the x-axis, *Step* denotes the number of commands computed by the high-level policy, as a delta in position and orientation, and executed by the real robot using with the lower-level cartesian impedance controller. An episode of 100 steps amounts to approximately 15 seconds of real world interaction, which in total corresponded to 8 minutes of training. The left plot shows the average reward obtain during learning. The center plot shows the average number of steps to terminate the task, where termination occurs when the red shape is correctly inserted. The right plot shows the average success rate of insertion. The plots show a qualitative trend that residual model-free RL is able to adapt ProMPs for the Ubongo3D task. The solid lines depict the mean over 5 trials with different random seeds in the real system.

Additionally, we include the results of the *Nominal Policy*, which is given by following the ProMP mean trajectory using the cartesian impedance controller, i.e. no learning or residual adaptation is involved.

Figures 4 and 5 show snapshots of the execution of the nominal controller and the learned residual policy with the variance-based adaptation strategy, respectively. Notice that the initial position starts far away from the goal, which makes it more difficult to precisely track the object trajectory until the insertion point.

The average reward, number of time steps to succeed, and the success rate during training of the three adaptation strategies are depicted in Fig. 6. The results show that while the nominal controller (red dashed line) cannot solve the task due to the low-gain controller, our method (yellow line), and residual RL (blue line) can improve it, as seen in the average reward plot. Using the residual RL policy for the whole trajectory can lead to explore far from the insertion point and small deviations can accumulate over time, leading to a point where the red shape gets stuck and cannot finish the insertion. In the rightmost plot, the success rate at step 0 is different from 0, because some trials could already perform an insertion just by adding small Gaussian noise around the nominal trajectory. Even though we initialize the policy neural network to output means close to 0, as is common in RRL, by construction the network computing the variance outputs a value different from 0, making it sufficient to move the shape slightly around the nominal trajectory. For the distance-based adaptation (green lines) we made sure to only switch fully to the learnable controller when the red shape is already in contact with the green shape (4th frame in Fig. 4). While this strategy is slowly learning - note the increase in reward the leftmost plot of Fig. 6 - meaning the policy is successfully bringing the red shape to the goal location, it could not complete the task in any of the trials, as was expected due to the random exploration. Lastly, the steps it takes for the episode to terminate are correlated with the success rate.

V. Conclusion and Future Work

In this paper, we studied how to use human demonstrations of object-centric trajectories in combination with ProMPs and residual reinforcement learning. Because the cartesian impedance controller has a low gain to guarantee safe interaction, simply following the trajectory from a conditioned ProMP resulted in task failure. We overcome this problem by proposing to learn a residual policy in position and orientation with model-free reinforcement learning. Making use of the variability in the demonstrations, we present an adaptation strategy based on the variance of the ProMP as an indication of a region where an insertion task can take place, and thus where the policy needs to be learned, thereby increasing the sample efficiency. The experimental evaluations in the real robot showed that our method is able to learn a policy that corrects the ProMP trajectory to perform a block insertion task in the Ubongo3D game.

In future work we plan to evaluate our approach in tasks with different objects, such as a peg-in-hole task where rotation is needed, include a formulation of orientation ProMPs in Riemannian Manifolds [46], and study trajectory representation methods using state-space information [35].

References

[1] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Survey: Robot programming by demonstration,” Springrer, Tech. Rep., 2008.

[2] O. Kroemer, S. Niekum, and G. D. Konidaris, “A review of robot learning for manipulation: Challenges, representations, and algorithms,” *Journal of machine learning research*, vol. 22, no. 30, 2021.

[3] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, “Residual policy learning,” *arXiv preprint arXiv:1812.06298*, 2018.

[4] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, “Residual reinforcement learning for robot control,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6023–6029.

[5] G. Schoettler, A. Nair, J. Luo, S. Bahl, J. A. Ojea, E. Solowjow, and S. Levine, “Deep reinforcement learning for industrial insertion tasks with visual inputs and natural rewards,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5548–5555.

[6] T. B. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy, "Residual learning from demonstration: Adapting dmps for contact-rich manipulation," *IEEE Robotics and Automation Letters*, 2022.

[7] I. Nematollahi, E. Rosete-Beas, A. Röfer, T. Welschehold, A. Valada, and W. Burgard, "Robot skill adaptation via soft actor-critic gaussian mixture models," *arXiv preprint arXiv:2111.13129*, 2021.

[8] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013.

[9] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, "Using probabilistic movement primitives in robotics," *Autonomous Robots*, vol. 42, no. 3, pp. 529–551, 2018.

[10] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, "Adaptation and robust learning of probabilistic movement primitives," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 366–379, Mar. 2020.

[11] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proceedings of International Conference on Machine Learning (ICML)*, vol. 80, 2018, pp. 1856–1865.

[12] M. Alakuijala, G. Dulac-Arnold, J. Mairal, J. Ponce, and C. Schmid, "Residual reinforcement learning from demonstrations," *arXiv preprint arXiv:2106.08050*, 2021.

[13] P. Kulkarni, J. Kober, R. Babuška, and C. Della Santina, "Learning assembly tasks in a few minutes by combining impedance control and residual recurrent reinforcement learning," *Advanced Intelligent Systems*, p. 2100095, 2021.

[14] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80, 2018, pp. 1587–1596.

[15] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1010–1017.

[16] B. Kim, J. Park, S. Park, and S. Kang, "Impedance learning for robotic contact tasks using natural actor-critic algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 40, no. 2, pp. 433–443, 2009.

[17] J. Peters and S. Schaal, "Natural actor-critic," *Neurocomput.*, vol. 71, no. 7–9, p. 1180–1190, Mar. 2008.

[18] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, T. Nishi, S. Kikuchi, T. Matsubara, and K. Harada, "Learning force control for contact-rich manipulation tasks with rigid position-controlled robots," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5709–5716, 2020.

[19] C. C. Beltran-Hernandez, D. Petit, I. G. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Applied Sciences*, vol. 10, no. 19, p. 6923, 2020.

[20] Y. Wang, C. C. Beltran-Hernandez, W. Wan, and K. Harada, "Robotic imitation of human assembly skills using hybrid trajectory and force learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 11 278–11 284.

[21] Y. Ding, C. Florensa, P. Abbeel, and M. Phielipp, "Goal-conditioned imitation learning," *Advances in neural information processing systems*, vol. 32, 2019.

[22] Y. Wang, C. C. Beltran-Hernandez, W. Wan, and K. Harada, "Hybrid trajectory and force learning of complex assembly tasks: A combined learning framework," *IEEE Access*, vol. 9, pp. 60 175–60 186, 2021.

[23] M. A. Lee, C. Florensa, J. Tremblay, N. Ratliff, A. Garg, F. Ramos, and D. Fox, "Guided uncertainty-aware policy optimization: Combining learning and model-based strategies for sample-efficient policy learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 7505–7512.

[24] S. Hoppe, M. Giftthaler, R. Krug, and M. Toussaint, "Sample-efficient learning for industrial assembly using qgraph-bounded ddpg," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 9080–9087.

[25] A. Ranjbar, N. A. Vien, H. Ziesche, J. Boedecker, and G. Neumann, "Residual feedback learning for contact-rich manipulation tasks with uncertainty," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 2383–2390.

[26] Y. Shi, Z. Chen, Y. Wu, D. Henkel, S. Riedel, H. Liu, Q. Feng, and J. Zhang, "Combining learning from demonstration with learning by exploration to facilitate contact-rich tasks," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1062–1069.

[27] G. Wang, M. Xin, W. Wu, Z. Liu, and H. Wang, "Learning of long-horizon sparse-reward robotic manipulator tasks with base controllers," *arXiv e-prints*, pp. arXiv–2011, 2020.

[28] O. Spector and D. Di Castro, "Insertionnet-a scalable solution for insertion," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5509–5516, 2021.

[29] N. J. Cho, S. H. Lee, J. B. Kim, and I. H. Suh, "Learning, improving, and generalizing motor skills for the peg-in-hole tasks based on imitation learning and self-learning," *Applied Sciences*, vol. 10, no. 8, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/8/2719

[30] J. Kober and J. Peters, "Policy search for motor primitives in robotics," *Machine Learning*, vol. 84, no. 1-2, pp. 171–203, 2011.

[31] A. Albu-Schaffer, C. Ott, U. Frese, and G. Hirzinger, "Cartesian impedance control of redundant robots: recent results with the dlr-light-weight-arms," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 3, 2003, pp. 3704–3709 vol.3.

[32] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: Learning attractor models for motor behaviors," *Neural Computation*, vol. 25, no. 2, pp. 328–373, 2013.

[33] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[34] Y. Huang, L. Rozo, J. Silvério, and D. Caldwell, "Kernelized movement primitives," *The International Journal of Robotics Research*, vol. 38, pp. 833–852, 05 2019.

[35] J. Urain, M. Ginesi, D. Tateo, and J. Peters, "Imitationflow: Learning deep stable stochastic dynamic systems by normalizing flows," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020. [Online]. Available: https://www.ias.informatik.tu-darmstadt.de/uploads/Team/JulenUrainDeJesus/2020iflowurain.pdf

[36] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems (NIPS)*, 1999, pp. 1057–1063.

[37] T. Degris, M. White, and R. S. Sutton, "Off-policy actor-critic," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12. Madison, WI, USA: Omnipress, 2012, p. 179–186.

[38] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[39] L. Rozo and V. Dave, "Orientation probabilistic movement primitives on riemannian manifolds," *CoRR*, vol. abs/2110.15036, 2021. [Online]. Available: https://arxiv.org/abs/2110.15036

[40] D. Koert, J. Pajarinen, A. Schotschneider, S. Trick, C. Rothkopf, and J. Peters, "Learning intention aware online adaptation of movement primitives," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3719–3726, 2019.

[41] "Ubongo 3d," May 2021. [Online]. Available: https://www.kosmosgames.co.uk/games/ubongo-3d/

[42] M. Conforti, G. Cornuejols, and G. Zambelli, *Integer Programming*. Springer Publishing Company, Incorporated, 2014.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[44] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, and J. Peters, "Mushroomrl: Simplifying reinforcement learning research," 2021.

[45] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[46] L. Rozo* and V. Dave*, "Orientation probabilistic movement primitives on riemannian manifolds," in *Conference on Robot Learning*, vol. 5, 2021, p. 11. [Online]. Available: https://cps.unileoben.ac.at/wp/orientation_probabilistic_move.pdf, ArticleFile