FORGE: Supplementary Material

I. Related Work

A. Real World Reinforcement Learning

A large body of work focuses on learning assembly tasks directly on the real-robot. Learning directly on the robot side-steps the *sim-to-real* gap by using data (and contact-interactions) from the same distribution expected at deployment. These works typically address problem of data efficiency by leveraging demonstrations [1], [2], [3], [4], [5] or using model-based approaches [6], [7], [8], [9]. To ensure excessive forces are not exceeded during training, these papers typically use control methods designed to be safe [5], [10], [11].

B. Sim-to-Real Methods

Learning directly in simulation is often preferable for robot safety, increased task variability, and access to privileged state. With advancements in RL and parallelizable simulation [12], [13], [14], [15], there has been much interest in *sim-to-real* transfer for complex control problems. Of note include legged locomotion [16], [17], [18], [19] and in-hand manipulation [20], [21], [22].

Recent advances in contact-rich simulation have enabled efficient simulation of assembly tasks [23], [24], [25], [26]. However, as discussed throughout the paper, the key challenge becomes the sim-to-real gap: how can we *safely* and *successfully* deploy policies that were trained in simulation?

Although *system identification* is a principled approach to minimize the *sim-to-real* gap [27], it is often time-consuming and difficult to apply to contact-rich tasks [28], [29]. Instead, many approaches use *dynamics randomization*: randomizing parameters such as part friction/stiffness [30], [31], [32], [33], [34], controller gains [35], [33], or F/T observation scale [35], [36]. A third class of approaches uses a small amount of real-robot data to finetune the policies learned in simulation [31], [37], [38].

Even with dynamics randomization, excessive forces can occur when deployed on the real-robot. In some work, parts are fixed to the gripper [32] and slip will not occur. When this is not the case, several methods have been proposed to ensure safe policy deployment. An expert can tune the controller gains at deployment time or design the action space in a way such that all actions are safe [39], [40], [41]. Gains can also be adapted online via optimization [42] or an explicit *gain-tuning* model [36].

Similar to FORGE, other works have proposed to use a force-threshold [31], [43], [32]. These works have a fixed threshold during training which is often very large to primarily prevent damage (e.g., $40N$). However, especially with small parts, slip can occur with much lower contact forces. Most similar to FORGE, [31] introduces a method to specify the *desired* interaction force at deployment time.

Most prior work focus on insertion-style tasks. We show how the combined application of a force-threshold and dynamics randomization can lead to robust sim-to-real transfer for a range of tasks, including the complicated nut-threading task. Prior work on sim-to-real for nut-threading [44] focused on large parts ($M48$ nuts) that were fixed to the gripper. In addition, we show these techniques are applicable for sim-to-real transfer of success prediction procedures.

II. Randomization

All randomization ranges are reported in Table I. In addition to the dynamics randomization described in the text, we also randomize the initial state distribution and observation noise.

Initial State Randomization: At the start of an episode, we randomize the position of the fixed part, the relative pose of the hand above the fixed part, and the relative position of the held part in the gripper (where the default position has the top of the held part aligned with the bottom of the gripper).

Observation Randomization: In simulation, the position of the fixed asset is randomized once per episode by adding Gaussian noise. Independent Gaussian noise is added to each observation at every timestep (except velocity, where positional noise is propagated through finite differencing).

III. Reward

A. Keypoint Reward

Here we describe the keypoint reward in more details. Keypoint distance is calculated as: $d_t^{kp}(p_t^{held}, p^{fixed}) = ||k_t^{held} - k^{targ}||$. The target keypoints, k^{targ}, represent the desired position of the held part, while k_t^{held} represent its current position. We use a logistic kernel as in [20] to transform keypoint distances into a bounded reward: $\mathcal{K}_{a,b}(d_{kp}) = (e^{-ax} + b + e^{ax})^{-1}$. The kernel can be tuned to be sensitive to distances at different scales using parameters a and b (see Table I).

Using a single kernel parameterization was not sufficient for the nut-threading task due to small geometry. Different phases of the task require motion at different scales. For example, initial placement of the nut on the bolt requires movement ranging from $0 - 2cm$. However, lowering the nut by the final thread changes the position by $< 0.1cm$. Instead, we propose a *coarse-to-fine* keypoint reward. The final reward is a sum of: (1) A *coarse reward* directing the arm towards the tip of the fixed part and; (2) a *fine reward* incentivizing more detailed motion once the arm is close to

Initial State Randomization			
Parameter	All Tasks		
Fixed: x, y, z	$[0.55, 0.65]m, [-0.05, 0.05]m, [0.0, 0.1]m$		
Hand: x, y (rel)	$[-2, 2]cm, [-2, 2]cm$		
Held: x, y (rel)	$[-3, 3]mm, [0, 0]mm$		
Parameter	$8mm$ Peg	Medium Gear	$M16$ Nut
Hand: z (rel)	$[3.7, 5.7]cm$	$[2.5, 4.5]cm$	$[0.5, 2.5]cm$
Hand: yaw	$[-45, 45]^\circ$	$[-45, 45]^\circ$	$[-120, -90]^\circ$
Held: z (rel)	$[14, 20]mm$	$[12, 15]mm$	$[10, 16]mm$
Observation Randomization			
Parameter	$8mm$ Peg	Medium Gear	$M16$ Nut
Pos-Est Noise	$2.5mm$	$2.5mm$	$2.5mm$
Force Noise	$1N$	$1N$	$1N$
EE-Pos. Noise	$0.25mm$	$0.25mm$	$0.25mm$
Dynamics Randomization			
Parameter	$8mm$ Peg	Medium Gear	$M16$ Nut
Part Friction	$[0.5, 1.0]$	$[0.38, 0.75]$	$[0.1, 0.38]$
Controller Gains	$[400, 800]$	$[400, 800]$	$[400, 800]$
Action Scale: λ	$[1.6, 2.5]cm$	$[1.6, 2.5]cm$	$[1.6, 2.5]cm$
Dead Zone	$[0, 5]N$	$[0, 5]N$	$[0, 5]N$
Force Threshold	$[5, 10]N$	$[5, 10]N$	$[5, 10]N$
Reward Specification			
Parameter	$8mm$ Peg	Medium Gear	$M16$ Nut
Coarse (a^c, b^c)	$(50, 2)$	$(50, 2)$	$(100, 2)$
Fine: (a^f, b^f)	$(100, 0)$	$(100, 0)$	$(500, 0)$
Contact-Pen: β	0.2	0.05	0.05
Success Dist.	$24mm$	$19mm$	$2.5mm$
Place Dist.	$2.5mm$	$2mm$	$2.5mm$
Episode Length	150 ($10s$)	300 ($20s$)	450 ($30s$)

TABLE I

SIMULATION PARAMETERS USED TO TRAIN FORGE POLICIES.

the part. These are implemented using different parameters for the logistic kernel,

$$R_{kp}(p^{fixed}, p_t^{held}) = \mathcal{K}^{coarse}_{a^c,b^c}(d_t^{kp}) + \mathcal{K}^{fine}_{a^f,b^f}(d_t^{kp}). \quad (1)$$

Parameters for each task can be found in Table I.

B. Task Success

Each task defines success based on the relative positions between the held and fixed parts (Table I shows *Success Dist.* as the distance between the top of the fixed part and bottom of the held part when success is achieved):

- *Peg Insertion*: The bottom of the peg is within $1mm$ of the base of the socket (equivalently, $24mm$ below the top of the socket).
- *Gear Meshing*: The bottom of the gear is within $1mm$ of the base of the gear plate (equivalently, $19mm$ below the tip of the gear peg).
- *Nut Threading*: The $M16$ nut is lowered a quarter thread (corresponding to $2.5mm$ below the tip of the bolt, as the first thread is chamfered).

For all tasks, success also requires the parts to be laterally centered.

C. IndustReal Baseline

IndustReal [41] proposes a series of techniques for sim-to-real transfer of contact-rich tasks:

- Simulation Aware Policy Updates (SAPU, sim): Penalize a policy when its actions lead to interpenetration.
- SDF Rewards (sim): Compute rewards based on signed distances between target and current asset point clouds.
- Sampling Based Curriculum (SBC, sim): Training progresses in difficulty by descreasing the fraction of environments that start near the goal state.
- Policy Level Action Integrator (PLAI, real): A method to smooth actions and reduce steady-state error.

The authors extensively evaluate the system on two contact-rich tasks: peg insertion and gear meshing.

Key Differences: Although IndustReal presents strong results, it struggles for tasks that require delicate manipulation such as nut-threading. IndustReal policies do not use force observations and are not trained to avoid excessive forces. In simulation, IndustReal policies use small gains to avoid large forces by default (resulting in maximum applied forces of $3N$). On the real robot, as our results show, a policy's forceful behaviour is largely determined by the controller gains used at deployment. In the IndustReal work, these gains were set to large values, resulting in maximum achievable forces of $15N$ or higher. In addition to IndustReal policies causing part slip, we noticed another common failure case for the nut-threading task. The arm would rotate before the nut was in contact with the bolt, causing failed thread alignment. We posit that force is a useful modality to detect when parts are aligned. This is otherwise difficult to do when the pose estimation error is too large.

Implementation Details: Beyond the key components of each framework, there are several additional differences between FORGE and the original IndustReal implementation. To make the comparison as informative and fair as possible, we used our own version of IndustReal with the following changes:

- Policy Frequency: Like FORGE, our version of IndustReal used a policy inference rate of $15Hz$ (compared to the original $60Hz$). This also required increasing the PLAI action scales by a factor of 4.
- Policy Network and Training Hyperparameters: We use the same network structure and training parameters for all methods.
- Pose Estimation Noise: Our implementation uses FORGE's noise model in simulation ($\sigma = 2.5mm$). Although this is higher than the $unif(-1mm, 1mm)$ sampling distribution originally used in IndustReal, we found the policies still trained reliably.

Nut Threading: In our work, we also consider the nut-threading task which was not considered in the original IndustReal work. We directly applied our IndustReal implementation but found that the SDF-reward function was poorly suited to learn successful threading. This is because there are only small SDF distances between partially threaded and unthreaded nuts with the same orientation. Instead, for this task only, we used the coarse-to-fine keypoint reward discussed in App. III. All other components of IndustReal were kept the same.

D. Early-Termination

Making decisions based on the success prediction action, a_t^{ET}, involves choosing a threshold, p_{term}. In this section, we investigate the trade-offs made when choosing this threshold.

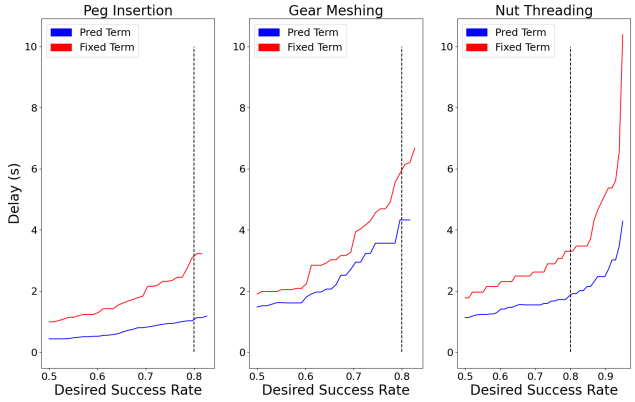

Fig. 1. **Success Prediction Analysis** Relationship between Delay Time and Success Rate for two early termination methods (generated by varying each method's respective parameter: T or p_{term}). The *Pred Term* method leads to lower delays than the *Fixed Term* method, especially at higher success rates. The vertical line shows a 0.8 success rate.

We use *Delay Time (s)* to capture efficiency (lower values are better). Delay time measures the time between when success occurred and when the episode was terminated ($a_t^{ET} > p_{term}$). We compare the proposed method (Pred Term) to a standard termination method that stops the policy after a fixed duration, T (Fixed Term). Each method has a parameter that can be tuned to produce a different success rate (fraction of episodes that are successful when terminated). However, this will introduce a trade-off with delay time:

- Fixed Term (T): Waiting too long is inefficient while terminating too early will harm success rates.
- Pred Term (p_{term}): A high threshold can cause extra delay while a low threshold can affect success rate.

Fig. 1 is a simulated analysis that shows the relationship between *Delay Time* and *Success Rate* for each method. Each line was generated by measuring the success rate and corresponding delay time across a fine discretization of each method's termination parameter. These were then sorted by success rate and plotted.[1] As a practitioner, one could choose a desired success rate and find the resulting delay.

Across all tasks, we see that the *Fixed Term* method leads to longer delays, especially at higher success rates (we plot a vertical line to show the 0.8 success rate). The early termination action, a^{ET}, allows for dynamic episode lengths, leading to high success rates with smaller delay times.

E. Snap-fit Task

Simulation Model: For forceful insertion, we consider a snap-fit task. In the real world, these parts typically have clips that deform for successful insertion. Because *Factory* [25] uses a rigid-body simulator, we approximate deformation by using a spring model on the snap-fit clips. Varying the stiffness of this spring changes the amount of force necessary for insertion.

Training Details: At deployment time, we assume the amount of force required for insertion is unknown. To ensure a single policy can solve snap-fit tasks with varying force requirements, we randomize both the gains of the torsion spring and the force threshold when training in simulation. We ensure the force threshold is higher than the necessary force required for insertion. For this environment only, we found it was helpful for the policy to have noisy proportional gains of the controller as input.

[1]Similar to an ROC plot, but higher areas above the curve are better.

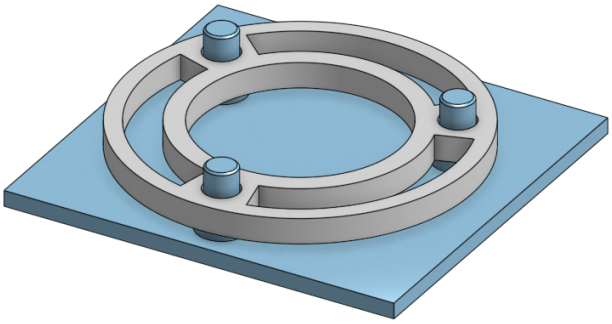

Fig. 2. Simulated assets for the ring insertion task. The ring gear (grey) is inserted onto the gearbox plate (blue).

IV. Planetary Gearbox

For the planetary gearbox, we trained policies for the following tasks: Ring Insertion, Small Gear Meshing, Large Gear Meshing, and M16 Nut Threading.

Gear Tasks: The gearbox requires insertion of three small gears, each with one abutting gear, and one large gear with three abutting gears. In simulation, the small and large gear meshing tasks had one abutting gear each. This is similar to deployment for the small gear which achieved a high success rate ($15/15$). However, when the large gear is deployed, it needs to mesh with the three already inserted small gears. This is much harder than how the policy was trained and could be a cause of the performance drop for this task ($3/5$). Note these statistics come from five executions of the entire planetary gearbox assembly (and hence a different number of trials per gear size).

Ring Insertion: The outer ring gear must be inserted onto the three bolts of the gearbox base. We designed simulation assets for the corresponding parts (see Fig. 2) and trained a policy using the FORGE framework. We assume there is small orientation error on the ring ($< 5°$) during training. Success is defined as having the ring gear placed close to the gearbox base ($< 2mm$ displacement) and all three bolt holes aligned.

Gearbox Design: Note, we also designed a "lock" for the gear carrier which is removed by the robot after the small gears are inserted. This ensures a fixed base during the small gear insertions (see video).

Policy Selection: All policies were trained using the FORGE framework including force observations. We trained one policy per task without any additional checkpoint selection procedure. The M16 policy was chosen as the best policy from our main evaluation. For the gearbox experiments only, we selected high control stiffness for the roll and pitch

dimensions of the impedance controller, as the policy does not generate actions for these degrees of freedom.

Task Execution: To pick up the held parts, we assume a known grasp location which was predetermined (with small noise from placement error). However, the location of the corresponding fixed parts were estimated from the *IndustReal* perception system [41]. Grasping and movement to the initial state for policy execution was performed with a standard position controller. No additional artificial noise or initial-state randomization was added for the gearbox experiments.

Perception: The perception system from IndustReal [41] assumes the z-position of parts are known and uses a Mask-RCNN model [45] to estimate bounding boxes from which planar locations can be backed out. We retrained the Mask-RCNN model using data we collected. Perception errors are largely caused by extrinsic calibration and bounding box prediction errors.

V. Generalization across Part Geometry

For our real robot experiments, we focused on a single part size for each task. In this section, we show that FORGE policies achieve similar simulated performance across part sizes. To do so, we train and evaluate specialist policies for three part sizes for each task (see diagonal grid elements in Figure 3).

- Peg Insertion: $8mm$, $12mm$, $16mm$
- Gear Meshing: Small, Medium, Large
- Nut Threading: $M12$, $M16$, $M20$

To assess policy generalization, we also evaluate policy performance in simulation for the part sizes they were not trained on. The results in Figure 3 show the success rates of this evaluation. Policies achieve high performance on the part size for which they were trained. In most cases, policies also generalize to other part sizes. The case with the worst generalization is "Train on Medium/Large Gear" and "Evaluate on Small Gear". This can be explained because of the significant geometry differences between the parts. The small gear has a much smaller base, so a search strategy that would work for the larger gears, would cause the small one to fall off the peg. In future work, we are hopeful we can train a single policy per task which generalizes across multiple part geometries by randomizing geometry in simulation.

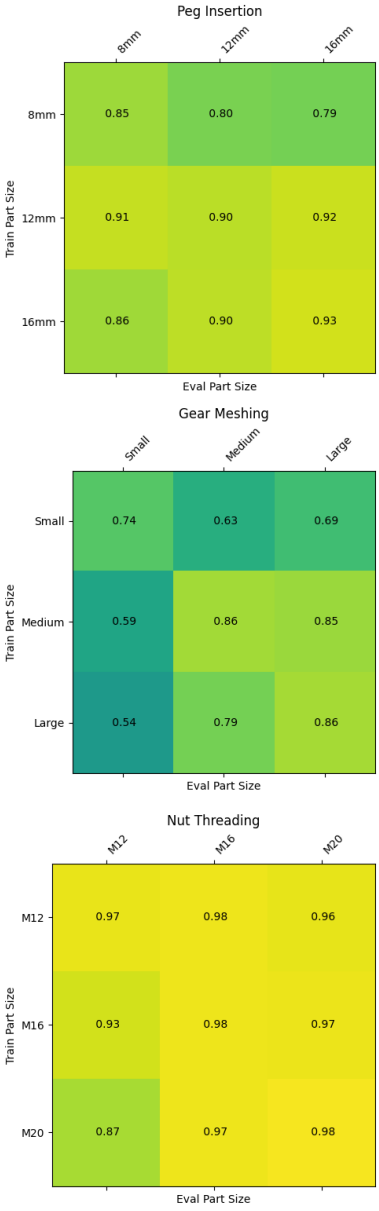

Fig. 3. **Part Size Generalization** Specialist policies trained on a single part size tend to generalize to other part sizes. Each cell aggregates success rates from 3 policies trained with different random seeds and evaluated 128 times each.

REFERENCES

[1] F. Abu-Dakka, L. Rozo, and D. Caldwell, "Force-based learning of variable impedance skills for robotic manipulation," in *Humanoids*. IEEE, 2018.

[2] T. Davchev, K. S. Luck, M. Burke, F. Meier, S. Schaal, and S. Ramamoorthy, "Residual Learning From Demonstration: Adapting DMPs for Contact-Rich Manipulation," *IEEE RA-L*, 2022.

[3] J. Luo, O. Sushkov, R. Pevceviciute, W. Lian, C. Su, M. Vecerik, N. Ye, S. Schaal, and J. Scholz, "Robust Multi-Modal Policies for Industrial Assembly via Reinforcement Learning and Demonstrations: A Large-Scale Study," in *RSS*, 2021.

[4] 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 *ICRA*. IEEE, 2019.

[5] J. Luo, Z. Hu, C. Xu, Y. L. Tan, J. Berg, A. Sharma, S. Schaal, C. Finn, A. Gupta, and S. Levine, "SERL: A Software Suite for Sample-Efficient Robotic Reinforcement Learning," *arXiv:2401.16013*, 2024.

[6] J. Luo *et al.*, "Reinforcement Learning on Variable Impedance Controller for High-Precision Robotic Assembly," in *ICRA*. IEEE, 2019.

[7] Y. Fan, J. Luo, and M. Tomizuka, "A Learning Framework for High Precision Industrial Assembly," in *ICRA*. IEEE, 2019.

[8] M. A. Lee, C. Florensa, J. Tremblay, N. Ratliff, A. Garg, F. Ramos, and D. Fox, "Guided Uncertainty-Aware Policy Optimization: Combining Learning and Model-Based Strategies for Sample-Efficient Policy Learning," in *ICRA*. IEEE, 2020.

[9] 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 *IROS*. IEEE, 2018.

[10] T. Inoue, G. De Magistris, A. Munawar, T. Yokoya, and R. Tachibana, "Deep reinforcement learning for high precision assembly tasks," in *IROS*. IEEE, 2017.

[11] 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," *IEEE T-RO*, 2020.

[12] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IROS*. IEEE, 2012.

[13] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016–2021.

[14] R. Tedrake and the Drake Development Team", "Drake: Model-based design and verification for robotics," 2019.

[15] V. Makoviychuk *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv:2108.10470*, 2021.

[16] J. Hwangbo *et al.*, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, 2019.

[17] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged Locomotion in Challenging Terrains using Egocentric Vision," in *CORL*, 2022.

[18] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid Locomotion via Reinforcement Learning," in *RSS*, 2022.

[19] N. Rudin, D. Hoeller, M. Hutter, and P. Reist, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," in *CORL*, 2021.

[20] A. Allshire *et al.*, "Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger," in *IROS*. IEEE, 2022.

[21] I. Akkaya *et al.*, "Solving rubik's cube with a robot hand," *arXiv:1910.07113*, 2019.

[22] A. Handa *et al.*, "DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality," in *ICRA*. IEEE, 2023.

[23] L. Lan, D. M. Kaufman, M. Li, C. Jiang, and Y. Yang, "Affine body dynamics: fast, stable and intersection-free simulation of stiff materials," *ACM Trans. Graph.*, 2022.

[24] M. Macklin, K. Erleben, M. Müller, N. Chentanez, S. Jeschke, and Z. Corse, "Local optimization for robust signed distance field collision," *Proc. ACM Comput. Graph. Interact. Tech.*, 2020.

[25] Y. Narang *et al.*, "Factory: Fast Contact for Robotic Assembly," in *RSS*, 2022.

[26] J. Yoon, M. Lee, D. Son, and D. Lee, "Fast and Accurate Data-Driven Simulation Framework for Contact-Intensive Tight-Tolerance Robotic Assembly Tasks," *arXiv:2202.13098*, 2022.

[27] L. Ljung, "System identification," in *Signal analysis and prediction*. Springer, 1998, pp. 163–173.

[28] B. Acosta, W. Yang, and M. Posa, "Validating robotics simulators on real-world impacts," *IEEE RA-L*, 2022.

[29] M. Guo, Y. Jiang, A. E. Spielberg, J. Wu, and K. Liu, "Benchmarking Rigid Body Contact Models," in *LDCC*, 2023.

[30] A. Apolinarska *et al.*, "Robotic assembly of timber joints using reinforcement learning," *Automation in Construction*, 2021.

[31] C. Beltran-Hernandez, D. Petit, I. Ramirez-Alpizar, and K. Harada, "Variable compliance control for robotic peg-in-hole assembly: A deep-reinforcement-learning approach," *Applied Sciences*, 2020.

[32] M. Hebecker, J. Lambrecht, and M. Schmitz, "Towards Real-World Force-Sensitive Robotic Assembly through Deep Reinforcement Learning in Simulations," in *AIM*. IEEE, 2021.

[33] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *ICRA*. IEEE, 2018.

[34] O. Spector and M. Zacksenhouse, "Learning Contact-Rich Assembly Skills Using Residual Admittance Policy," in *IROS*. IEEE, 2021.

[35] S. Jin, X. Zhu, C. Wang, and M. Tomizuka, "Contact Pose Identification for Peg-in-Hole Assembly under Uncertainties," in *ACC*. IEEE, 2021.

[36] X. Zhang, M. Tomizuka, and H. Li, "Bridging the Sim-to-Real Gap with Dynamic Compliance Tuning for Industrial Insertion," in *ICRA*. IEEE, 2024.

[37] G. Schoettler and et al., "Meta-reinforcement learning for robotic industrial insertion tasks," in *IROS*. IEEE, 2020.

[38] Z. Wu, Y. Xie, W. Lian, C. Wang, Y. Guo, J. Chen, S. Schaal, and M. Tomizuka, "Zero-shot policy transfer with disentangled task representation of meta-reinforcement learning," in *ICRA*. IEEE, 2023.

[39] N. Vuong, H. Pham, and Q. Pham, "Learning Sequences of Manipulation Primitives for Robotic Assembly," in *ICRA*. IEEE, 2021.

[40] K. Zhang, M. Sharma, J. Liang, and O. Kroemer, "A modular robotic arm control stack for research," *arXiv:2011.02398*, 2020.

[41] B. Tang *et al.*, "IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality," in *RSS*, 2023.

[42] X. Zhang, C. Wang, L. Sun, Z. Wu, X. Zhu, and M. Tomizuka, "Efficient Sim-to-real Transfer of Contact-Rich Manipulation Skills with Online Admittance Residual Learning," in *CORL*, 2023.

[43] R. Martín-Martín, M. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning," in *IROS*. IEEE, 2019.

[44] D. Son, H. Yang, and D. Lee, "Sim-to-Real Transfer of Bolting Tasks with Tight Tolerance," in *IROS*. IEEE, 2020.

[45] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *ICCV*. IEEE, 2017.