Residual Learning from Demonstration: Adapting Dynamic Movement Primitives for Contact-rich Insertion Tasks

Todor Davchev[†•*] Kevin Sebastian Luck[†] Michael Burke[†] Franziska Meier[◇]
Stefan Schaal[‡] and Subramanian Ramamoorthy[†]

Abstract— **Contacts and friction are inherent to nearly all robotic manipulation tasks. Through the motor skill of insertion, we study how robots can learn to cope when these attributes play a salient role. In this work we study ways for adapting dynamic movement primitives (DMP) to improve their performance in the context of contact rich insertion. We propose a framework we refer to as residual learning from demonstration (rLfD) that combines dynamic movement primitives (DMP) that rely on behavioural cloning with a reinforcement learning (RL) based residual correction policy. Our evaluation suggests that applying residual learning directly in task space and operating on the full pose of the robot can significantly improve the overall performance of DMPs. We show that rLfD outperforms alternatives and improves the generalisation abilities of DMPs. We evaluate this approach by training an agent to successfully perform both simulated and real world insertions of pegs, gears and plugs into respective sockets.**

I. Introduction

Insertion is a prototypical robot manipulation skill required in a variety of practical applications, ranging from the home to modern manufacturing settings [1]. It remains difficult to find robust and general solutions for this task that do not depend on specialised fixtures and other aids. Mechanical design engineers have long devised ingenious methods for part insertion, however those are either highly case-specific as in manufacturing or not very robust in the face of hardware wear and tear, or other environmental uncertainties. Emerging applications call for a degree of flexibility in skills that may be best achieved through learning and adaptation. This problem is the focus of our work.

Learning from demonstration (LfD) [2] enables effective and rapid skill acquisition in physical robots. This can be particularly useful in contact-rich manipulation tasks such as peg, plug or gear insertions. Existing solutions to LfD [3], [4] have shown that it is possible for robots to gain proficiency in tasks even when we may not have detailed models or simulations of complex environments. However, generalising to a range of scenarios without explicitly modelling the effects of contacts and friction remains difficult, limiting robot learning in these contact-rich settings. Model-free RL has been shown to be very useful for solving challenging tasks [5], [6], but requires large amounts of data and often physically impractical numbers of training episodes. Furthermore, many hours of training can result in higher levels of equipment wear and tear, and increase the risk of more serious hardware damage due to the inherently exploratory nature of RL.

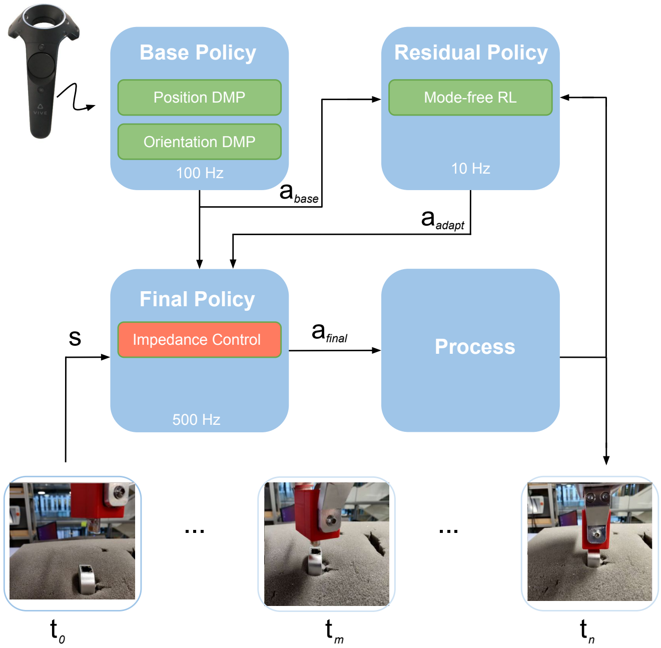


Fig. 1: Outline of the proposed framework. We collect demonstrations using an HTC Vive tracker, and extract an initial full pose policy using Dynamical Movement Primitives (DMPs, running at 100Hz). The control command produced by the DMP is corrected by an additional residual policy trained using model-free RL (run at 10Hz). The resulting term is then fed into a real time impedance controller (running at 500Hz) in a Franka Panda that performs peg, gear or LAN cable insertion in physical setups.

Roboticists have recognised the need for adaptive learning from demonstration schemes that occupy a middle ground [7]. We achieve this by combining LfD and model-free RL in a residual pose correction framework comprised of

This research was supported in part by an EPSRC iCASE award with Thales Maritime Systems provided to Todor Davchev, EPSRC UK RAI Hub NCNR (EPR02572X/1) provided to Kevin Sebastian Luck, and by the Alan Turing Institute, as part of the Safe AI for surgical assistance project through funding Subramanian Ramamoorthy and Michael Burke, all of which were issued to the University of Edinburgh.

[*] Corresponding Author.

[•] Research done in part during a Google X AI Residency.

[†] Authors are with the School of Informatics, University of Edinburgh, 10 Crichton St, EH8 9AB, United Kingdom, {t.b.davchev, michael.burke, s.ramamoorthy}@ed.ac.uk

[‡] Author is with Google X, Mountain View, CA, USA, stefan.k.schaal@gmail.com

[◇] Author is with Facebook AI Research, Menlo Park, CA fmeier@fb.com

Range of Considered Tasks

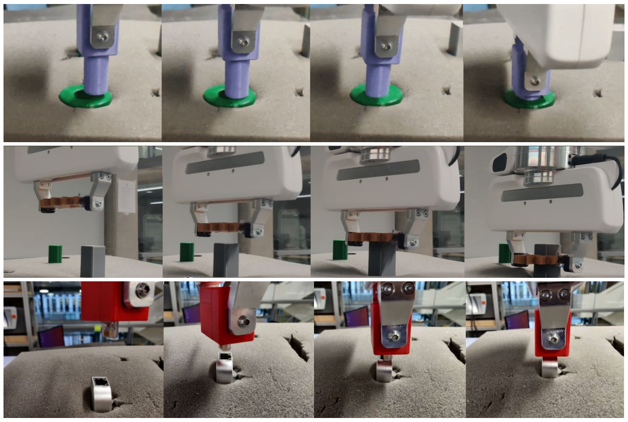


Fig. 2: The considered peg, gear and LAN cable insertion.

two complementing policies. DMPs [8] act as a base policy which we further correct using variants of state-of-the-art on- and off-policy RL trained in an online fashion. This yields a solution that is more sample-efficient and is applicable to a range of start positions, orientations, geometries as well as levels of friction. We study the efficacy of this framework using peg, gear and LAN cable insertions - in simulation and on a physical robot (Figure 2).

DMPs are ubiquitous in behaviour cloning settings, and a popular technique for learning from demonstration. Although existing work has explored adapting DMP or coupling terms directly to improve task generalisation [9]–[11], it is still unclear how best to do this. This work exhaustively explores DMP correction techniques and shows that a residual correction policy in task space is the most effective strategy in insertion contexts. Furthermore, this framework explicitly accounts for orientation-based policy corrections in task space. Our results show that orientation-based correction, which is typically not applied in residual learning frameworks [12], is essential for reliable and robust plug insertion. This is challenging in part because additive orientations can introduce catastrophic singularities and locks while formulating residual orientation-based corrections in quaternion space can be difficult to learn.

Contributions The key contributions of this paper[1] are:

C1 *A general framework for residual full pose corrections of DMPs and the demonstration of its applicability to three types of physical insertion tasks;*
C2 *Showing that learning a residual corrective policy using model-free RL applied directly in task space is more robust and sample efficient than adapting DMP parameters in forcing or coupling term space;*
C3 *Showing that a residual formulation results in less forceful and more sample efficient policies than alternative schemes.*

[1] More details and videos are available at `https://sites.google.com/view/rlfd/`.

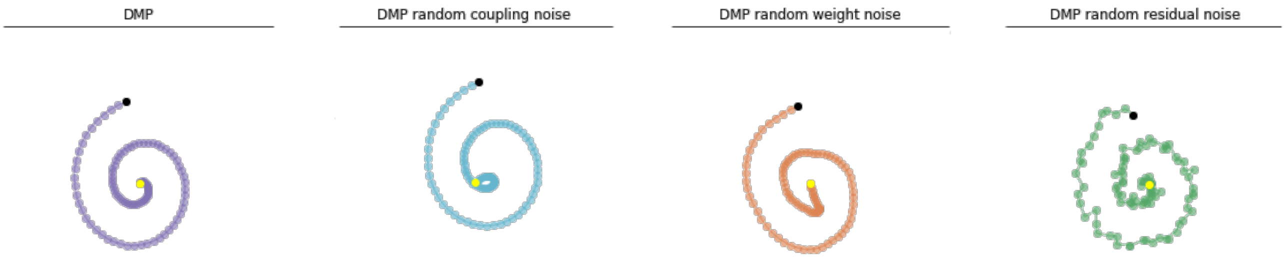
II. Related Work and Contributions

Insertion is a broadly relevant motor skill in many robot manipulation applications. It is a key part of manufacturing assembly, home automation, laboratory testing and even surgical automation [1]. Existing literature on this problem can be organised into two broad categories based on whether or not the methods split the task execution into phases based on contact times. This section provides an overview of some of these methods, and positions our work in context.

Modelling Contacts It is common practice in extant approaches to first model the time(s) of contact and then to structure the insertion strategy in terms of two separate sub-problems - contact-state recognition and compliant control [7]. Compliant control generally relies on a human engineer to characterise the underlying friction and contact model [13]. Some recent approaches have demonstrated high-precision assembly through the use of additional force information [14]. These methods require careful design of the manipulated objects and the external environment, e.g. assuming a flat surface surrounding the hole. In many practical settings, these assumptions may not hold, e.g. consider RJ-45 connector insertion (Figure 2, bottom row). Also, it tends to be hard to obtain good models of the distributional changes in contact dynamics over time, which affects reliability. In contrast, rLfD overcomes these limitations by relying on a strategy extracted from human demonstrations, and a learned policy that adapts the full pose of the end-effector directly in task space.

Learning from Demonstration (LfD) Learning policies for complex robot tasks can benefit from expert demonstrations [15], [16]. LfD [2] is a widely adopted approach formalising this idea and has been applied previously in the context of peg-in-hole insertion [17]. However, no further effort was made to account for model imperfections. DMPs [8] have been shown to improve the generality of considered demonstrations in a variety of manipulation tasks [3], [9], [11], [18]. However, these works do not look into contact-rich manipulation scenarios. The methods proposed in Vecerik et al. [19] and Luck et al. [20] use positive and negative demonstrations to initialise a replay buffer and use off-policy deep RL to solve an insertion task using only sparse rewards. Vecerik et al. [5] uses demonstrations in a similar way, but relies on visual guidance to solve for randomised target positions. These solutions utilise demonstrations implicitly through a replay buffer relying on sample heuristics that often require more demonstrations and training time which might lead to damaging the robot or the considered plug. In contrast, rLfD adapts the full end-effector (EEF) pose extracted from as little as a single demonstration and requires ~ 600 episodes to solve the considered insertion tasks.

Reinforcement Learning (RL) The standard approach to using RL in robotics tasks is to learn incrementally from trial and error. For tasks such as insertion, this can be impractical as the approaches may not run in real time, or consider issues of concurrency in control. There is also



$$\dot{y} = \frac{1}{\tau^2}(\alpha_v(\beta_v(g-x) - \tau y) + z f_{\omega+\boxed{\eta}} + C_t(\boxed{\eta})) + \boxed{\dot{\eta}}$$

Fig. 3: Types of adaptive perturbations with Gaussian noise, η, applied on a positional DMP policy from the equation above for a simple Archimedian spiral. Perturbing directly in task space, $\dot{y}$ (green) results in local exploration that is important for contact-rich manipulation. In contrast, perturbing the coupling term, C_t (blue) or the parameters ω of the forcing term, f_ω (orange) results in locally smooth trajectories. At test, we remove stochasticity and use the mean action instead which leads to a smoother trajectory.

the issue of the number of training episodes needed, which can further affect practicality. Nevertheless, these algorithms exhibit promising generalisation abilities. Some notable recent approaches focus on improving the state representation for conventional model-free solutions [21], [22]. There has been interest in model-based approaches [23], [24], and more recently hybrid approaches [25]. However, it remains difficult to scale these solutions to the scenarios we have in mind. Further, our evaluations suggest that hybrid policies that rely on model-free learning may result in applying high levels of force during execution, which increases the risk of hardware damage during training.

Residual RL Acknowledging the difficulties of executing RL in physical systems, works such as Johannink et al. [26] combine conventional contact model-based control with model-free reinforcement learning. However, they do not handle orientation-based corrections - instead, they focus primarily on sliding at low levels of friction. Schoettler et al. [12] proposes using stronger priors and combines SAC with a proportional (P) controller to solve industrial position-only insertion tasks. They do not scale to full pose corrections too. Moreover, this approach encourages the use of higher forces via a dense reward function. This limits the solution to Cartesian space perturbations and increases the risk of hardware damage while reducing the generality of the proposed solution.

Summary We propose a framework that bridges LfD and model-free RL through an adaptive residual learning policy operating alongside dynamic movement primitives applied directly to the full pose of the robot. We explore the effects of parameter adaptation for DMPs, and show that residual policy learning is more effective than model adaptation schemes that modify DMP parameters directly. Further, the proposed formulation leads to a significant reduction in the sample requirements during training while preserving the overall improved accuracy achieved with model-free RL. In practice, this is important as it limits the risk of hardware damage and thus makes this approach feasible for real world applications. This framework is evaluated in both simulated and physical systems, running in real time.

III. Methods

In this work we learn a base policy, π_b, from expert demonstration and apply it directly in task space. We use two separate formulations - namely positional and orientational DMPs [8], [27]. The learned behaviour is then executed using a suitable hardware controller, i.e. positional or impedance, that produces the final motions of the robot arm. The positional base policy $\hat{\pi}_b$ is described by position and velocity terms, $(\mathbf{x}, \dot{\mathbf{x}})$, in 3D Cartesian space. The orientation-based base policy, $\tilde{\pi}_b$ is described using the 4D orientation in quaternion and angular velocity, $(\mathbf{q}, \boldsymbol{\omega})$. The resulting base policy, π_b is then capable of imitating the full pose of the demonstrated behaviour. We learn π_b using a single expert demonstration (see Figure 7a). Such a formulation can generalise well to perturbations of the initialisation and timing conditions especially when executed in free space. However, it cannot adapt well to previously unseen task settings and environmental perturbations which are typical for contact-rich manipulation tasks. In this work we study how to alleviate this limitation using an additional residual policy, π_θ by employing state-of-the-art model-free RL.

A. Problem Formulation

We consider a finite-horizon discounted Markov decision process (MDP), $\mathcal{M} = (S, A, P, r, \gamma, T)$ with system dynamics defined by a transition function $P : S \times A \times S \mapsto [0, 1]$ and a reward function $r(x, a) \in \mathbb{R}$. Let $x \in S \subseteq \mathbb{R}^{n_x}$ be an element in the state space S, and $a \in A \subseteq \mathbb{R}^{n_a}$ denote the robot velocity which we refer to as action. Then, a stochastic control policy parameterised by θ, $\pi_\theta(a|x)$ defines the probability of selecting an action a given a state x. Let $\zeta = (\mathbf{s}_{t_0}, \ldots, \mathbf{s}_T)$ be a trajectory with a horizon T, a discount function $\gamma(\cdot)$ and a state-action tuple $s_t = (x_t, a_t)$ and a trajectory return $R(\zeta) = \sum_{t=t_0}^{T} \gamma r(x_t, a_t)$. In this context, we can define an optimal policy as $\pi^* = \arg\max_{\pi \in \bar{\pi}} J(\pi)$, where $J(\pi) = \mathbb{E}_{s_0 \sim p(s_0), a_t \sim \pi(s)}[R(\zeta)]$ and $\bar{\pi}$ is the set of all policies π.

We can use a policy gradient method to optimise the objective from the sampled trajectories. By relying on a DMP to extract a fixed and pre-computed offline continuous base policy, π_b, we significantly reduce the complexity of the task for π_θ. The residual policy has to learn how to deviate from the base policy by correcting for model inaccuracies and potential environmental changes during execution. Thus, the final policy can compensate for system uncertainties through an adaptive term sampled from π_θ while relying on a prior, π_b acquired through human demonstrations.

B. Residual Learning with DMP

Contact-rich manipulation has inherent non-linear dynamical effects between the robot and the surrounding environment. In most insertion tasks, slight perturbations can have drastic consequences on the performance of π_b. To this end, utilising model-free solutions can help relax the challenge of modelling contact dynamics. However, very little has been done to study the effects of applying residual signals to a DMP formulation in the context of contact-rich manipulation. In this subsection, we propose a formulation that directly adapts the positional $\hat{\pi}_b$ in task space (Figure 3).

We propose extracting a residual positional policy $\hat{\pi}_\theta$, parameterised by θ that adapts the base policy, $\hat{\pi}_b$ to compensate for model inaccuracies and environmental uncertainties. The final positional policy can then be defined as $\hat{\pi}_f = \hat{\pi}_b + \hat{\pi}_\theta$.

Corrective actions $\hat{\mathbf{a}}_\triangle \sim \hat{\pi}_\theta(\hat{\mathbf{a}}|\mathbf{s})$ for a stochastic policy can be modelled as a Gaussian distribution with state $\mathbf{s}$ and action $\mathbf{a}$. The state $\mathbf{s}$ can be comprised of the current pose and sensed force of the robot $(\mathbf{x}, \mathbf{q}, \mathbf{f})$, with its position $\mathbf{x}$ relative to the goal and $\mathbf{q}$ being a unit quaternion and the calibrated 3D force, $\mathbf{f}$ at the end-effector. Applying the 3D position and velocity correction to $\hat{\pi}_b$ requires a straightforward addition, and namely $\hat{\mathbf{a}}_f = \hat{\mathbf{a}}_b + \hat{\mathbf{a}}_\triangle$, similar to Johannink et al. [26].

The adaptation process of contact-rich manipulations can benefit from local policy perturbations that enable effective exploration. However, applying corrections to different components of the DMP formulation can lead to varying results. Figure 3 illustrates this notion with a simple Archimedian spiral and a positional base policy $\hat{\pi}_b$ extracted from a single demonstration. In this context, correcting the weights, ω, of the nonlinear term f_ω (orange), for example, can lead to locally smooth but very distinct nonlinear motions. This itself is not that useful in settings that require careful local explorations. Similarly, applying residual corrections as part of the coupling term formulation, C_t (blue) leads to changing the overall smooth behaviour but does not take into account the local exploration requirements. Thus, we propose to apply corrections directly in task space, $\dot{y}$ (green). This allows extracting residual policies that can effectively learn to adapt to a range of perturbations by performing local online exploration through adding noise to the $\dot{x}$ or $\ddot{x}$ term directly at the end-effector task space, $x_{t+1} = x_t + \hat{a}_f$. We propose to extract this policy by relying on model free RL. The final learned policy is no longer comprised of random noise. Instead, it turns into a structured corrective term that operates in velocity or acceleration space. Our empirical evaluation suggests that rLfD leads to a smooth, less forceful solution compared to alternative approaches.

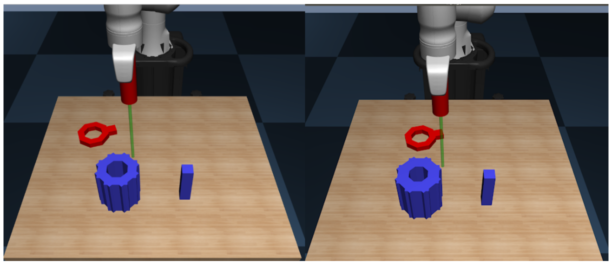


Fig. 4: An easy task (left) and a harder task (right). The robot is initialised with a position sampled from a uniform distribution centered at the demonstrated trajectory's initial position with variance of ± 11.8cm along all dimensions-chosen randomly from a range between 10 and 12, distances that led to tasks that were challenging enough for a base policy π_b to solve. The difficulty is defined by the size of the hole. A task is complete when the peg is fully inserted.

C. Learning Position and Orientation based Corrections

Given environmental changes and assuming inaccuracies in the robot itself, the positional residual formulation introduced above can correct this by combining an adapting policy $\hat{\pi}_\theta$ using the DMP as a prior policy $\hat{\pi}_b$, resulting in $\hat{\pi}_f = \hat{\pi}_b + \hat{\pi}_\theta$.

This form of additive correction is unsuitable for orientations due to the risk for singularities or locks. Assuming static orientations or constant angular velocity could in principle relax this constraint. However, such approximations would restrict applicability as motion is not static in orientation by nature. In order to address this, a residual formulation capable of accommodating quaternion based representations is needed. Thus, we define an orientation-based residual framework given by

$$\tilde{\pi}_f = \tilde{\pi}_\theta \circ \tilde{\pi}_b, \tag{1}$$

where $\circ$ describes composition in Shuster's notation. Then, the resultant quaternion correction is associative and distributive, but not commutative. However, directly predicting orientation as a quaternion is hard. This makes it challenging to predict feasible actions as a learned term. To accommodate this, we employ an angle-axis representation output which consists of a 1D desired angle α of rotation in radians and a 3D unit vector $\mathbf{r}$ around which the rigid body is rotated with angle α, resulting in an action $\tilde{\mathbf{a}} = \{\alpha, \mathbf{r}\}$. Assuming that $\alpha \in [-\pi, \pi]$, which covers all rotations, it follows that $cos(\alpha/2)$ is strictly positive. Then, a correction $Q_\triangle = \{q^w_\triangle, q^x_\triangle, q^y_\triangle, q^z_\triangle\}$ can be computed as $q^w_\triangle = cos(\alpha/2)$ for the real part of the quaternion, $Q_\triangle$, and a rotation vector $\mathbf{r}$ defined as

$$\mathbf{r}_\triangle = \frac{\omega^k}{||\boldsymbol{\omega}||}. \tag{2}$$

Here, ω^k is one dimension of the rotation vector $\mathbf{r}$, for $k \in \{x, y, z\}$ and $||\boldsymbol{\omega}||$ is the L2 norm of the vector. The

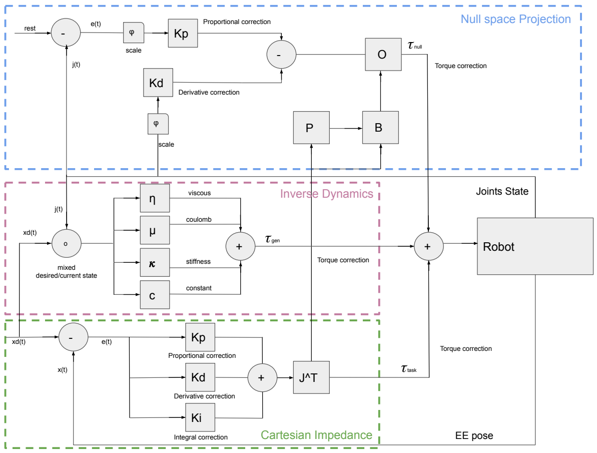


Fig. 5: Cartesian Impedance controller with null space correction and without any dynamics model.

orientational adaptation can be described as quaternion by

$$Q_\triangle = [cos(\alpha/2),\ \mathbf{r}_\triangle sin(\alpha/2)]. \tag{3}$$

This allows us to obtain a quaternion correction term using real numbers. However, adding the correction to the prior orientation is achieved with quaternion multiplication thus

$$Q_f = Q_\triangle \circ Q_b. \tag{4}$$

Therefore, the complete final policy π_f is defined as the concatenation of both 3D Cartesian positional policy $\hat{\pi}_f$ and the 4D quaternion-based orientation policy $\tilde{\pi}_f$ and is thus given by

$$\pi_f = \begin{bmatrix} \hat{\pi}_f \\ \tilde{\pi}_f \end{bmatrix}. \tag{5}$$

Implementation Details Both environments are built using Tf-Agents [28]. Training is performed using a sparse reward with small margin κ, selected based on the width of the walls and depth of the hole. All of the proposed policies rely on a normal projection for continuous actions which uses $\tanh$ to limit its output similar to Haarnoja et al. [29]. Further details can be found on the project page[2].

This work makes use of two independent, publicly available environments - MuJoCo [30] and SL [31] for experimenting in simulation and the physical world respectively. In simulation, we utilise Robosuite [32] and use a position controller applied on low level actions obtained using inverse kinematics with PyBullet [33]. To evaluate rLfD in the physical world, we used a Jacobian transpose Cartesian impedance control (see Figure 5 for a control diagram) run in real time. Both controllers were run at 500Hz, while π_b at 100Hz and π_θ at 10Hz (see Figure 1).

The framework supplies set-points to a real-time running controller, and uses a real-time clock to synchronise the concurrently running modules. We extrapolate the DMP corrections using a standard fifth order polynomial while assuming static repetition of the produced residual term.

IV. Experimental Evaluation

We evaluate the effects of using adaptive corrections in the context of skill acquisition with residual learning from demonstration. We perform a sequence of empirical evaluations using a 7 DoF Panda Franka Emika arm. Our results indicate that direct pose corrections using model-free reinforcement learning techniques is both sample efficient and improves the performance and generality of DMPs in contact rich manipulation tasks. We split this section in two main parts: a study of the effects of using residual corrections on a simulated environment using MuJoCo [30] and Robosuite [32]; and applying rLfD to the physical world on three different in complexity insertion tasks.

A. Applying Residual Corrections

Applying adaptive corrections is a widely studied approach to improving the performance of a base policy as discussed in Section II. However, little has been done towards studying the effect of correcting different parts of the DMP (See Fig. 3). To this end, we provide an exhaustive study of applying corrections to a DMP in the context of contact-rich manipulation. Our results show that adapting DMP's with residual policy in task space directly, instead of forcing term or coupling term space is more robust and sample efficient than the alternatives considered.

We study the utility of applying residual corrections in the context of learning to insert. We utilise two separate tasks we refer to as 'easy' and 'hard' (see Figure 4). The difficulty of a task is defined by the size of the hole a peg has to be inserted in. A tighter hole indicates more difficulties such as potential jamming due to friction during insertion. The task with a larger hole could be solved using only position-based corrections. In this section, we train on the easy task but evaluate on both.

TABLE I: Insertion accuracy table. Model names are followed by the type of correction applied, coloured according to Fig. 3. Eff. is efficiency- the number of episodes a model was trained for.

Model/Task	Easy	Hard	Average	Eff.	Reward
No corrections	31/100	8/100	20/100	n/a	n/a
Random position	34/100	8/100	21/100	n/a	n/a
Linear position [34]	34/100	8/100	21/100	n/a	n/a
eNAC coupling [9]	0/100	0/100	0/100	8K	$exp\{-L_1\}$
FD parameters [11]	79/100	45/100	62/100	8K	$exp\{-L_1\}$
PoWeR parameters [10]	55/100	28/100	42/100	8K	$exp\{-L_1\}$
(no DMP) SAC position [29]	96/100	47/100	71/100	17K	$-(\alpha * L_1 + \frac{\beta}{L_2 - \epsilon})$
(hybrid) SAC position	46/100	29/100	37/100	8K	$\mathbb{1}[L_2 \leq \kappa]$
SAC position (ours)	**98/100**	65/100	81/100	**600**	$\mathbb{1}[L_2 \leq \kappa]$
PPO position (ours)	94/100	**74/100**	**84/100**	25.5K	$\mathbb{1}[L_2 \leq \kappa]$

We summarise the effect of correcting different aspects of the movement primitives formulation after training on the 'easier' task in Table I. We used 40 basis functions with exponentially spaced kernels to learn π_b and a single demonstration. We used a single demonstration in our evaluation to extract π_b as we found that using average over multiple demonstrations more likely to hit a surrounding wall as summarised in Figure 7a. We train rLfD using a 0/1 sparse reward with margin $\kappa = 0.048$. However the baselines performances were significantly worse with sparse

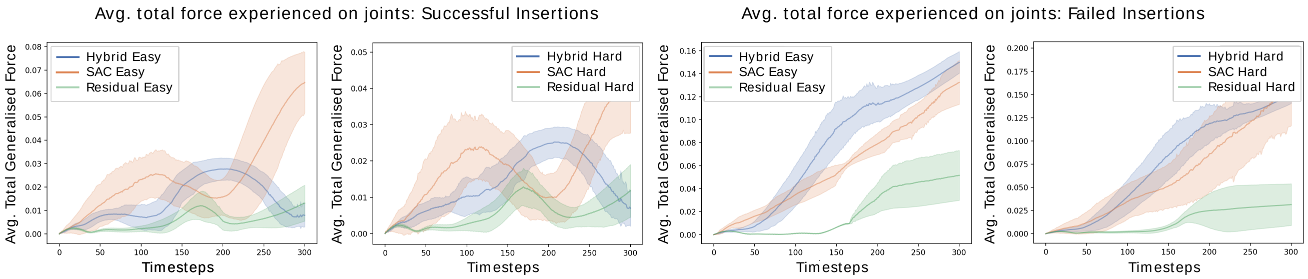
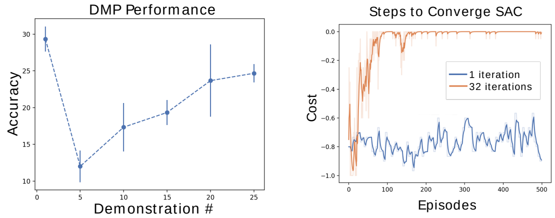


Fig. 6: Comparison between using residual, hybrid, DMP and model-free policies. The residual policy (green) consistently results in experiencing forces comparable to that using only a DMP (red) and even lower force across both the easy and hard tasks. Lower is better.

rewards. Therefore, we chose to take the reward function that is typically used in literature for each of them. This choice improved those baselines' performance so we report those results while measuring accuracy the same way across all methods. In all cases we used the distance of the end-effector from the goal as the input state. Our findings indicated that using only SAC with no prior (row 7) can get comparable performance to rLfD. However, this relies on a well engineered reward function and required significantly more episodes to reach that performance. We used $\alpha = 10$, $\beta = 0.002$, $\epsilon = 0.0001$ for the engineered dense reward case.

A promising related approach is to solve the problem with a hybrid formulation. Lee et al. [25] learn policies from visual pre-trained state space representations with SAC in a sample efficient manner, by switching between model-based and model-free policies. We also applied a similar switching strategy to the proprioceptive state representations used by rLFD, switching between a base-policy (learned via DMP) and a model-free policy. In our experiment and task, we found switching to perform worse than rLfD when measured on overall experienced force on all joints and insertion accuracy. This seems to indicate that rLfD may also perform well if the more informative, pre-trained visual representations of Lee et al. [25] are included. We leave this study for future work.

We also considered learning corrective residual policies using both on- and off-policy algorithms. In both cases, we relied on the DMP policy only for the first half of an episode and used both the base and residual policies for the rest of the execution. While using PPO [35] performs better both on average and in terms of generalising to harder tasks, it required a higher number of steps to reach this level of performance. In contrast, using SAC [29] can produce similar results in only a fraction of the required steps (600 episodes). We reduce the exploration requirements from 50 to less than 13 thousand steps (∼600 episodes) for the cost of training time. We used a fixed alpha term with $\alpha = 0.01$ and a recurrent neural policy. In the simulation-based scenario, we used a batch of size 32 and a single training step per iteration. We reduced episode requirements significantly by increasing the update steps to 32 which required uniformly sampling from the same replay buffer multiple times per update step. This resulted in 4 times fewer episodes required for a policy to converge. Figure 7b illustrates a simpler task where the robot position was initialised from a uniform distribution, centred at the initial position of the provided demonstration with a variance of $\pm 3cm$. Finally, we used a target update τ value of 0.05 and sampled sequence of 20 steps. More details and experiments can be found on the project web page[2].



(a) Fitting a DMP using averaged demonstrations generalised less than a single demo.

(b) More iterations reduce the number of episodes required but increase training time.

Our analysis (Figure 6) showed that relying on SAC in both hybrid (blue) and purely model-free (orange) approaches resulted in applying more force during an execution compared to rLfD (green). The DMP (red) failed at times by constantly pushing downwards, which contrasts with the less forceful searching of rLFD.

TABLE II: Overall performance on the 3 tasks considered. The baseline policies are named as *base / position / orientation* policy while 'rLfD (PPO)' uses *PPO* for the full pose. We propose a reliable solution that is faster, more accurate and generalising better than the alternatives considered.

Task	Policy	Accuracy	Generalisation	Data Efficiency	Speed
	DMP	52.62% ±0.71	47.87% ±1.27	1 episode	7sec ±0.1
Peg	DMP / RLS / n/a	80.66% ±3.95	79.72% ±4.78	$\leq$ 10 episodes	5.4sec ±0.2
Insertion	DMP / RLS / Unif	60.25%	60.25%	$\leq$ 10 episodes	7.1sec
	DMP / Unif / Unif	91.02%	90.97%	1 episode	6.3sec
	rLfD (PPO)	**97.94%** ±1.2	**97.08%** ±0.7	$\leq$ 500 episodes	**5.1sec** ±0.1
	DMP	41.52% ±1.5	35.27% ± 2.19	1 episode	8.1sec ± 0.2
Gear	DMP / RLS / n/a	58.67% ± 1.08	53.33% ± 1.22	$\leq$ 10 episodes	7.2sec ± 0.1
Insertion	DMP / RLS / Unif	76.19% ±2.57	73.12% ±2.9	$\leq$ 10 episodes	6.6sec ± 0.1
	DMP / Unif / Unif	86.86% ±1.68	85.16% ±1.9	1 episodes	6.2sec ± 0.1
	rLfD (PPO)	**92.2%** ± 2.6	**91.4%** ±3.1	$\leq$ 500 episodes	**5.9sec** ±0.1
	DMP	0.0%	0.0%	1 episode	10sec
LAN Cable	DMP / RLS / n/a	14.6% ±1.6	10.0%	$\leq$ 10 episodes	9.6sec ± 4.6
Insertion	DMP / RLS / Unif	57.3% ±2.5	51.2% ± 2.9	$\leq$ 10 episodes	8.7sec ± 0.1
	DMP / Unif / Unif	64.8% ±1.2	60.0% ±1.5	1 episode	8.6sec ±0.1
	rLfD (PPO)	**70.6%** ±1.4	**66.4%** ±1.5	$\leq$ 500 episodes	**8.4sec** ±0.0

[2] `https://sites.google.com/view/rlfd/additional-experiments`

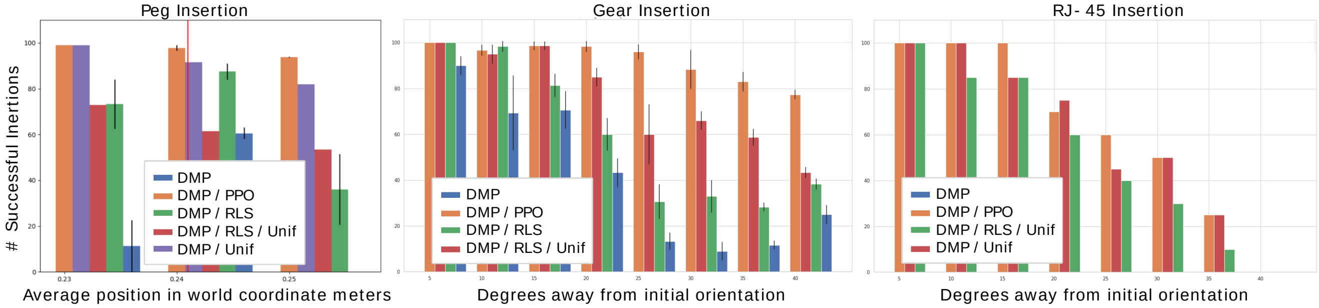


Fig. 8: Successful peg, gear and RJ-45 insertions utilising LfD (blue) with (DMP /·) and without residual corrections. Peg insertion results are cm away from the initial position (illustrated by a red vertical line). Gear and RJ-45 results are with respect to degrees away from initial orientation. Evaluated against 500 samples. Higher is better.

B. Pose Corrections with Concurrent Real-time Control

In this subsection we scale rLFD to correct the full pose in a sequence of real-world insertion tasks using a less precise impedance controller. We consider peg insertion, similar in setup to the simulated tasks discussed earlier, as well as gear and RJ-45 connector insertion (see Figure 2). Each of those tasks introduces an additional mode of complexity within the context of insertion. While physical peg insertion poses challenges due to the friction-heavy interactions with the highly nonlinear surrounding world, it can be solved by relying mostly on positional corrections. On the other hand, the gear insertion task requires perfect alignment of the squared peg with the hole. Due to the tight fit, it also requires a certain amount of downward force to be able to achieve insertion. Finally, inserting an RJ-45 connector requires both perfect position and orientation. Moreover, the connector's plastic tip requires that a precise amount of force be applied to avoid breaking, further increasing task complexity.

In practice, random noise corrections (as in Figure 3) or an iterative linear residual policy might be sufficient to achieve near perfect robustness [34]. A major factor for this is the compliance of the physical environment, which is difficult to simulate. However, latent external factors in the environment might have a significant impact on the overall performance. For example, incorrect levelling of the surface a hole is positioned on may be hard to notice but can result in unexpected failures. Learning to cope with such biases is thus important. Further, extracted policies must be able to generalise to starting poses outside the training distribution and solve the task as quickly as possible. In this section, we study the overall accuracy, generalisation ability, speed of completing a task as well as the efficiency of using random, linear and non linear corrective policies as part of the rLfD formulation. Our experiments suggest that a non-linear policy extracted using the model-free approaches does provide the best performance and can overcome some of the limitations of the alternatives considered (Table II).

In the peg insertion case, we chose a set of 500 starting points that were sampled randomly from six, three dimensional uniform distributions, each centred at up to +/- 3cm away along each axis from the demonstrated starting pose. In the gear and LAN insertion tasks, we sampled uniformly 160 unit vectors which were split into 8 bins of 20 samples, spreading up to 40 degrees away from the initial demonstrated orientation. Table II summarises the overall performance of the proposed approach on the three tasks.

We found that recursive least squares (RLS) performed best compared to other linear policies and thus report it as representative of the linear policy family. We used publicly available implementation provided as part of padasip[3]. The associated reference [34] aims to give attribution to the idea of adapting with a linear policy and does not make a general statement of the performance of the full solution. We used separate filters with scalar outputs for each of the 3 positional output dimensions of the policy. We assume state-dependent predictions, but a time dependent policy due to the nature of the quadratic cost. Inputs were of degree 2 for each dimension; e.g. $(x^2, 2xy, 2xz)$ was used as input for the prediction of the x axis. Then, an actions was defined by the difference between the RLS predictor and the current state. Uniform policy assumed the same structure but sampled actions from a uniform distribution.

As state space to the non-linear policy, we used the orientation and the L2 distance between the tip and the demonstration goal, along with the normalised 3D force experienced at the EEF as well as the action produced by the DMP. The test environment had a small, one degree slope which seems to have made policies pushing away from the robot more effective. The RL agent successfully compensated for the environment and learned a successful corrective policy. Finally, we found that starting from a higher position decreases the performance of baseline policies as there was less time to rely on the compliance of the environment[2]. Figure 8 depicts the results achieved on the insertion tasks. Using a linear policy such as RLS is not directly applicable to the RJ-45 insertion task since the required target trajectory needs to be extracted from a successful DMP insertion but the basic DMP formulation was unable to solve the task at all. Instead, we can use an adapted DMP with uniform noise to extract a successful solution. This is what we used to populate the relevant results in Table II.

[3] https://pypi.org/project/padasip/

V. Conclusion

This work introduces the notion of residual learning from demonstration (rLfD), a framework that improves the generalisation abilities of DMPs in the context of contact-rich insertions both in simulation and in real-world experiments. rLfD specifically accounts for orientation corrections. Our results highlight the benefits of applying pose corrections directly in task space, rather than DMP parameter space, both in terms of accuracy and overall experienced force on the joints. rLFD was shown to be more sample efficient than alternative model-free approaches, and able to find better and faster solutions when compared to linear and random policies. Potential future extensions to this framework include consideration of hardness in terms of sequences of contacts and investigating ways to reduce the overall forces applied during insertion as well as utilising visual policy inputs.

Acknowledgment

The authors would like to thank Giovanni Sutanto, Ning Ye, Daniel Angelov, Martin Asenov and Michael Mistry for the valuable insights and discussions on drafts of this work.

References

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

[2] S. Schaal, "Learning from demonstration," in *Advances in neural information processing systems*, 1997, pp. 1040–1046.

[3] G. Sutanto, Z. Su, S. Schaal, and F. Meier, "Learning sensor feedback models from demonstrations via phase-modulated neural networks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1142–1149.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[5] M. Vecerik, O. Sushkov, D. Barker, T. Rothörl, T. Hester, and J. Scholz, "A practical approach to insertion with variable socket position using deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 754–760.

[6] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, A. M. Agogino, A. Tamar, and P. Abbeel, "Reinforcement learning on variable impedance controller for high-precision robotic assembly," *arXiv preprint arXiv:1903.01066*, 2019.

[7] M. T. Mason, "Toward robotic manipulation," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 1–28, 2018.

[8] 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.

[9] J. Peters and S. Schaal, "Applying the episodic natural actor-critic architecture to motor primitive learning." in *ESANN*, 2007, pp. 295–300.

[10] J. Kober and J. R. Peters, "Policy search for motor primitives in robotics," in *Advances in neural information processing systems*, 2009, pp. 849–856.

[11] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

[12] 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," *arXiv preprint arXiv:1906.05841*, 2019.

[13] D. E. Whitney, "Quasi-static assembly of compliantly supported rigid parts," *Journal of Dynamic Systems, Measurement, and Control*, vol. 104, no. 1, pp. 65–77, 1982.

[14] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 819–825.

[15] I. Havoutis and S. Ramamoorthy, "Motion planning and reactive control on learnt skill manifolds," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1120–1150, 2013.

[16] 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," *arXiv preprint arXiv:1802.09564*, 2018.

[17] Z. Zhu, H. Hu, and D. Gu, "Robot performing peg-in-hole operations by learning from human demonstration," in *2018 10th Computer Science and Electronic Engineering (CEEC)*. IEEE, 2018, pp. 30–35.

[18] F. Stulp, E. A. Theodorou, and S. Schaal, "Reinforcement learning with sequences of motion primitives for robust manipulation," *IEEE Transactions on robotics*, vol. 28, no. 6, pp. 1360–1370, 2012.

[19] 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.

[20] K. S. Luck, M. Vecerik, S. Stepputtis, H. B. Amor, and J. Scholz, "Improved exploration through latent trajectory optimization in deep deterministic policy gradient," *arXiv preprint arXiv:1911.06833*, 2019.

[21] G. Thomas, M. Chien, A. Tamar, J. A. Ojea, and P. Abbeel, "Learning robotic assembly from cad," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[22] 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 *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8943–8950.

[23] S. Levine, N. Wagener, and P. Abbeel, "Learning contact-rich manipulation skills with guided policy search," *arXiv preprint arXiv:1501.05611*, 2015.

[24] J. Luo, E. Solowjow, C. Wen, J. A. Ojea, and A. M. Agogino, "Deep reinforcement learning for robotic assembly of mixed deformable and rigid objects," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2062–2069.

[25] 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," *arXiv preprint arXiv:2005.10872*, 2020.

[26] 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.

[27] P. Pastor, L. Righetti, M. Kalakrishnan, and S. Schaal, "Online movement adaptation based on previous sensor experiences," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 365–371.

[28] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo, "Tf-agents: A library for reinforcement learning in tensorflow," https://github.com/tensorflow/agents, 2018, [Online; accessed 25-June-2019]. [Online]. Available: https://github.com/tensorflow/agents

[29] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *arXiv preprint arXiv:1801.01290*, 2018.

[30] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[31] S. Schaal, "The sl simulation and real-time control software package," Technical report, University of Southern California, Tech. Rep., 2009.

[32] L. Fan, Y. Zhu, J. Zhu, Z. Liu, O. Zeng, A. Gupta, J. Creus-Costa, S. Savarese, and L. Fei-Fei, "Surreal: Open-source reinforcement learning framework and robot manipulation benchmark," in *Conference on Robot Learning*, 2018.

[33] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[34] V. Kumar, E. Todorov, and S. Levine, "Optimal control with learned local models: Application to dexterous manipulation," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 378–383.

[35] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

VI. Appendix

A. Base Policy from Demonstrations

As a base policy, this framework learns a trajectory-based DMP representation [8] from demonstrations, which can be executed using a suitable controller (i.e. position or impedance). In order to allow for orientation and position-based residual corrections we use separate DMP formulations.

Positional DMP Position-based DMP [8] rely on a transformation system, responsible for generating a trajectory plan, a canonical system responsible for timing the motion generation, and a forcing term, that is a nonlinear function approximation term which shapes the attractor landscape. In this context the transformation system can be written as

$$\tau^2 \dot{y} = \alpha_v(\beta_v(g - x) - \tau y) + z f_\omega + C_t, \tag{6}$$

which assumes a one-dimensional system, for some time duration τ, and and a velocity $y = \dot{x}$. The nonlinear forcing term f_ω is scaled by $z = \frac{g - x_0}{g_{demo} - x_{0,demo}}$, that is the ratio of distances between the start position x_0 and the goal position g recorded during demonstration. In order to scale a motion primitive to varying time duration, the canonical system utilises a phase variable ρ, representing the current phase of the primitive. Finally, the transformation system is driven by a nonlinear forcing term f_ω and a coupling term C_t. The forcing term f_ω is responsible for the final shape of a primitive and is typically modelled as a weighted sum of N Gaussian basis functions ψ_i which are functions of the phase s, with width parameter h_i and centre at c_i. Mathematically, this can be expressed by

$$f_\omega(\rho) = \frac{\sum_{i=1}^{N} \psi_i(\rho) w_i}{\sum_{i=1}^{N} \psi_i(\rho)} \rho, \tag{7}$$

for $\psi_i(\rho) = exp(-h_i(\rho - c_i)^2)$. The forcing term weights w_i are parameters learned from human demonstration. In addition to the forcing term, the transformation system can also be modified by a coupling term C_t, a sensory coupling, which can be either state-dependent or phase-dependent or both.

In this work, we consider three degrees of freedom (DoF) position-based systems. Each DoF has its own transformation system but all DoF share the same canonical system.

Orientational DMP Although position based DMPs have proved to be sufficient for a variety of tasks, the task of insertion is highly dependent on orientation. Thus, we utilise an orientation-specific DMP, first introduced in [27]. Like positional DMPs, orientational DMPs have a transformation and a canonical system [3]. We define a transformation system as

$$\tau^2 \dot{\omega} = a_\omega(\beta_\omega 2 log(Q_g \circ Q^*) - \tau\omega) + f + C, \tag{8}$$

for a unit quaternion Q which represents the orientation. Then, Q_g is the associated goal orientation and ω, $\dot{\omega}$ are the 3D angular velocity and acceleration terms. C and f are the associated coupling and forcing terms. Quaternion DMPs, however, can no longer rely on simple addition and multiplication for the hypercomplexity of the quaternion representation. Instead we need to rely on the operator $\circ$ (composition) and * (conjugation). Moreover, we require the generalised log and the exponential maps ($log(\cdot)$, and $exp(\cdot)$), similar to [3].

Thus, the final base policy π_b proposed in this work uses both the end-effector pose and velocities, described by $(\mathbf{x}, \dot{\mathbf{x}})$ as 3D Cartesian position and velocity representations as well as the 4D quaternion-based orientation and angular velocity $(\mathbf{q}, \boldsymbol{\omega})$ to describe the base desired motion.

B. Residual RL Policies

SAC We used a fixed alpha term with $\alpha = 0.01$ and a recurrent neural policy. We used two feed-forward layers as embedding of the input that were fed in the recurrent policy with a cell state of 40. The actor policy was comprised of a 400 and 300 dimensional feed-forward layers with ReLU activation functions. The critic had a single feed-forward layer of size 300. Both policies had a single dense layer after the LSTM output of size 100 with a ReLU activation. We used truncated normal for weight initialisation. Those parameters were chosen in a grid search across policy network sizes (Figure 9).

We considered sizes of multiples of 2, where we scaled both actor and critic together. Smaller embedding sizes resulted in failure to converge while smaller cell sizes in faster but less stable convergence. Unlike the followed up on-policy approach, we were not able to use a simple feed forward policy due to the concurrent nature of the task at hand and the off-policy learning set up. The lower frequency of execution for the learned policy compared to the controller resulted in a gap of 50 control steps between an observed state at time t that led to selecting an action a_t and the actual state at $t + k \cdot t_\Delta$, for $k = 50$ that action was used in. We found sufficient to rely on a recurrent policy with a relatively small cell state for SAC and in the on-policy case, utilising the predicted value from the DMP deemed sufficient. We then minimise the loss using Adam and a batch of a single episode's length (100 steps) and learning rates for both actor and critic of 0.0003.

In the simulation-based scenario, we used a batch of size 32 and a single training step per iteration. We reduced episode requirements significantly by increasing the update steps to 32 which required uniformly sampling from the same replay buffer multiple times per update step. This resulted in 4 times fewer episodes required for a policy to converge. Figure 7b illustrates a simpler task where the robot position was initialised from a uniform distribution, centred at the initial position of the provided demonstration with a variance of $\pm 3cm$. Finally, we used a target update τ value of 0.05 and sampled sequence of 20 steps.

PPO We utilise the clipped objective introduced in Schulman et al. [35] which discourages large policy changes and focuses on improving the current solutions allowing for safer

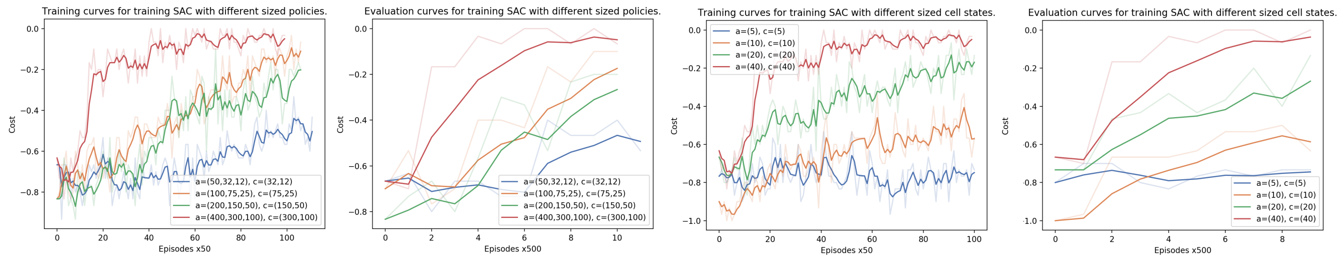


Fig. 9: Training and evaluation using different sized actor and critic policies. First two figures consider varying the size of the feed-forward policy, second two the cell size. The smaller the policy the less stable the resulted solution. The experiment was conducted on the 'easy' task with a sparse reward and a 3D input.

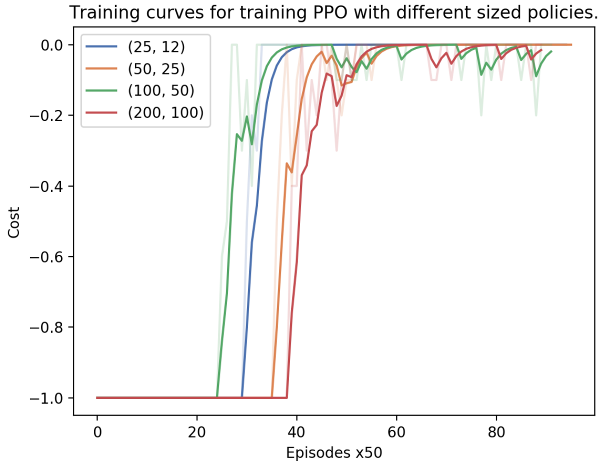


Fig. 10: Training PPO with varied actor and value policies sizes. The experiment used samples 6cm away from the demonstrated trajectory. Starting positions were sampled uniformly with variance $\pm np.sqrt(1/3) * \sigma$, for $\sigma = 0.001$.

(more constrained) exploration when using the hardware,

$$\mathcal{L}_{\theta_k}^{CLIP} = \mathbb{E}[\sum_{t=0}^{T} min(r_t(\theta)\hat{A}_t^{\pi_k}, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t^{\pi_k})]. \tag{9}$$

We define a replay buffer of a size equalling the length of a single episode and assume that the continuous actions are sampled from a learned Normal distribution. We define both the actor and value networks using standard shallow neural networks with 2 layers, Figure 10 reports on performance of the algorithm on a simple task using different sized policies. We project the outputs from the actor through a normal projection network with a $\tanh$ normalised output, thus ensuring that values near the spec bounds cannot be returned.

Further, we define a value for the bias of the independent from each other standard deviation projections as $log(\exp{(\eta)} - 1)$ for η being the initial action's standard deviation. We found using a curriculum over generalisation ranges was particularly useful in decreasing the training time for position-based corrections.

C. *Simulated Experiments*

Additional Implementation Details

In simulation we used a 0/1 sparse reward with $\kappa = 0.048$ where we used the L2 distance to assign 0 or 1 reward. We used $\alpha = 10$, $\beta = 0.002$, $\epsilon = 0.0001$ in the dense reward case. As state, we used the end-effector position relative to the goal. Our results suggested that this was sufficient for training on the 'easy' task only. Sampling for starting positions was done uniformly centered at the initial demonstration's starting position with variance of 11.8cm. We chose this value randomly from a pool of values between 10 and 12 as those led to tasks that were challenging enough for our base policy to solve. We used 40 basis functions to learn π_b and a single demonstration. We tried using an average over multiple demonstrations but found the resultant policies more likely end up hitting a surrounding wall (see Figure 7a).

D. *Additional Physical Experimental Details and Analysis*

We used a single demonstration to build the base policy which was sufficient enough to extract 100% successful insertions when an episode starts within 2/3 mm away from that starting position. Unlike the simulated experiments (Appendix VI-C) we used the full pose of the robot's end effector and the 3D calibrated sensed force, ($\mathbf{x}$, $\mathbf{q}$, $\mathbf{f}$) in all physical experiments. We represented $\mathbf{x}$ relative to the goal position and used 40 and 70 basis functions for $\hat{\pi}_b$ and $\tilde{\pi}_b$ respectively. We found that using only $\mathbf{x}$ as state space achieved similar performance to the linear policy with quadratic input (RLS from Table II).

Generalising to Residual Pose Corrections

In this section we provide additional evaluation and experimental setup details for the use of the concurrent-based implementation of residual LfD to the full pose. Using position based corrections can be useful but relies on unnatural assumptions such as moving on a 2D plane and perfect alignment. A full pose correction can result in more general solutions capable of addressing broader range of tasks and improving on existing approaches. For example, strategies like tilting the peg by some angle and then pushing or aligning the orientation of the held object in the exact direction cannot be achieved with position-based solutions.

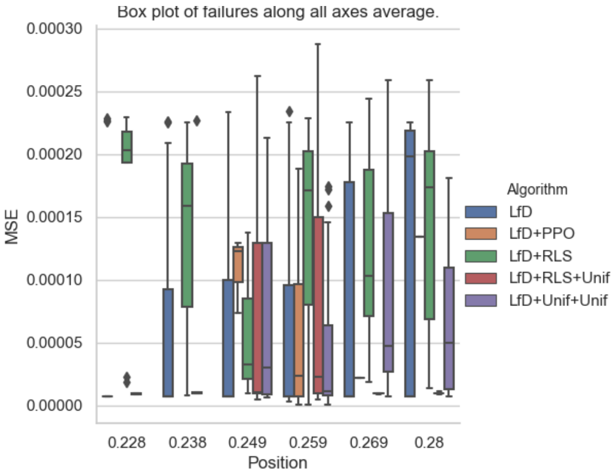


Fig. 11: MSE from the goal averaged across all axes. All considered policies fail with a similar mean squared error with and without using RL.

Peg Insertion The peg's width, depth and height are 28x28x77mm with a hole with 0.4mm clearance. We used a curriculum for training PPO where we started by ensuring the base policy can solve about 80% of the task itself, with the added noise from the untrained policy, this itself resulted to about 90-100% of successful insertions. After a few iterations we made the task harder and continuously kept increasing the complexity ensuring that the task would not get harder than being 80% solvable by the base policy and the currently existing trained one. We kept increasing the complexity by sampling initial critical points of up to 1.5 cm away from the initial demonstration. We trained for 500 episodes and used a sparse reward. This approach reduced the sample requirements drastically since at each stage we ensured a higher chance for the random exploration to land at a stage where the policy knows how to solve from.

Gear and Lan Insertion The gear is of size 79.2x79.85x10.79mm with a square hole of 23x23x10mm and Lan cable is standard RJ-45. Similarly to the previous subsection, we extract a base policy using position and orientation-based DMPs which we then learn to adapt onto. We trained π_θ using sampled unit vectors around which the starting position of the rigid body was rotated with 25 degrees only. Further, we used the corrective term after 3.9 seconds of executing the base policy and scaled its orientation based corrections by the frequency of the DMP while constraining the corrections within 6 degrees ($\approx 0.1 rad$). The starting position in x,y,z was sampled uniformly around the demonstrated trajectory and was within 3mm range. We trained the policy for 350 episodes, where each episode lasted at most 10 seconds.

Further Evaluation

Similar to Figure 8 we plot the performance measures across the X, Y and Z axes individually. Once more, linear and random policies were influenced by the diagonal slope of the surface. Note that the performance along the Z axis (i.e.

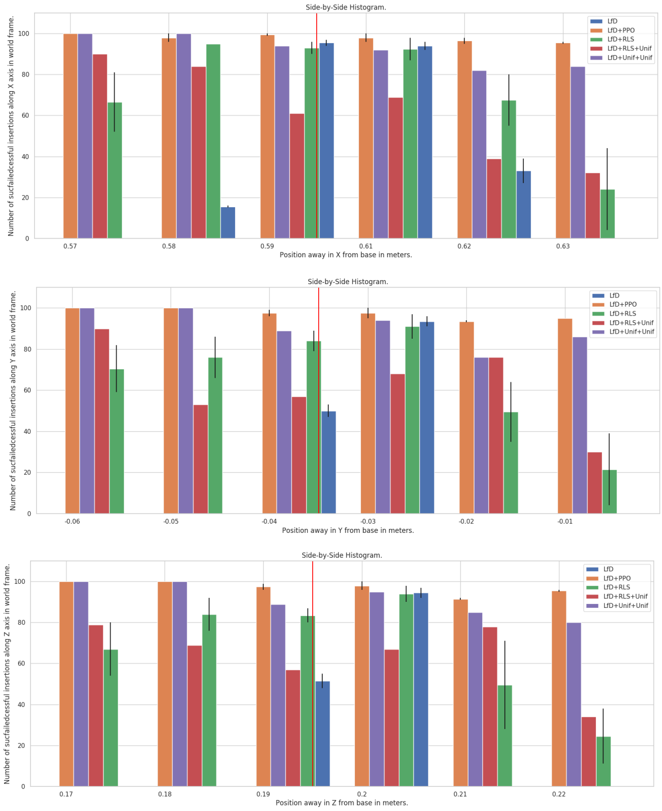


Fig. 12: Performance of the different considered policies on the peg insertion task along each axis.

3rd row) was not influenced by the slope of the surface the whole was position on. However, the distance away from the peg had influenced the simpler policies resulting in reduction of the overall accuracy. In addition to the overall achieved accuracy, we measured the mean squared error obtained in all failure cases along all three axes. Figure 11 shows that there is no difference between the linear and reinforcement learned policies in terms of mean squared error when it comes to failing at completing the task.

Finally, we plot the performance of the gear and Lan cable connector insertion tasks across different degrees away from the original orientation shown in Figure **??**. We evaluate against 8 different degrees away from the orientation of the provided human demonstration. Our results indicate that learning a residual policy capable of the full pose corrections can improve the average insertion accuracy across configurations outside the training scope of the considered policies.

E. More Baseline Implementation Details

We provide additional details around the considered baselines in this work. Our goal with the considered comparisons is to assess the capacity of the proposed ideas to adapt in the considered contact-rich insertion set ups. We used the publicly available implementations of eNAC, FD and PoWeR algorithms provided on the web page of the authors. We used an adapted version of tf-agents' implementation of SAC and PPO as described in Appendix VI-B.