Residual Feedback Learning for Contact-Rich Manipulation Tasks with Uncertainty

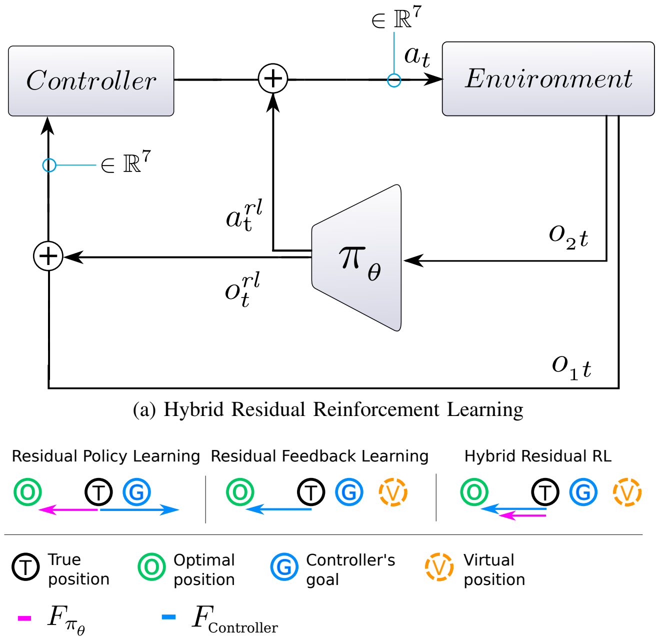
Alireza Ranjbar[1,2,3] Ngo Anh Vien[3] Hanna Ziesche[3] Joschka Boedecker[1] Gerhard Neumann[2]

***Abstract*— While classic control theory offers state of the art solutions in many problem scenarios, it is often desired to improve beyond the structure of such solutions and surpass their limitations. To this end, *residual policy learning (RPL)* offers a formulation to improve existing controllers with reinforcement learning (RL) by learning an additive ”residual” to the output of a given controller. However, the applicability of such an approach highly depends on the structure of the controller. Often, internal feedback signals of the controller limit an RL algorithm to adequately change the policy and, hence, learn the task. We propose a new formulation that addresses these limitations by also modifying the feedback signals to the controller with an RL policy and show superior performance of our approach on a contact-rich peg-insertion task under position and orientation uncertainty. In addition, we use a recent Cartesian impedance control architecture as the control framework which can be available to us as a black-box while assuming no knowledge about its input/output structure, and show the difficulties of standard RPL. Furthermore, we introduce an adaptive curriculum for the given task to gradually increase the task difficulty in terms of position and orientation uncertainty. A video showing the results can be found at `https://youtu.be/SAZm_Krze7U`.**

I. INTRODUCTION

Humans' skills for manipulating their environment has historically been foreseen as being overtaken by machines for which recent decades of research in artificial intelligence have promised more dexterity, adaptability, and cost efficiency with the help of accumulated experience or data. On the frontier, deep reinforcement learning (DRL) has proven its capability at learning similar skills through data and prior-knowledge, offering novel solutions while often surpassing humans' performance similar to other advances in machine learning (ML). However, as the research in this field continues, major challenges such as sample complexity and generalization capacity of the algorithms are yet addressed differently given each problem scenario and in many cases no solution is known to be optimal.

In many problem settings improving over available solutions appears more applicable than learning skills from scratch. Especially when samples are expensive, more guarantees are necessary, or the need for a solution is perceived more crucial than discovery while engineering costs are undesirable. Residual policy learning (RPL) [1], as one of possible solutions in this regard, suggests a formalism where the reinforcement learning (RL) agent learns to compensate for the imperfections of the up-stream controller by superposing its actions with it. This formulation provides a trade-off between capturing more information from the environment or allowing more exploitation of prior knowledge. Yet, the up-stream controller in many cases sees the intervention of the RL policy as external perturbation and error, and therefore tries to resist it. Fig. 1 (b) illustrates an example on the left where an ideal RL policy applies force toward the optimal direction while the controller has a different goal. In this case, one can instead modify the feedback to the controller and obtain different results. In this formulation which we denote as ”residual feedback learning” (RFL) the controller observes, for example, a *virtual* position, instead of the true feedback, and is therefore promoted in a different direction. In addition, in places where each formulation has its own advantage, combining both, i.e. an RL policy that outputs the residual controls as well as residual feedback commands, allows leveraging the distinct advantages of both methods simultaneously in one framework.



(a) Hybrid Residual Reinforcement Learning

(b) Expected behaviour of each Residual policy formulation

Fig. 1: (a) RPL combined with RFL. Notations $o_{1}t$, $o_{2}t$, a_t^{rl}, and a_t, correspond to the observations of the prior controller, and the RL policy, inferred actions from the RL policy, and final actions applied to the environment. (b) While RPL regards the intervention of the RL policy as external perturbation and error compared to its goal, it resists the intervention. In contrast, RFL changes the goal itself through feedback, that is for example, the controller sees a virtual position instead of the true position.

In general, for tasks that require wide spatial movements,

Affiliations with
[1] Albert-Ludwigs-Universität Freiburg
[2] Karlsruhe Institute of Technology
[3] Bosch Center for Artificial Intelligence (BCAI)

e.g, moving toward the hole in a peg insertion task, RPL appears limited as the up-stream controller observes the external intervention of an ideal RL policy as error from its goal and tries to recover from it. In contrast, RFL causes the controller to observe a virtual feedback (e.g., position) instead of the true feedback and hence does not result in a competition between the RL policy and the controller. On the other hand, for tasks that require sudden actions or high frequency vibrations such as releasing a stuck peg due to the orientation uncertainty of the hole, RFL does not appear suitable as the up-stream controller is often designed to only have smooth outputs while filtering feedbacks in different ways. In contrast, in this scenario RPL can apply such sudden actions. For this reason, we want to leverage the distinct advantage of each formulation at the same time. We refer to this approach as Hybrid Residual Reinforcement Learning (HRRL) and illustrate in our experiments the beneficial performance of such approach.

Furthermore, we build our residual reinforcement learning algorithm on the recently developed manipulation framework of [2] which provides an adaptive and compliant controller based on impedance control for several manipulation tasks. Using this baseline, we show the applicability of our formulation for industrial assembly environments represented by the common challenging peg-in-the-hole task. In addition, this approach allows having the option to choose where the RL policy should intervene for improvement as shown in Fig. 2. Finally, we evaluate and compare different variations of our residual policies for each sub-task of peg-in-the-hole, as well as the complete task in simulation. We considerably increase the task complexity by adding significant uncertainty in position and orientation of the hole, rendering it impossible for the standard controller to overcome. In order to cope with this challenging scenario, we apply adaptive curriculum learning to vary the task difficulty in terms of these uncertainties, leading to significant performance improvements.

Our primary contributions are as follows:

- We propose an alternative and an extension to the RPL formulation [1] to address its limitations for a wide range of tasks and controllers.
- We extend the manipulation framework of [2] using our approach and illustrate how the addition of residual feedback removes some limitations of standard residual policy learning.
- An empirical evaluation of the original method and ours along with their variants in simulation that we train within a recently proposed adaptive curriculum formalism [3].

II. RELATED WORK

a) Residual Reinforcement Learning: Two concurrent works [1] and [4] demonstrated the RPL formulation and highlighted advantages such as sample efficiency, better sim-to-real adaptation, as well as the ability in handling sensor noise and controller miscalibration. A follow-up work [5] developed this idea further using visual inputs and sparse rewards for industrial insertion tasks. Other work investigated improving the performance of RPL by exploiting the uncertainty of the policy architecture to decide when only the bare controller should be used [6] or taking advantage of using more than one controller [7].

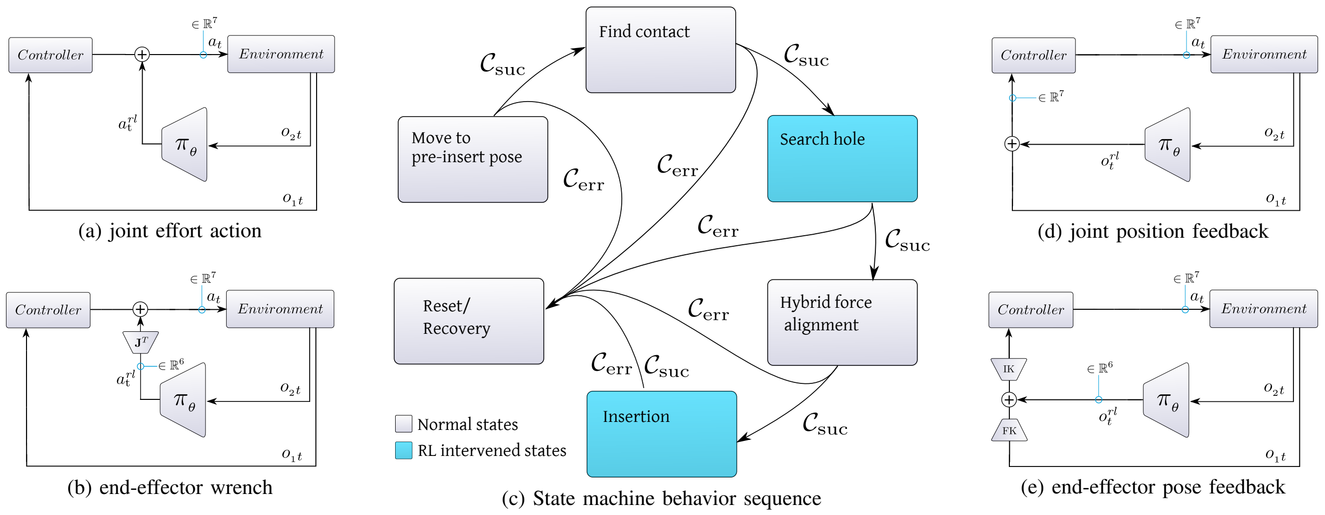
b) Contact Rich Manipulation and Assembly: Early works regarding peg-in-the-hole insertion had a rather theoretical view for analyzing contact models between the peg and the hole [8], [9]. A number of works focused on task specific engineering efforts or obtaining an accurate state estimation of contact through analytical or statistical methods [10]. On the other hand, some of the learning-based approaches include learning from demonstration (LfD) [11], model-free RL with proprioceptive and/or visual feedback [12], [13], [14], model-based RL [15], [16], and meta-RL [17]. A concurrent work [18] also leveraged residual-RL for an insertion problem. However, the authors mainly focused on analyzing the performance of a newly proposed graph-based structure for the experience replay buffer used commonly in off-policy RL methods. There has been recent effort proposing to take the advantages of the complementary nature of both haptic and visual inputs for industrial manipulation tasks [19], [20], [14], [21]. The formulations we discuss and propose in this work can certainly leverage the above ideas as well, as they remain agnostic to the choices of state representation, policy architecture, and the training algorithm.

c) Impedance Control: Variable impedance actuators (VIA) offer various natural characteristics of human motion such as safety, robustness, and energy efficiency while still possessing fast response time to impacts as well as energy efficiency [22], [23]. A recent work that applied VIA [23] proposed the first controller that can simultaneously adapt force and impedance within unknown dynamics to handle unstable conditions without requiring sensation of interaction forces. This work was then extended by Johannsmeier et. al. [2] to Cartesian space and full feed-forward tracking to also offer a structure for Cartesian impedance control that is applicable in a variety of tasks. Accordingly, the authors exploit the knowledge regarding constraints that come with every hardware such as stiffness adaptation speed. Furthermore they define a graph based manipulation skill formalism that can reduce the complexity of the solution space for robots' force-sensitive manipulation skills. We leverage these ideas in this work, while in contrast, we resort to a finite state machine controller.

III. PROBLEM STATEMENT

A. Partially Observable Markov Decision Process

We assume a controller is already available over which we aspire to improve using RL. Similar to most RL works in manipulation skills that involve uncertainty, we also formulate our problem as a discrete time and episodic Partially observable Markov decision process (POMDP) described by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{E}, \mathcal{R}, \gamma)$. These entries respectively correspond to the state-space, action-space, observation-space, transition probability $\mathcal{P}\left(s_{t+1} \mid s_t, a_t\right)$, emission probability $\mathcal{E}\left(o \mid s\right)$, reward function $r(s, a)$, and discount factor γ; where $s \in \mathcal{S}, a \in \mathcal{A}, o \in \mathcal{O}$. We also define $R\left(\tau\right) = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$ as our discounted return where $\tau = (s_t, a_t, \ldots, s_T, a_T)$. The objective is to optimize the parameters θ of a policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$ to maximize the



(a) joint effort action

(b) end-effector wrench

(c) State machine behavior sequence

(d) joint position feedback

(e) end-effector pose feedback

Fig. 2: (c): State machine behavior sequence of our insertion formalism. (a): The vanilla residual policy formulation from [1], [4]. (b): An alternative to allow inferring residual wrench values instead of residual joint torques. (d): The vanilla residual feedback formulation in contrast to (b). (e): Similar to (b) it is an alternative allowing to infer residual feedback in task space instead of joint space, e.g., the end effector pose. We elaborate more details regarding each in section IV.

expected return R, that is $\underset{\theta}{\operatorname{argmax}}\ \mathbb{E}_{\tau\sim p_\theta(\tau)}[R(\tau)]$ where p_θ is the trajectory distribution induced from the stochasticity of transitions, observations, and policy. In the following section we elaborate on how we integrate the RL policy π_θ along with the controller to improve beyond its structure efficiently.

B. Finite State Controller for Manipulation

Following the skill formalism described in [2], we implemented a state machine for our peg-in-hole task, shown in Fig. 2(c) and include it within our black-box controller, over which we seek improvement. This state machine controller includes five states each of which evaluates predefined success conditions $\mathcal{C}_{\text{suc}}$ at run time to allow proceeding with next states or subtasks. Moreover, we use an additional state to proceed with a recovery behaviour if any of the states are not successful, i.e. evaluate to $\mathcal{C}_{\text{err}}$, or the insertion sequence has finished. This state in the context of RL serves as a reset behavior for re-initializing the environment again at test or train time. As shown in Fig. 2(c), we have the advantage of optionally choosing which state of the state machine should improve and suggest intervention of the RL policy only at those states, highlighted in blue. The inputs to the controller include joint position, velocity, and effort at 1KHz along with the outputs of target joint torques at the same frequency. In addition, the state machine updates set-point commands of the controller at 40Hz similar to the frequency at which residual commands are inferred from π_θ. For each episode, this result in *environment steps* of approx. 6000 while the *learning steps* at which we use the RL policy (episode length) was approx. 150 which is also because we use the policy during only two out of five states of the state machine. As illustrated in Fig. 2(c) it includes five primary states starting with "*Move to pre-insert pose*" that brings the peg in a tilted orientation above a pre-defined hole position. Next, in "*Find contact*", the peg is moved toward a pre-defined direction until it touches the surface after which the controller proceeds with the "*search hole*" state that moves the peg on the surface while maintaining a constant vertical force on it to find the hole. Afterwards, the peg's tip is assumed to be between the hole's edges where in "*Hybrid force alignment*" the robot aligns the peg with a pre-defined orientation vertical to the surface while applying a constant force on the surface. Finally, in the "*Insertion*" state, the controller, applies a vertical force to insert the peg while applying sinusoidal oscillations over the applied wrench at the end-effector frame.

IV. Hybrid Residual Feedback & Actions

We first define the configuration space of our robot as $C \in \mathbb{R}^7$ in addition to its joint's angular velocity $\mathcal{V} \in \mathbb{R}^7$, joint's torque space $\mathcal{T} \in \mathbb{R}^7$, end-effector pose $\mathcal{P} \in \mathbb{R}^7$ (Cartesian position and quaternion), and wrench space $\mathcal{W} \in \mathbb{R}^6$. For each modality we use superscripts "o", "u", and "π" if they represent the input to the controller, its output, and the output from sampling the policy distribution π_θ. Accordingly, we assume f is a a conventional impedance feedback controller with a mapping $f : \mathcal{C}^{\text{o}} \times \mathcal{V}^{\text{o}} \times \mathcal{T}^{\text{o}} \mapsto \mathcal{T}^{\text{u}}$. Furthermore, we distinguish between the observations of the controller f and the RL policy by $o_1 \in \mathcal{C}^{\text{o}} \times \mathcal{V}^{\text{o}} \times \mathcal{T}^{\text{o}}$ and $o_2 \in \mathcal{O}$ respectively. Nevertheless, the methods discussed in this section are yet agnostic to the choices of f, observation, and action spaces. In the following we define each sketched formulation in Fig. 2, and Fig. 1(b).

A. Residual Policy Learning (RPL)

The original formulation of RPL proposes to use a residual policy whose actions are added to the outputs of the controller. However, depending on the controller's architecture, the output of the controller can be adapted at different levels of the control command, e.g., the joint effort or the end-effector wrench. Both approaches are described below.

*1) **joint effort action**:* The original RPL formulation, $\pi_\theta : \mathcal{O} \times \mathcal{T}^\pi \mapsto [0,1]$, proposed by [1] suggests superposing outputs of a RL policy with those of the up-stream controller f, that is

$$a_t^{rl} \sim \pi_\theta\left(\cdot \mid o_{2t}\right), \quad a_t = f(o_1, t) + a_t^{rl}. \tag{1}$$

Here we define $\mathcal{T}^\pi$ as the action space of the policy π_θ which is of the same modality as the controller's outputs, in this case joint efforts. We refer to this formulation in our comparison as ”joint effort action” and illustrate a schematic of it in Fig. 2(a).

*2) **end-effector wrench**:* As mentioned earlier, although we can assume no knowledge about the input/output structure of the controller, in cases where such knowledge is available, we can certainly exploit them to our problem's advantage. In our case for RPL we i) first map the upstream controller's output to the wrench space, ii) then superpose them with their equivalent output from the π_θ, and finally iii) map the result back to the original joint space. This procedure can also be done by only mapping residual wrench values to joint space, yet with less possibility for post-processing of the controller's output in wrench space. In our case, we do this conversion by $\mathcal{F} = \mathbf{J}^{\dagger^T} \tau$ and $\tau = \mathbf{J}^T \mathcal{F}$ where $\tau \in T$, $\mathcal{F} \in W$, $\mathbf{J}$, and $\mathbf{J}^\dagger$ are joint torque commands, wrench at end-effector frame, Jacobian, and damped-pseudo inverse of the Jacobian respectively. We note this formulation as ”end-effector wrench”, $\pi_\theta : \mathcal{O} \times \mathcal{W}^\pi \mapsto [0,1]$, with a superposed action computed as

$$w_t \sim \pi_\theta\left(\cdot \mid o_{2t}\right), \quad a_t = f(o_1, t) + \mathbf{J}^T w_t. \tag{2}$$

This formulation allows using a control space that is more relevant for the task, i.e., the end-effector space, and therefore, intuitively, it should be easier to solve for the residual policy depending on the task. This policy design is sketched in Fig. 2(b).

B. Residual Feedback Learning (RFL)

There is a wide range of tasks where the above formulation does not perform well as the residual policy causes a feedback distribution shift that the controller sees as external perturbation which it tries to resist. Hence, the residual actions from the RL policy results in a competition between f and π_θ. That is, as shown for example in Fig. 1 (b), if the π_θ optimal action moves the end-effector to the left, the prior-controller fights this perturbation and generates forces to retrieve the previous position. Furthermore, depending on the controller's structure, the response to such perceived external perturbation or persistent error can vary extensively. In some cases it may even lead to lower safety, especially where the residual-policy architecture lacks bounded output guarantees. To address such limitations, we propose learning a *residual feedback* as an alternative to residual-action policies. Here, instead of superposing residual actions to the output of the controller, we superpose residual feedback to the feedback it receives from the environment.

*1) **joint position feedback**:* We start with a vanilla formulation where residual feedbacks are used in the original feedback space of the robot, in our case the joint positions. Our residual feedback policy can be defined as $\pi_\theta : \mathcal{O} \times \mathcal{C}^\pi \mapsto [0,1]$. The superposed action is then computed as follows

$$o_{1t}^{rl} \sim \pi_\theta\left(\cdot \mid o_{2t}\right), \quad a_t = f(o_1 + o_{1t}^{rl}, t). \tag{3}$$

For residual feedback however, we only superpose residual joint position feedback $q \in C$ [1], while optionally other feedback modalities can be included, e.g.. joint's angular velocities and torques, depending on the application. This policy design is sketched in Fig. 2(d), where we only show an example of a superposition of residual joint position feedback.

*2) **end-effector pose feedback**:* We can again use a task-space centric feedback signal by converting the feedback from the environment to the task space and, after superposition with the output of the RL-agent, map them back to their original space. For instance, we do so in our case we map joint positions to end-effector position in the base frame using forward kinematics (FK). Afterwards we map the superposition results back using inverse-kinematics (IK). We denote this residual feedback approach ”end-effector pose feedback”. In addition, this conversion can be done rapidly as the agent's outputs only make a small modification of the feedback and the optimizer that we use for inverse kinematics can use the true feedback as an initial optimization point. This procedure results in a definition of our policy $\pi_\theta : \mathcal{O} \times \mathcal{P}^\pi \mapsto [0,1]$, where $\mathcal{P}^\pi$ is the action space (i.e. residual end-effector pose), and the superposed action is computed as

$$\begin{aligned} &ee = \mathrm{FK}(q_t), \quad a_t^{rl} \sim \pi_\theta(\cdot|o_{2t}), \\ &q_{\text{residual}} = \mathrm{IK}(ee + a_t^{rl}), a_t = f(q_{\text{residual}}, v_t, t_t). \end{aligned} \tag{4}$$

where $ee \in \mathcal{P}^o$ and we assume the normal input to f is $o_{1t} = (q_t, v_t, t_t)$ (before superposing residual feedback). This policy design is sketched in Fig. 2(e). The formulation again allows adapting the feedback in a space that is more relevant for the task.

C. Hybrid Residual Reinforcement Learning (HRRL)

Each formulation of residual reinforcement learning and feedback learning has advantages depending on the stage of the task execution (cf. our experiments in section V-C). This allows, for example, sudden actions or high frequency vibrations that are needed for releasing a peg that is stuck due to overcome orientation uncertainty with residual actions as well as more flexible spatial movement of the peg in search for the hole with residual feedback. For this reason we propose a combination of residual action and residual feedback to form a new residual hybrid model, called ”joint space hybrid”. We illustrate a schematic of our hybrid model in Fig. 1(a). In our case, we extend the action space to 14 dimensions where the first 7 dimensions contribute within the original residual policy formulation and the remaining dimensions modify the feedback. In particular, our hybrid residual policy is defined as $\pi_\theta : \mathcal{O} \times \mathcal{T}^\pi \times \mathcal{C}^\pi \mapsto [0,1]$, with a superposition,

$$a_t^{rl}, o_{1t}^{rl} \sim \pi_\theta\left(\cdot \mid o_{2t}\right), \quad a_t = f(o_{1t} + o_{1t}^{rl}, t) + a_t^{rl}. \tag{5}$$

[1] where we implicitly assume the sum $o_{1t} + o_{1t}^{rl}$ being a superposition of o_{1t}^{rl} and corresponding joints' dimensions of o_{1t}.

Note that $o_{1t}^{rl} \in \mathcal{C}^{\pi}$ only represents residual feedback of the joint modality. For simplicity we implicitly assume the sum $o_{1t} + o_{1t}^{rl}$ is a superposition of o_{1t}^{rl} and corresponding the joints' dimensions of o_{1t}.

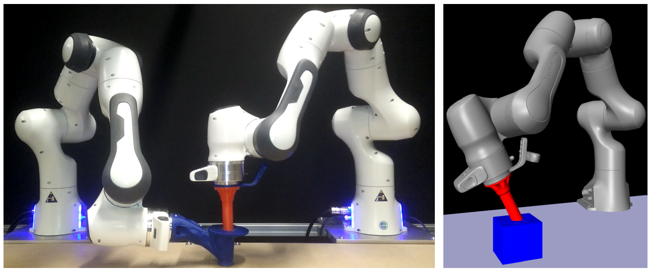
V. Experiments

In the following we evaluate the above formulations experimentally to observe their advantages and disadvantages within different problem settings. This includes comparing them in terms of final-performance, sample efficiency, as well as well as an analysis of the benefits of each method over the others. While denoting our experimental settings, we start with simulations first and illustrate our result in the real environment in the end.

A. *Workspace*

We base our experiments on a peg-in-the-hole task where a shaft of 70mm length and 25mm diameter is used as the peg. We use the Franka Emika robot arm with seven degrees of freedom (DOF) to fully insert this peg in a hole of 25.8mm diameter. Two robot arms were used — one for peg insertion while the second arm rearranged the hole between each episode for simulating the pose and orientation uncertainty. To leverage quick experiments while focusing on the difficulty of contact rich insertion rather than grasping, we fixated the shaft and hole designs to the robots' arm as shown in Fig. 3. Compared to some of the earlier peg insertion works, e.g. [9], [14], [17], [18] our task is more difficult due to the larger size of the peg in terms of diameter, length, and raw surface roughness owing to the 3D-printer's outlet. The end-effector commands and state readings are computed with respect to the tool central point (TCP) at the tip of the peg. For our RL learner we use the PyTorch proximal policy optimization (PPO) implementation from [24].

We introduce uncertainty to our task in terms of the position and orientation of the hole in meters and radians respectively, i.e., before each episode, the hole position and orientation are sampled from a Gaussian distribution and the second arm rearranges the hole accordingly. The variance of this Gaussian distribution directly relates to the task difficulty and is specified by the curriculum. The position of the hole is not known to the second robot (the learning agent). In addition, to allow evaluation and debugging in simulations for orientation uncertainty separate from that of the position, the orientations are calculated with respect to a central point on the upper side of the hole to allow the peg's tip to always fall down between the hole edges. In all training experiments we choose the sparse reward of $r(s, a)$ which is 1 if $\left\|P_{tcp} - P^{*}_{tcp}\right\|_2 < \epsilon$, and 0 otherwise , where P_{tcp} and P^{*}_{tcp} are the current and goal TCP positions. The value of P^{*}_{tcp} is computed based on the lower side of the hole's position and ϵ was set to 5mm. We initialize the last layer of the policies with zero weights such that we start with a plain execution of the prior-controller. We use the first 50 episodes to only learn the critic without updating the policy. The starting points of all curves are regardless of the chosen algorithm.


(a) Real scene

(b) Simulation scene

Fig. 3: Franka Emika robots at train time for peg-in-the-hole task. While one robot is responsible for insertion of the peg, the other robot serves as a mechanism to introduce uncertainties of the environment.

We use MuJoCo [25] for simulation of our peg-in-the-hole assembly task. Only one robot is simulated as the hole can be rearranged programmatically. Since the physics engines require all simulated objects to be convex, we automated the design of CAD models to fulfill this requirement using a python script in Blender [26] that decomposes separate mesh parts for MuJoCo. A simulation scene can be found in Fig. 3.

B. *Searching for Hole using Visual Inputs*

Although our focus regarding industrial assembly is mainly on using contact feedback, we find using visual inputs while moving towards the hole in the air better to contrast the advantage of using residual feedback compared to residual actions. We do so during the ”Move to pre-insert pose” state of the state machine, shown in Fig. 2 (c) and consider 1.6 centimeters position uncertainty for the hole without any orientation uncertainty. For our observations $\mathcal{O}$ we choose 84x84x3 RGB images from a hand-mounted camera, as well as the robot's end-effector's Cartesian position and orientation in Euler angles along with its wrench. In addition, we choose a convolutional policy architecture similar to [27]. These convolutional layers that receive RGB inputs comprise of 32 filters of 8x8 size with stride 4 followed by 64 filters of size 4x4 with stride 2, and 32 filters of size 3x3 with single stride, all of which use ReLU activation. We concatenate the output of the convolutional layers with a latent representation of the robot's state similar to [3] before the subsequent actor's and critic's fully connected layers.

Here, with the original formulation of RPL (e.g., ”joint effort action” in our case), ideal residual actions try to move the peg above the hole while the upstream controller observes this interaction as external perturbation and resists it. This effect becomes more damaging when the frequency of the controller is higher than the frequency of the RL-agent interventions. That is, if we update each residual action after every 10 controller steps, those 10 steps represent opportunities to counteract the external perturbation and hence compete with the RL-policy. One way to investigate such an issue is using different RL-agent frequencies. However, this would also change the episode length and the learning problem. For this reason, we only add a number of controller ”buffer steps” during which the controller works without any update from the RL policy, giving the aforementioned opportunity to recover from any external perturbations, such as those from the RL policy. We add these buffer steps at the end of the

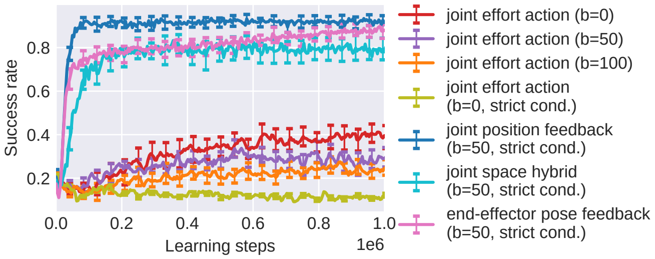


Fig. 4: Performance of using residual feedback learning compared to the original residual policy learning. The environment difficulty includes only 1.6 (cm) position uncertainty. Error bars correspond to half of the standard deviation over 30 seeds.

"Move to pre-insert pose" each of which correspond to 1 millisecond of using the controller without any RL policy inference. Furthermore, regardless of the sensitivity of the low-level control layers to their feedback (e.g., impedance controller in our case), higher level layers such as state machines or a behavior trees can also respond critically to the feedback. We show an example of such case that we refer to it as "strict condition" in our comparison in Fig. 4, where the state machine raises an error if it does not observe achieving its goal that can be the result of any external perturbation such as those coming from the residual action policy. As illustrated in this figure, higher buffer steps, e.g., b=100, gives the up-stream controller more opportunity to resist the residual policy, resulting in lower success rates. In contrast, using residual feedback does not result in any competition between the RL policy and the low level controller, while also keeping the higher level state machine satisfied in achieving its goal. We observe these results from the **joint position feedback** baseline in Fig. 4.

C. *Using contact and proprioceptive Features as Inputs*

In this experiment, the peg-in-the-hole task needs to be learned without vision feedback, only relying on the proprioceptive feedback and the contact force readings at the joints. We choose the relative position of the end-effector from the position where the first contact takes place denoted as P_{tcp}, along with its orientation in euler angels, θ, and the measured contact wrench $\mathcal{F}$ at the end-effector frame. For the policy architecture, we use long short term memory (LSTM) similar to [28], which is shared between the actor and the critic as shown in Fig. 5. The actor computes the mean of a d_a dimensional Gaussian distribution from which actions are sampled, (see Section IV for a description of the action definitions). Here, achieving a successful insertion includes solving two main sub-tasks that are i) search for the hole by moving the peg's tip on the surface, and ii) insertion, during which orientation uncertainty is of challenge. We aim for improving the prior controller in obtaining this goal by allowing the RL policy to intervene during relevant sub-tasks as shown in Fig. 2(c). In addition, for training our policy we leverage the adaptive curriculum formulation from [3] with the difference that our environments adapt the difficulty independent of each other. This curriculum increases or decreases the degree of domain randomization (variance in the hole position and orientation) depending on the current success rate of the policy which helps us significantly on sparse-reward domains like our current setting. That is, when the current success rate is below 0.6, we decrease the difficulty, and increase if the success rate is above 0.7. We list the parameters of such adaptive domain randomization as well as the uncertainties involved in our evaluation environments in Table I. Moreover, Fig. 7 shows the history of this adaptation in our experiments and Fig. 6 illustrates the results of our evaluations concurrent with training. These evaluations were done for position and orientation uncertainty separately as well as together. Additionally, we include a comparison with learning from scratch where the controller is only used for finding contact and alignment of the peg while the rest is purely controlled by an RL agent using joint torques as the action space. In addition, we also experiment the "joint space hybrid" formulation that had shown superior performance but without the adaptive curriculum. This baseline that we denote "joint space hybrid (without curriculum)" in Fig. 6 trains the environments with the same difficulty at which the other baselines are evaluated, and due to this high difficulty, it rarely observe any reward to learn.

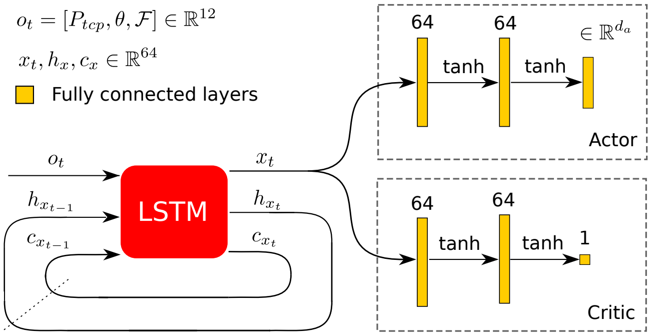


Fig. 5: The network architecture used in Section V-C. We initialized our policy architecture with (semi) orthogonal matrices of gain $\sqrt{2}$, except the last layer of the actor which similar to [4] is zero initialized.

As the results shown in Fig. 6 suggest, **joint position feedback** appears superior for handling position uncertainty, **joint effort action** for orientation uncertainty, while mitigating both uncertainty types could be achieved best with **joint space hybrid**. We see the same observations from Fig. 7 where **joint space hybrid** can adapt to the widest range of uncertainties, i.e. always maintain a high success rate on the largest standard deviations of domain parameters. This makes sense as overcoming position uncertainty requires

Adaptive domain randomization parameters		Experiment		
		Only Position	Only Orienration	Both Unc. types
Position std. (Meters)	Initial	0.007	0	0.007
	Evaluation	0.016	0	0.015
	Increment	0.001	0	0.001
Orientation std. (Radians)	Initial	0	0.05	0.05
	Evaluation	0	0.15	0.1
	Increment	0	0.01	0.01
success rate bounds		[0.6, 0.7]	[0.6, 0.7]	[0.6, 0.7]

TABLE I: Adaptive curriculum parameters used in each experiment that we describe in section V-B. The evaluation values correspond the constant difficulty at which the policies were evaluated in Fig. 6 concurrent with training.

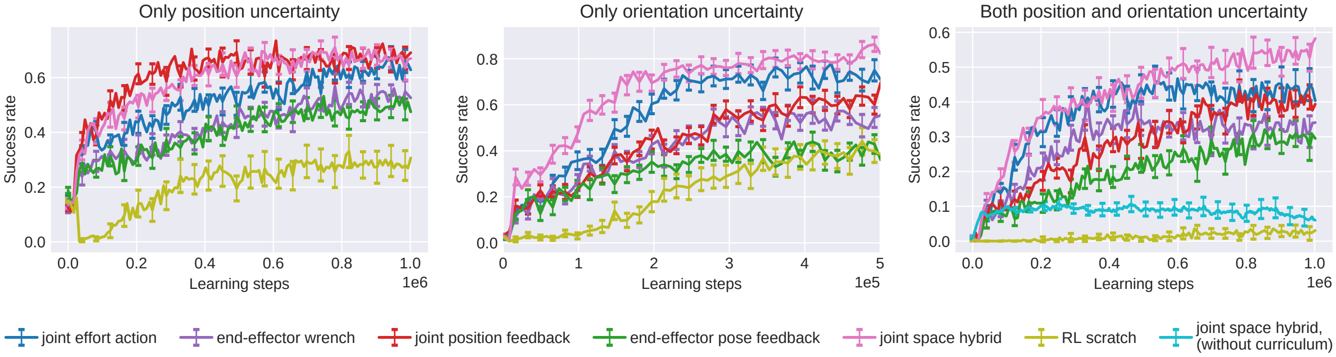


Fig. 6: Evaluation of each residual policy variant in simulation averaged over 40 seeds using 8 parallel environments four of which are for training (with variable difficulty) and the rest for evaluation (with constant difficulty). The constant difficulty of these evaluations are noted in Table I. The starting point of each curve correspond the zero-shot evaluation of controller for 50 episodes. Error bars correspond to a quarter of the standard deviation.

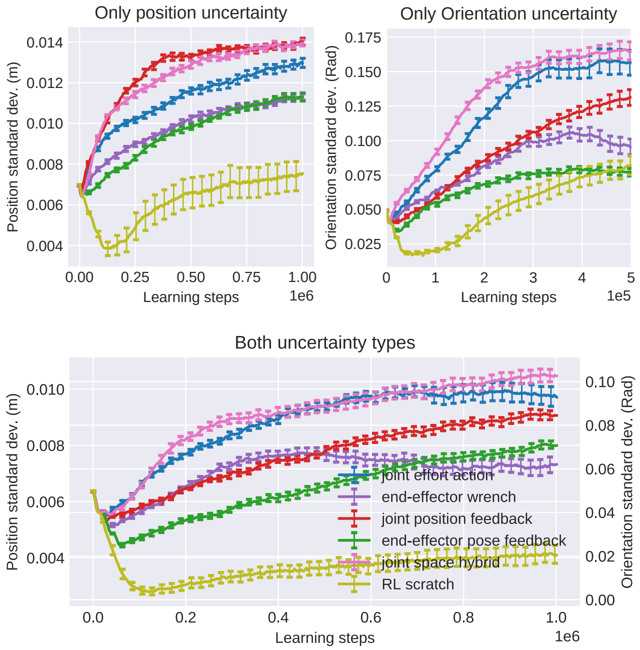


Fig. 7: History of the domain randomization parameters, i.e., the curriculum difficulty at train-time. Error bars correspond to a quarter of the standard deviation.

more spatial movement of the peg which is difficult with ”joint effort action” (i.e., RPL) due to the resistance of the controller. In contrast, for orientation uncertainty, behaviors such as a rather high frequency vibration of the peg in the hole help pushing the peg into the hole. Yet, this may not be possible with residual feedback learning (RFL) learning as the controller is often designed to produce smooth output actions. In this case, the RPL can, for example, conduct non smooth or sudden actions through the residual outputs that are directly applied to the environment. From our results, it can be seen that our hybrid residual RL framework combines the strength of both approaches and shows superior performance in the single pose or orientation uncertainty tasks as well as the complete task.

D. *Hardware experiments*

We trained our policy on hardware with a speed of approx. 3 episodes per minute with the same observation space we described in section V-C and only using our ”joint space hybrid” formulation. As we also leverage adaptive domain randomization here, the evaluation of the policy concurrent with training similar to the simulations would have doubled our experiment time. For this we use our real environment only for training and evaluate it afterwards. We illustrate the history of the adaptation in Fig. 8 as well as success rate at train time. Every time the environment’s difficulty increases with respect to the curriculum the observed success rate decreases as well that is the reason for the oscillatory shape of the learning curve. Our training starts with position and orientation uncertainty of 5 millimeters and 0.015 Radian. In addition, our adaptive curriculum only modifies the orientation uncertainty using the second robot as shown in Fig. 3(a), by an increment of 0.0025 Radian depending on the success rate over 15 episodes. Using this adaptive formulation offered us the convenience of not having to predefine the difficulty of the environment without knowing if it is too high or low while automatically increasing the degree of domain randomization on hardware. One single experiment in total took 12 hours. Finally, our evaluations demonstrated the success rate of 0.92 within 25 trials orientation and uncertainty of 0.08 Radians. A video showing the results can be found at `https://youtu.be/SAZm_Krze7U`.

VI. CONCLUSION AND FUTURE WORK

Every controller comes with numerous imperfections that limit its performance. Hence, it is desirable to improve beyond their underlying structures using least engineering effort and in a sample efficient way. We demonstrated the limitations of the existing RPL formulation for controller improvement that uses residual actions and proposed a more flexible extension that can also exploit residual feedback. As we demonstrated, both residual formulations have distinct advantages that can be exploited in different task settings. We demonstrated leveraging both residual models simultaneously within a hybrid model and gained superior performance over all cases. In contrast to the RPL formulation where the

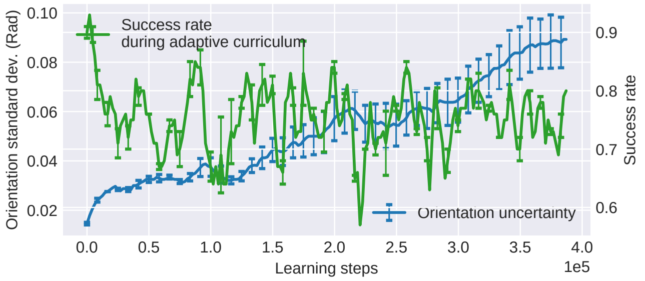


Fig. 8: Adapted orientation uncertainty based on curriculum as well as success rate of the training environment on hardware over two different seeds. Error bars include half of standard deviation around the mean.

up-stream controller resists the external perturbation caused by the RL policy, the RFL applies changes in the feedback signal, which can be regarded as similar to changing the setpoint of the controller in many cases.

Furthermore, we chose industrial assembly as a good example of a scenario where lack of accurate models to represent contact rich dynamics is of great challenge especially if uncertainties need to be considered. While we demonstrate sample efficient improvement of an assembly task example, i.e., peg-in-hole using raw sensory inputs, using learned latent representations of those inputs should promise even more efficiency and generalization within POMDP settings. This applies also to different choices of RL algorithms that are more sample efficient than PPO as our formulations are agnostic such a choice.

Finally, sim-to-real transfer of policies that only observe contact inputs is a major challenge that we kept for future work. In addition, one can also investigate better exploration strategies as well as more guarantees regarding residual policy formulations. As we took the advantage of having another robot side by side with the main robot responsible for our task, an interesting future extension is to have two robots competing with each other in an adversarial setting where one robot tries to learn successful insertions while the other robot learns to make insertions more difficult.

References

[1] 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,” 2018.

[2] L. Johannsmeier, M. Gerchow, and S. Haddadin, “A framework for robot manipulation: Skill formalism, meta learning and adaptive control,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5844–5850.

[3] L. Hermann, M. Argus, A. Eitel, A. Amiranashvili, W. Burgard, and T. Brox, “Adaptive curriculum generation from demonstrations for sim-to-real visuomotor control,” 2019.

[4] T. Silver, K. Allen, J. Tenenbaum, and L. Kaelbling, “Residual policy learning,” 2018.

[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,” 2019.

[6] K. Rana, B. Talbot, M. Milford, and N. Sünderhauf, “Residual reactive navigation: Combining classical and learned navigation strategies for deployment in unknown environments,” 2019.

[7] M. Barekatain, R. Yonetani, and M. Hamaya, “Multipolar: Multi-source policy aggregation for transfer reinforcement learning between diverse environmental dynamics,” 2019.

[8] H. Qiao, B. S. Dalay, and R. M. Parkin, “Robotic peg-hole insertion operations using a six-component force sensor,” *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, vol. 207, no. 5, pp. 289–306, 1993.

[9] J. Ding, C. Wang, and C. Lu, “Transferable force-torque dynamics model for peg-in-hole task,” 2019.

[10] T. Lefebvre, H. Bruyninckx, and Joris De Schutter, “Online statistical model recognition and state estimation for autonomous compliant motion,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 1, pp. 16–29, 2005.

[11] A. Wan, J. Xu, H. Chen, S. Zhang, and K. Chen, “Optimal path planning and control of assembly robots for hard-measuring easy-deformation assemblies,” *IEEE/ASME Transactions on Mechatronics*, vol. 22, no. 4, pp. 1600–1609, 2017.

[12] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[13] M. Kalakrishnan, L. Righetti, P. Pastor, and S. Schaal, “Learning force control policies for compliant manipulation,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 4639–4644.

[14] M. A. Lee, Y. Zhu, P. Zachares, M. Tan, K. Srinivasan, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, “Making sense of vision and touch: Learning multimodal representations for contact-rich tasks,” 2019.

[15] S. Levine, N. Wagener, and P. Abbeel, “Learning contact-rich manipulation skills with guided policy search,” 2015.

[16] J. Fu, S. Levine, and P. Abbeel, “One-shot learning of manipulation skills with online dynamics adaptation and neural network priors,” *CoRR*, vol. abs/1509.06841, 2015. [Online]. Available: http://arxiv.org/abs/1509.06841

[17] G. Schoettler, A. Nair, J. A. Ojea, S. Levine, and E. Solowjow, “Meta-reinforcement learning for robotic industrial insertion tasks,” 2020.

[18] 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)*, 2020, pp. 9080–9087.

[19] H. Hoof, N. Chen, M. Karl, P. van der Smagt, and J. Peters, “Stable reinforcement learning with autoencoders for tactile and visual data,” 10 2016.

[20] 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.

[21] R. L. Haugaard, J. Langaa, C. Sloth, and A. G. Buch, “Fast robust peg-in-hole insertion with continuous visual servoing,” *arXiv preprint arXiv:2011.06399*, 2020, accepted to CORL 2020.

[22] A. Albu-Schäffer, S. Haddadin, C. Ott, A. Stemmer, T. Wimböck, and G. Hirzinger, “The dlr lightweight robot: design and control concepts for robots in human environments,” *Ind. Robot*, vol. 34, pp. 376–385, 2007.

[23] C. Yang, G. Ganesh, S. Haddadin, S. Parusel, A. Albu-Schaeffer, and E. Burdet, “Human-like adaptation of force and impedance in stable and unstable interactions,” *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 918–930, Oct 2011.

[24] I. Kostrikov, “Pytorch implementations of reinforcement learning algorithms,” https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[25] 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*, 2012, pp. 5026–5033.

[26] Blender Online Community, “Blender - a 3d modelling and rendering package,” Blender Foundation, Blender Institute, Amsterdam, http://www.blender.org

[27] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, “Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation,” 2017.

[28] T. Inoue, G. D. Magistris, A. Munawar, T. Yokoya, and R. Tachibana, “Deep reinforcement learning for high precision assembly tasks,” *CoRR*, vol. abs/1708.04033, 2017. [Online]. Available: http://arxiv.org/abs/1708.04033