

FORGE: Force-Guided Exploration for Robust Contact-Rich Manipulation under Uncertainty

Michael Noseworthy¹, Bingjie Tang², Bowen Wen³, Ankur Handa³, Chad Kessens⁴, Nicholas Roy¹
Dieter Fox³, Fabio Ramos³, Yashraj Narang³, Iretiayo Akinola³

Abstract—We present FORGE, a method for sim-to-real transfer of force-aware manipulation policies in the presence of significant pose uncertainty. During simulation-based policy learning, FORGE combines a *force threshold* mechanism with a *dynamics randomization* scheme to enable robust transfer of the learned policies to the real robot. At deployment, FORGE policies, conditioned on a maximum allowable force, adaptively perform contact-rich tasks while avoiding aggressive and unsafe behaviour, regardless of the controller gains. Additionally, FORGE policies predict task success, enabling efficient termination and autonomous tuning of the force threshold. We show that FORGE can be used to learn a variety of robust contact-rich policies, including the forceful insertion of snap-fit connectors. We further demonstrate the multistage assembly of a planetary gear system, which requires success across three assembly tasks: nut threading, insertion, and gear meshing. Project website: <https://noseworm.github.io/forge/>

I. INTRODUCTION

We are interested in developing *sim-to-real* techniques for learning assembly primitives (e.g., low-clearance insertion or nut-threading). Over the past decade, sim-to-real techniques have led to advances in dexterous manipulation and legged locomotion [1], [2], [3], [4]. However, similar results have only recently been achieved for robotic assembly, which requires efficient and accurate simulation of the detailed, low-clearance parts [5], [6], [7], [8], [9], [10]. Even with these advances, successful sim-to-real deployment remains challenging for contact-rich tasks.

Naively, policies can be too aggressive, leading to catastrophic part slip or damage that makes the task difficult or impossible to complete (see Figure 1). This is particularly pronounced when there is pose uncertainty and search behaviours that rely on contact are necessary [11], [12]. The required contact between parts can lead to undesirable outcomes if the forces are too high. Heuristic approaches, such as spiral search [11], [13], can limit the applied force but these approaches are task-specific and can be inefficient.

Reinforcement learning offers a general paradigm for developing more flexible search behaviours. However, previous works typically rely on additional procedures to ensure policies are deployed safely with desirable force profiles. For example, policies trained using the *IndustReal* framework [7] do not observe or adapt to contact forces. Instead, as our experiments show, forceful behaviour is determined by careful controller design and gain tuning. Other works provide methods to optimize or adapt gains online [14], [15]. Importantly, the desired force profile of a policy depends on

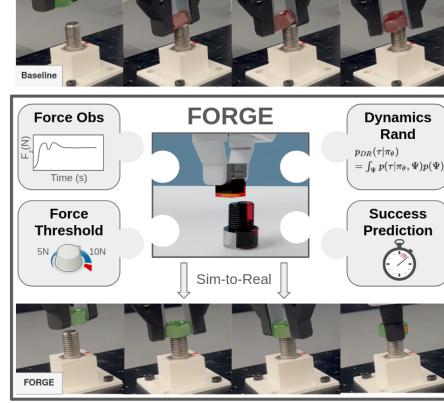


Fig. 1. FORGE uses force feedback to learn search behaviours for contact-rich tasks with position estimation uncertainty. It combines *dynamics randomization*, a *force threshold*, and *success prediction* for robust sim-to-real transfer. The resulting policies are *safe* and *efficient* (bottom) compared to aggressive baseline policies that cause parts to slip (top).

the task at hand. For example, threading a nut may fail if the applied force is too high, while a snap-fit connector might require large forces to ensure proper insertion. Therefore, it is important to have simple and efficient methods to tune the policy’s force profile.

In this work, we propose FORGE: a framework for developing force-aware sim-to-real policies for assembly tasks. FORGE policies, trained solely in simulation, use external force observations to achieve efficient and gentle behaviour. Additionally, policies are trained without precise knowledge of part poses, leading to emergent search behaviours that are robust to significant levels of pose uncertainty.

FORGE has two complementary components to ensure policies are robust to contact. First, we propose to condition policies on a *force threshold* that should not be exceeded during task execution. Second, policies are trained to maintain this threshold under a wide range of dynamics randomizations (we randomize *robot*, *controller*, and *part* properties). Together, these components result in policies that can modulate their actions to achieve a force profile that respects the interpretable scalar force threshold. By randomizing this threshold during training, we are able to tune it at deployment time without retraining the policy.

For tasks with high enough clearance, a small force threshold is sufficient and no tuning is necessary. However, tasks that require significant force to succeed (e.g., snap-fit connectors) may fail if the threshold is set too low. When the required force is not known a priori, we present an automatic tuning procedure which leverages a notion

¹MIT ²USC ³NVIDIA ⁴DEVCOM ARL.

of *success prediction*. Based on the outcome of a policy execution, the threshold can be iteratively adjusted for future trials. To automate this tuning, FORGE policies are trained to predict whether an episode succeeded or failed. We validate this procedure by showing successful sim-to-real transfer on a snap-fit connector requiring $15N$ for insertion.

Furthermore, we show how success prediction can lead to more efficient policy termination. Standard practice in *sim-to-real* assembly is to execute policies for a fixed duration [7] which can lead to premature termination or delays. Instead, the policy can terminate when it believes it is in a successful state. We show that success prediction, also trained in simulation, robustly transfers to the real world and does so more reliably when using force observations [16].

In summary, our contributions are:

- 1) **A method to specify maximum allowable contact force** during policy execution. This results in policies that exhibit safe search behaviour even with significant levels of position estimation error (up to $5mm$).
- 2) **A dynamics randomization scheme** that reduces tuning to an interpretable scalar force-threshold parameter (instead of controller gains).
- 3) **A method for success prediction** that enables automatic force-threshold tuning and efficient policy termination, reducing delay times up to 66%.
- 4) **A demonstration of multi-part assembly** of a planetary gearbox requiring a diverse set of skills, including the challenging task of fastening nuts and bolts.

Results are shown for over 1000 real-world trials and multiple tasks. We plan to release the code with the paper.

II. RL FOR CONTACT-RICH ASSEMBLY

We want to learn policies for tasks with tight tolerances and detailed geometry. We first describe the problem formulation before introducing FORGE in the next section.

A. Assembly Tasks

Each task involves mating two parts: one grasped and another fixed to the workspace. For our main evaluations, we consider all three tasks from *Factory* [5] and demonstrate the first sim-to-real transfer for threading a small M16 nut (see Fig. 2). We also consider forceful snap-fit insertion and multi-step assembly in Sec. V-D and Sec. V-E respectively.

Peg Insertion: A round peg with $8mm$ diameter needs to be inserted into a socket with $0.5mm$ diametrical clearance.

Gear Meshing: A gear needs to be inserted onto a peg with $0.5mm$ clearance. Other gears are present and the teeth of adjacent gears must be aligned for successful meshing.

Nut Threading: Instead of fully lowering a nut onto a bolt as in *Factory*, we define the *nut threading* task as successfully threading the nut such that it cannot be lifted by a vertical motion (we find lowering by a quarter-thread is sufficient). Because our robot has joint limits, and to prevent the need to regrasp, we assume the nut and bolt are initially oriented¹ such that success can be achieved with a

¹We leave the more challenging, yet realistic, scenario involving completely unobserved thread orientation to future work.

single revolution of the wrist joint. We consider nuts with a relatively small size (M16) compared to previous sim-to-real work (M48) [17]. A successful search behaviour will resolve lateral uncertainty and place the nut on the bolt before rotating the wrist (otherwise the threads may not mesh).

B. POMDP Formulation

We formulate our problem as a *Partially Observable Markov Decision Process (POMDP)* [18], [19]. The goal is to learn a policy, $\pi_\theta(a_t|o_1, \dots, o_t)$, that maximizes the expected return:

$$J(\pi_\theta) = \mathbb{E}_{\tau \sim p(\tau|\pi_\theta, \Psi)} [\sum_{t=0}^{\infty} \gamma^t r_t] \quad (1)$$

where $\tau = (s_0, a_0, o_0, s_1, a_1, o_1, \dots)$ is the trajectory of states, actions, and observations resulting from the robot following policy π_θ . Below, we further specify the components of the POMDP for contact-rich tasks.

States (\mathcal{S}): A state, $s_t \in \mathcal{S}$ consists of the pose and velocities of the end-effector (EE), fixed part, and held part: $p^{ee}, p^{fixed}, p^{held} \in SE(3)$ and $v^{ee}, v^{held} \in \mathbb{R}^6$. We also include the contact force experienced by the end-effector, $F^{ee} \in \mathbb{R}^3$, and time-invariant information about the dynamics properties of the robot, controller, and parts (e.g., mass or joint-friction): $\Psi = (\psi_{robot}, \psi_{control}, \psi_{parts})$.

Observations (Ω): As it is difficult to accurately estimate the full state, all our policies observe:

- Noisy EE pose and velocity: $\hat{p}^{ee} \in SE(3)$, $\hat{v}^{ee} \in \mathbb{R}^6$
- Estimated contact force: $\hat{F}^{ee} \in \mathbb{R}^3$
- Noisy estimate of the fixed part's pose: $\hat{p}^{fixed} \in SE(3)$

We do not include pose or velocity of the held part because it can move in the gripper and be difficult to track. Likewise, we do not observe Ψ , but include the previous action, a_{t-1} , to help infer dynamics. See the website for noise models.

Actions (\mathcal{A}): Control targets for a task-space impedance controller [7], [20]. As in previous work [5], [7], we assume all parts are in an upright orientation. Thus it is sufficient for the policy to only have control authority over the (x, y, z, yaw) -dimensions: $a_t \in \mathcal{A} = \mathbb{R}^4$.

Transition Function ($T_\Psi : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$): T is parameterized by the dynamics parameters, Ψ and is specified using the *IsaacGym* [21] simulator. The sim-to-real gap comes from the mismatch between Ψ^{sim} and Ψ^{real} .

Observation Function ($O : \mathcal{S} \times \mathcal{A} \rightarrow \Omega$): The position of the fixed part is assumed to have up to $5mm$ error. Gaussian noise is assumed for each of the other observations.

Reward Function ($R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$): Each task is described using a keypoint reward: $R_{kp}(p^{fixed}, p_t^{held})$ [5], [22], which is modified to account for small, threaded geometries (see the website for more details). We also add two discrete *bonus* rewards that are given when important phases of the tasks are reached: once the held part is centered on top of the fixed part and once the task is successful:

$$R_{bonus}(p^{fixed}, p_t^{held}) = \mathbb{I}_{place} + \mathbb{I}_{success}. \quad (2)$$

We found the bonuses led to more robust learning when there is significant pose uncertainty.

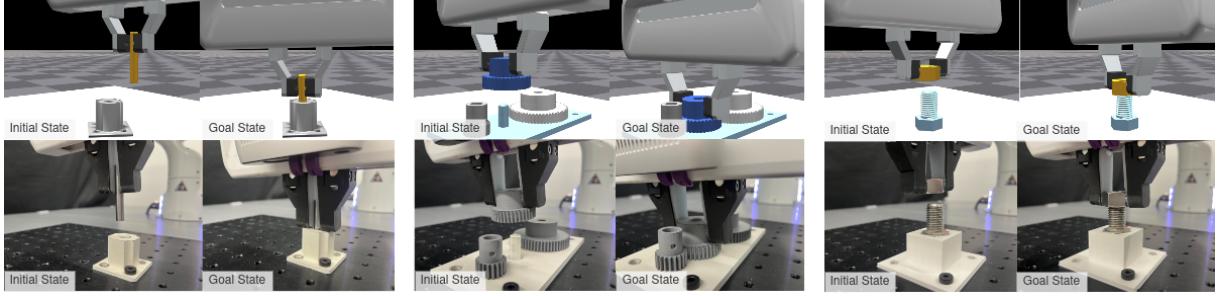


Fig. 2. FORGE is evaluated on three tasks from *Factory* [5]: Peg Insertion, Gear Meshing, and Nut Threading. Each task is trained solely in simulation (top) and transferred directly to the real robot (bottom).

III. FORGE: ROBUST SEARCH UNDER UNCERTAINTY

FORGE uses on-policy RL to learn search behaviours in simulation. A *force threshold* (Sec. III-A) and *dynamics randomization* (Sec. III-B) are introduced for robust sim-to-real transfer. FORGE also introduces *success prediction* (Sec. III-C) for efficient termination and force threshold tuning.

A. Force Threshold

During policy execution, excessive force can cause parts to slip or become damaged. Although it may be possible to recover from small amounts of slip with the right sensors (e.g., wrist camera or tactile), we prefer to avoid these scenarios. Instead, we propose to condition the policy on a *force threshold*, F_{th} : $\pi(a|o, F_{th})$. During training, the policy is penalized if the contact force, F_t^{ee} , exceeds the threshold. Concretely, we add an additional term to the reward function:

$$R_{contact_pen}(F_t^{ee}) = -\beta * \max(0, ||F_t^{ee}|| - F_{th}). \quad (3)$$

Note this is related to [10] which conditions the policy on a *desired force* instead of an *excessive force*.

At deployment time, F_{th} can be set or tuned based on task requirements. Most of the tasks we consider have positive clearance and do not require much force to succeed. A relatively low force threshold is sufficient to prevent slip and tuning the threshold is not necessary. For forceful insertion, nominal insertion forces are often specified on part datasheets and the threshold should be higher than this value. We also present an automatic tuning procedure in Sec. III-C.

B. Dynamics Randomization

To successfully deploy policies trained in simulation, it is important that the trajectory distribution experienced during training is similar to what it would be when deployed: $p(\tau^{real}|\pi_\theta, \Psi^{real}) \approx p(\tau^{sim}|\pi_\theta, \Psi^{sim})$. The difference between these distributions is usually referred to as the *sim-to-real gap*. This gap is usually handled by (1) system identification (Sys-ID) [23] or (2) dynamics randomization (DR) [10], [24]. The goal of Sys-ID is to tune Ψ^{sim} to be close to Ψ^{real} . This itself is a complicated tuning procedure that may need to be redone for every new set of parts.

Instead, we follow the DR approach which learns policies that are *robust* to a wide range of dynamics parameters. Concretely, we optimize a version of Eq. 1 where:

$$\tau \sim p_{DR}(\tau|\pi_\theta) = \int p(\tau|\pi_\theta, \Psi)p(\Psi)d\Psi. \quad (4)$$

The integral is approximated with Monte Carlo samples from a randomization distribution. We now describe the variables that are randomized (see website for values).

Controller Randomization: The controller has a large impact on contact forces. This work uses impedance-control where applied forces are computed as:

$$p_t^{targ} = clip(combine(a_t, p^{fixed}), \lambda), \quad (5)$$

$$F^{targ} = k_p(p_t^{targ} - p_t^{ee}) - k_d v_t^{ee}. \quad (6)$$

First, the policy outputs a relative pose, a_t , which is applied to the fixed part's pose to get an absolute target pose, p_t^{targ} . This pose is clipped by an action scale, λ , to ensure that the target is not too far from the EE's current pose. As in previous work, we use critically damped gains to ensure stable controllers: $k_d = 2\sqrt{k_p}$ [10], [14], [25]. The controller thus depends on two parameters which govern how much force can be commanded: $\lambda \times k_p$. We randomize both quantities so that the range of maximum commandable forces is in $[6.4, 20.0]N$. Note that the control parameters are not included in the observations, so the policy must adjust its behavior based on force measurements. This reduces the policy's dependence on a particular controller implementation.

Controller tuning [7] or optimization [14] is a costly and often complex procedure. Randomization [24] has the additional benefit that the policy is robust to a range of control parameters, greatly simplifying deployment.

Part Randomization: As parts slide against each other, material friction will affect lateral forces. To ensure policies can work across a range of materials, we randomize part mass and friction [24], [26], [27].

Robot Dynamics Randomization: Due to phenomena such as joint friction [24], [28], the applied force may be smaller than the commanded force. We implement a simple way to account for this: inducing a randomized *dead-zone* in simulation. Each episode, a dead-zone is selected for each dimension, F_i^{DZ} , where commanded forces below this value are clamped to zero: $|F_i^{applied}| = \max(0, |F_i^{targ}| - F_i^{DZ})$. This enables the policy to increase its target which can help apply more force when needed or reduce steady-state error.

These randomizations lead to a policy that is robust to a wide range of dynamics parameters. Combined with the force threshold, the policy can modulate its actions to achieve safe interaction. For example, with higher gains, the policy will output smaller actions to limit the contact force.

C. Success Prediction

Although success is clearly defined in simulation where we have access to noiseless poses, it is difficult to reliably predict in the real world [16]. Consider the nut-threading task, where the distance between a successfully threaded nut and a loose nut is a fraction of a millimeter. We propose to train a success predictor which can robustly transfer from sim-to-real. Concretely, we share the weights of the policy network with the success predictor by expanding the action space of the policy to include an early termination action: $a_t^{ET} \in [0, 1]$. To train the policy to output the correct action, we include an early termination penalty, R_t^{ET} , which penalizes incorrect success predictions:

$$R_t^{ET}(a_t, y_t) = -|a_t^{ET} - y_t|, \quad (7)$$

where y_t is the true success label at time t . This reward can also encourage behaviours that elicit the underlying success state (e.g., pull upwards on the nut to check if it is threaded).

Early Termination: Efficient termination is a desirable property for industrial applications where cycle times matter. We want the policy to terminate as soon as the task has succeeded and no sooner. During training, episodes are executed for the maximum length. At deployment, a confidence threshold, p_{term} , can be used to terminate the episode: $a_t^{ET} > p_{term}$. See the website for analysis on the performance trade-offs for choosing p_{term} .

Force Threshold Tuning: For forceful insertion tasks, we may not know how much force is required. Consider snap-fit connectors which require deformation. The required force depends on material properties that may be unknown and could change with extended use. To tune the force threshold, we leverage success prediction. Conservatively, we start with a low threshold of $7.5N$, which helps avoid slip and damage. If policy execution reaches a timeout before success is predicted, we increase the threshold and try again. This can be done automatically, without manual resets, until success occurs.

IV. EXPERIMENT SETUP

A. Robot System

We use a *Franka Panda* robot with the *FrankaPy* [29] library for impedance control. All policies send control targets at $15Hz$ while the controller operates at $1000Hz$. The Panda has joint-torque sensing, which is projected to EE-frame forces when needed by the policy [28]. Alternatively, a force-torque sensor could be used.

For the majority of our experiments, we calibrate the poses of each fixed object and artificially add noise. This allows us to analyze performance under known levels of position estimation error. The calibration is done by guiding the arm to a successful pose for the respective task from which a nominal initial pose can be backed out. Unless otherwise reported, our real experiments use the same initial state randomization as in simulation (see website). For our last experiment, we assemble a planetary gear box (Sec. V-E) using the perception system from *IndustReal* [7] (see the website for more details).

B. Policy Training

Simulator: All policies are trained using the *Factory* simulation methods within IsaacGym [5]. In simulation, we have access to external contact forces experienced by the end-effector (akin to what we have access to on the Panda). Noisy forces are used as policy input, whereas ground-truth forces are used to compute the excessive-force penalty. We use recurrent PPO [30] with asymmetric actor-critic [31] to handle partial observability. Details on initial state and observation randomization can be found on the website.

Checkpoint Selection: For all tasks and models, we train three policies with separate random seeds. On the real-robot, results are averaged across the three policies.

Observation and Action Frames: For generalization across the workspace, we assume actions and observations are relative to the fixed part. Specifically, the policy outputs a $4D$ relative transform from the tip of the fixed part (we assume upright parts). The control target is computed from the fixed part's pose estimate and the relative pose from the policy. The policy output is bounded, limiting the operational volume of the end-effector (targets can be up to $5cm$ away in all directions). Similar to the action space, all position observations are relative to the tip of the fixed part.

C. Baselines and Ablations

We compare FORGE to two baselines:

IndustReal [7]: Policies trained using the IndustReal framework do not have velocity or force observations. A full description of the differences can be found on the website. The PLAI parameters from IndustReal are set to achieve similar maximum forces to what FORGE policies can command.

Baseline: Similar to FORGE but does not use force observations, dynamics randomization, or an excessive force penalty. However, it is trained with success prediction so that meaningful episode durations can be reported.

Ablations: In addition to the baselines, we also ablate each of the main components of FORGE for our sim-to-real analysis: Force (*No Force*), Dynamics Randomization (*No DR*), and Excessive Force Penalty (*No FP*). For the *FORGE (No FP)* model, which ablates the contact penalty reward term, we evaluate using two P-gain levels. Note that *FORGE (No FP)* results are not reported for nut threading as we found that the nut always slipped out of the gripper.

V. RESULTS AND DISCUSSION

A. Baseline Comparisons

(Q1) Does FORGE lead to more robust sim-to-real transfer? (Q2) Do FORGE policies have more desirable behavioural properties?

Along with success rate, used to measure robustness for Q1, the following metrics are reported for Q2:

- Duration (s): For successful episodes, time to reach a successful state (independent of success prediction).
- $F_{mean}, F_{max}(N)$: Forces experienced by the robot.

	Episode		Force		Early Termination		
	Success Rate ↑	Duration (s) ↓	F_{mean} (N) ↓	F_{max} (N) ↓	Precision ↑	Recall ↑	Delay (s) ↓
8mm Peg							
FORGE	0.84 (0.05)	2.82 (0.20)	5.51 (0.24)	12.84 (0.37)	1.00 (0.0)	1.00 (0.0)	2.19 (0.08)
FORGE (No Force)	0.82 (0.06)	3.18 (0.36)	7.09 (0.35)	14.16 (0.39)	0.59 (0.08)	0.81 (0.07)	4.12 (0.37)
FORGE (No DR)	0.91 (0.04)	2.03 (0.19)	6.28 (0.24)	13.08 (0.36)	1.00 (0.0)	0.98 (0.02)	2.51 (0.04)
FORGE (No FP, 400kp)	0.64 (0.07)	3.21 (0.35)	6.94 (0.13)	11.94 (0.24)	0.83 (0.07)	0.92 (0.05)	2.85 (0.40)
FORGE (No FP, 600kp)	0.71 (0.07)	2.88 (0.35)	10.66 (0.15)	16.58 (0.32)	0.91 (0.05)	0.97 (0.03)	2.40 (0.22)
Baseline	0.64 (0.07)	2.35 (0.27)	11.81 (0.21)	17.93 (0.41)	0.97 (0.03)	1.00 (0.0)	2.74 (0.06)
IndustReal	0.82 (0.06)	3.41 (0.24)	9.45 (0.14)	21.15 (0.26)	N/A	N/A	6.59 (0.24)
Medium Gear	Success Rate ↑	Duration (s) ↓	F_{mean} (N) ↓	F_{max} (N) ↓	Precision ↑	Recall ↑	Delay (s) ↓
FORGE	0.98 (0.02)	3.14 (0.39)	7.95 (0.11)	15.10 (0.45)	0.95 (0.03)	1.00 (0.0)	3.20 (0.28)
FORGE (No Force)	0.93 (0.04)	3.06 (0.29)	8.49 (0.23)	14.68 (0.39)	0.60 (0.08)	1.00 (0.0)	5.96 (0.70)
FORGE (No DR)	0.87 (0.05)	3.42 (0.50)	7.15 (0.20)	13.94 (0.36)	0.90 (0.05)	1.00 (0.0)	3.04 (0.24)
FORGE (No FP, 400kp)	0.82 (0.06)	3.57 (0.26)	6.52 (0.14)	10.97 (0.24)	1.00 (0.0)	1.00 (0.0)	2.87 (0.16)
FORGE (No FP, 600kp)	0.73 (0.07)	3.08 (0.29)	9.48 (0.23)	15.73 (0.30)	0.94 (0.04)	0.97 (0.03)	3.91 (0.39)
Baseline	0.69 (0.07)	2.90 (0.37)	11.67 (0.45)	18.29 (0.40)	0.90 (0.05)	0.97 (0.03)	4.68 (0.41)
IndustReal	0.87 (0.05)	8.44 (0.61)	9.80 (0.16)	20.48 (0.26)	N/A	N/A	6.56 (0.61)
M16 Nut	Success Rate ↑	Duration (s) ↓	F_{mean} (N) ↓	F_{max} (N) ↓	Precision ↑	Recall ↑	Delay (s) ↓
FORGE	0.69 (0.07)	11.38 (0.47)	7.82 (0.13)	14.52 (0.22)	0.74 (0.08)	0.74 (0.08)	6.54 (1.25)
FORGE (No Force)	0.40 (0.07)	14.09 (1.11)	8.34 (0.15)	15.04 (0.17)	0.33 (0.11)	0.33 (0.11)	11.48 (1.63)
FORGE (No DR)	0.56 (0.07)	11.12 (0.14)	7.65 (0.17)	14.10 (0.24)	0.72 (0.09)	0.86 (0.08)	8.10 (1.45)
Baseline	0.40 (0.07)	11.19 (0.24)	10.51 (0.51)	17.34 (0.36)	0.72 (0.11)	0.93 (0.07)	10.16 (1.69)
IndustReal	0.36 (0.07)	22.27 (0.64)	12.63 (0.31)	22.34 (0.33)	N/A	N/A	7.73 (0.64)

TABLE I

BASELINE COMPARISON FORGE IS COMPARED TO BASELINES THAT ARE NOT FORCE-AWARE (INDUSTREAL [7] AND BASELINE). IT IS ALSO COMPARED TO ABLATIONS THAT DO NOT OBSERVE FORCE (NO FORCE), A FORCE PENALTY (NO FP), OR DYNAMICS RAND (NO DR). EVALUATIONS ARE PERFORMED OVER A TOTAL OF 855 TRIALS ON THE REAL ROBOT (45 PER ROW). STANDARD ERRORS ARE INCLUDED IN PARENTHESES.

Each reported metric represents 45 trials spread across 5 workspace locations for the fixed part, and 3 position-estimation error levels ranging from 0 – 5mm (see Fig. 3). Similar randomization ranges were used as in simulation except for the in-hand part randomization where the part was centered in the gripper. Results are reported in Table I.

One conclusion for Q1 is that FORGE outperformed the *Baseline* method for all tasks and *IndustReal* for both the gear meshing and nut threading tasks. Ablations show that that the primary performance gains of FORGE come from including force observations and the excessive force penalty. Although dynamics randomization did not significantly affect success rate, we later show it is important for robustness across controller gains.

Examining the behavioural metrics for Q2, we notice that FORGE used less force than both baselines and had significant improvements in trial durations when compared to *IndustReal*. During experiments, we observed FORGE led to gentler interactions between the parts (see accompanying video). The reduced force produced by this policy was especially helpful for the M16 Nut which was more susceptible to slipping than the peg or gear.

For FORGE, the main failure cases occurred when there was high position estimation error (above 1σ of the training noise, see next section). Parts got stuck on each other (peg insertion) or the nut was rotated before alignment with the bolt, causing the threads to miss. However, we found training unstable with noise above $\sigma = 2.5\text{mm}$. Adopting a curriculum or adding additional sensing modalities (e.g., tactile sensors or wrist-cameras) may help address this.

B. Noise Analysis

We next aim to answer (**Q3**): **How is policy performance, in terms of success rate, affected by position-estimation error?** We use the same trials from the previous section, but

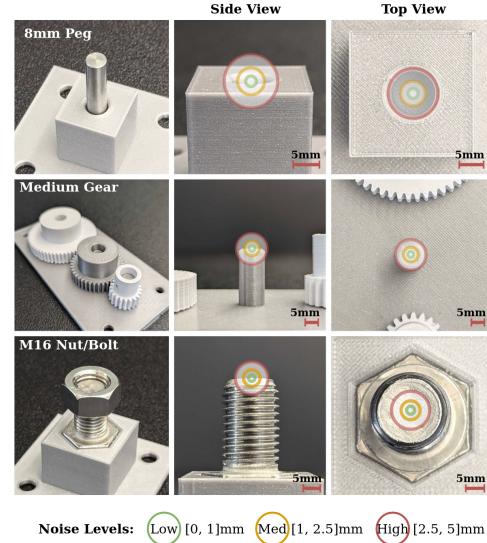


Fig. 3. **Perception Error** For each task, we visualize what the different position estimation errors look like overlaid on the fixed part.

show a breakdown of the results across different error levels. During each trial, artificial perception error was added to the fixed part’s position (calibrated as described in Section IV-A²). A third of the trials fell in each of the three considered error levels (see Fig. 3): Low (0-1mm), Medium (1-2.5mm), and High (2.5-5mm). We considered 3D position error by sampling a perturbation vector with a radius uniformly sampled in the desired error range and a direction uniformly sampled from the unit-sphere.

Figure 4 visualizes the performance of *IndustReal* vs.

²Adding artificial noise allows us to better characterize performance across error level compared to a perception system whose bias and variance can be difficult to estimate and control.

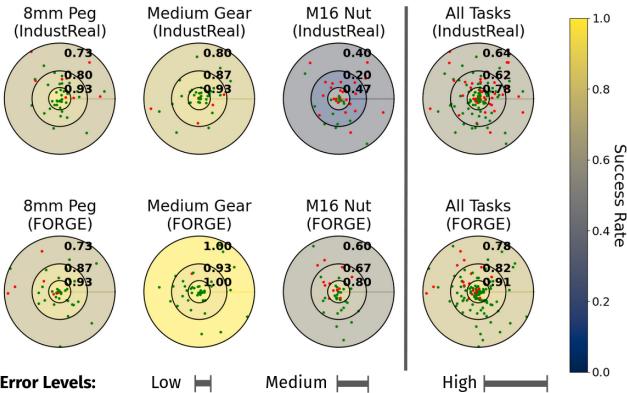


Fig. 4. **Noise Analysis** Performance broken down by level of position error. Each subplot is a planar representation of the error levels where each ring corresponds to low (0-1mm), medium (1-2.5mm), and high (2.5-5mm) error. Success rate, stated in black text, is also represented by the shade of the corresponding ring. Dots represent x-y noise samples for successful (green) and failed (red) trials. FORGE results in good performance across tasks even with high error levels.

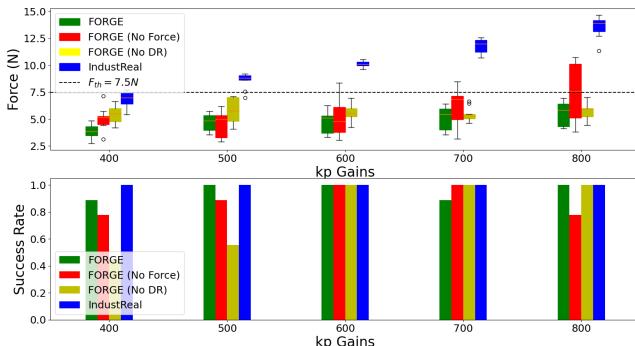


Fig. 5. **Gains Analysis** (180 trials, 8mm Peg) With force sensing, FORGE can achieve robust success rates (bottom) across varying controller gains at deployment time. Even with different gains, force sensing allows the policy to modulate its actions to achieve low contact forces (top).

FORGE policies at different noise levels. Each subplot is a 2D representation of how much x-y error there was for each trial (z-dimension error not visualized). Each point corresponds to either a successful (green) or unsuccessful (red) trial (only xy coordinates of the error vector are visualized as dots). The color of the ring represents the success rate at the corresponding error levels (increasing outwards).

Although performance is comparable for the peg insertion task, FORGE outperformed *IndustReal* for the gear meshing and nut threading tasks at all noise levels. This demonstrates that force is a useful modality to robustly recover from larger amounts of position estimation error. Performance generally degraded with error $> 2.5\text{mm}$ which is beyond 1σ of the observation noise added in simulation. With high error, the effects of contact are more pronounced because the robot may need to search longer before the task is complete.

C. Force Analysis

Next, we investigate how FORGE limits forceful interactions. **(Q4) How important is the excessive-force penalty for safe interactions? (Q5) Can FORGE limit the applied force without extensive controller tuning?**

Excessive-Force Penalty (Q4): In Table I, we compare to an ablation, *No FP*, that was trained without the excessive-

force penalty of FORGE (but still used force observations and dynamics randomization). We used the same evaluation procedure as for FORGE but deployed with two different controller gains (we chose values at the lower and middle of the gain randomization range). We found that policies deployed with the lower gains achieved similar average forces to FORGE while those deployed with higher gains naturally experienced more force. Both policies had lower success rates than FORGE which was deployed with controller gains at the middle of the randomization range.

Gains Robustness (Q5): To measure how robust FORGE is to controller gains, we performed an additional experiment where we varied the gains at deployment time and measured success rate. We compare FORGE to *IndustReal* and multiple ablations. The experiment was carried out for the 8mm peg task at a single workspace location, with medium position estimation error and limited initial-state randomization. We considered 5 proportional gain levels across the randomization range (corresponding to an $8N$ range in the maximum force the controller could apply) and each condition was evaluated 9 times (3 runs per checkpoint).

In Fig. 5, we see that FORGE achieves high success rates while respecting the force threshold across a wide range of controller gains. However, performance is less consistent without force observations or dynamics randomization. In Fig. 5 (top), we use a box plot to show the spread of F_{mean} across the 9 trials of each condition. The dotted line shows the deployment force-threshold: $F_{th} = 7.5\text{N}$. We see that when the force observation was included, contact force was consistently low across gains. However, without force observations, the spread of forces across episodes was high, often exceeding the threshold at higher gains. Similarly, the force exerted by *IndustReal* policies increased with controller gains. As *IndustReal* is not force-aware, achieving desired forceful properties requires tuning controller gains. Overall, these results highlight the importance of force sensing to enable the policy to effectively modulate the contact force.

D. Success Prediction Analysis

To evaluate success prediction, we ask: **(Q6) Does success prediction, trained in simulation, transfer to the real world? (Q7) Can success prediction be used to tune the force-threshold for tasks that require forceful insertion?**

Sim-to-Real Transfer (Q6): To measure the effectiveness of success prediction, we report additional metrics for each of the trials in Table I:

- Early Term. Precision: The fraction of early-terminated trials that were actually successful.
- Early Term. Recall: The fraction of successful trials which were terminated correctly with $a^{ET} > p_{term}$.
- Early Term. Delay (s): For successful episodes, how long after success occurred did the policy terminate.

Results show that success prediction transferred well to the real world. The termination method correctly identified successes (high precision and recall) and worked best when using force observations for all tasks. We also see that delay times are shortest when using force observations. This

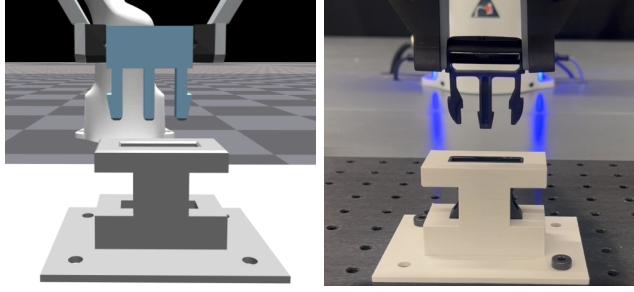


Fig. 6. Simulated (left) and real (right) parts for a snap-fit insertion task which requires $15N$ of force to succeed. See the video for policy execution.

shows the benefit of force for sensing task completion: when the gear has been fully meshed or the nut threads successfully engaged. Using success prediction also leads to shorter delays than *IndustReal* which uses a fixed duration.

Force Threshold Tuning (Q7): To evaluate the utility of *success prediction* for force threshold tuning, we introduced a new *snap-fit* task (see Fig. 6). In simulation, the snap-fit buckle was implemented with torsion springs for each of the clips. The stiffness of the springs was randomized to vary the amount of force needed for insertion. We also ensured the robot’s gains and force-threshold were randomized such that success was possible. In the real world, we used a snap-fit buckle that required $15N$ of force for insertion. We report real world results from running the automatic tuning procedure described in Sec. III-C. The initial force threshold was set to $7.5N$ and increased by $5N$ each policy execution until the policy predicted success. No initial state randomization or noise were added for these experiments.

Out of 10 trials for the complete tuning procedure (each consisting of up to 3 policy executions with increasing force thresholds), the tuning procedure succeeds 8 times while successful insertion occurred 10 times. The two failures were instances where the insertion succeeded, but the policy failed to predict success. Success occurred on the third execution 9/10 times (it once occurred on the second trial), meaning the policy generally respected the force threshold (success should not occur until the force threshold exceeds $15N$). Furthermore, of the 29 policy executions, the success prediction by the policy was correct 27 times.

We also evaluated the task using a sufficiently high threshold and on a larger initial state distribution, similar to what was used during training in simulation. Here the success rate dropped to 6/10, which we attribute to an unstable grasp leading to part slippage during contact. This reveals a limitation of having a single force threshold: for certain tasks, the optimal force threshold may vary depending on the phase of the task. For example, low forces are required until the buckle is aligned with the socket, only then is it safe to use high forces.

E. Multi-Stage Assembly

To culminate this work, we show that FORGE enables the multi-stage assembly of a planetary gearbox using a simple perception system (see Fig. 7 for the initial and final states). We assume the assembly sequence is known *a priori* and

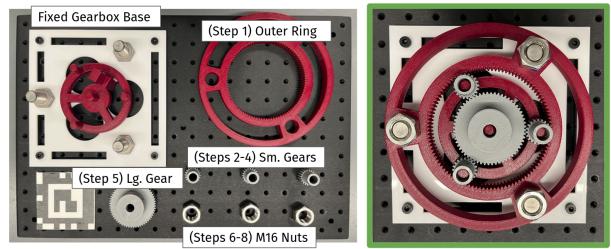


Fig. 7. FORGE policies enable a robot to complete long-horizon tasks such as assembling a planetary gearbox (from initial state [left] to goal state [right, enlarged]).

train FORGE policies for *Small Gear*, *Large Gear*, and *M16 Nut* tasks. We additionally introduce a new *Ring Insertion* task, which must also be robust to orientation estimation noise such that the three bolts align with the holes in the outer ring. Successfully assembling the planetary gearbox requires executing 8 contact-rich primitives.

We ran 5 trials resulting in the following success rates: Ring Insertion (5/5), Small Gear (15/15), Large Gear (3/5), M16 Nut (15/15). Early terminations saved on average 65s in a single trial compared to executing policies for a fixed duration. Overall, the complete assembly succeeded in 3/5 trials where the failures correspond to the large gear insertion (which has to align the teeth of three already inserted small gears). Please see the accompanying video for a demonstration of the multi-stage assembly and the website for more experimental details.

VI. RELATED WORK

Assembly tasks typically involve mating parts with tight clearances and detailed geometries [32], [33]. Various approaches have been proposed to handle pose uncertainty in such tasks. Mechanically, *remote centers of compliance* [34] or chamfers can mitigate small misalignments. Compliant control [35] and strategies such as spiral search [11], [36] have also be used for insertion. These strategies typically consider low noise levels and are task-specific.

Sim-to-Real Transfer: System identification [23] is often time-consuming and difficult to apply to contact-rich tasks. Instead, *dynamics randomization* randomizes parameters such as part friction/stiffness [10], [24], [25], [26], [27], controller gains [12], [24], or F/T observation scale [12], [15]. Even with randomization, excessive forces can occur when deployed. An expert can tune the controller gains at deployment or choose an action-space that is safe by design [7], [29], [37]. Gains can also be adapted online via optimization [14] or an explicit *gain-tuning* model [15].

Similar to FORGE, other works have proposed to use a force-threshold [10], [20], [27]. 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, [10] 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 [17] 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. Additional discussion of related works can be found on the website.

Success Prediction: Previous *sim-to-real* approaches execute policies for a fixed duration [7]. Instead, we would like to terminate once success is achieved. For some tasks, success can be manually specified from sensor data [38], [39]. For others, a classifier can be learned from visual data [40], [41]. However, for contact-rich tasks, visual and proprioceptive data alone may be insufficient to determine success [42]. In such cases, the robot can execute actions to verify success [43]. Previous work learns a separate policy to check success *after* task execution [16]. Instead, we jointly trained a policy to predict success *during* task execution.

VII. CONCLUSION

In conclusion, we present FORGE, a force-aware method to train robust sim-to-real policies with pose estimation uncertainty. FORGE uses a force threshold and dynamics randomization to learn *safe* exploration behaviours, enabling successful policy execution with up to 5mm of position estimation error. In addition, FORGE can predict task success, allowing efficient policy execution and force threshold tuning. In future work, we plan to investigate torque sensing for more efficient search strategies. We also believe research in *real-to-sim* will help automatically tune simulation models for more adaptive behaviours.

ACKNOWLEDGMENT

The authors thank the Seattle Robotics Lab and the Robust Robotics Group for their valuable feedback.

REFERENCES

- [1] I. Akkaya *et al.*, “Solving rubik’s cube with a robot hand,” *arXiv:1910.07113*, 2019.
- [2] A. Handa *et al.*, “DeXtreme: Transfer of Agile In-hand Manipulation from Simulation to Reality,” in *ICRA*. IEEE, 2023.
- [3] J. Tan *et al.*, “Sim-to-Real: Learning Agile Locomotion For Quadruped Robots,” in *RSS*, 2018.
- [4] J. Hwangbo *et al.*, “Learning agile and dynamic motor skills for legged robots,” *Science Robotics*, 2019.
- [5] Y. Narang *et al.*, “Factory: Fast Contact for Robotic Assembly,” in *RSS*, 2022.
- [6] 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.
- [7] B. Tang *et al.*, “IndustReal: Transferring Contact-Rich Assembly Tasks from Simulation to Reality,” in *RSS*, 2023.
- [8] G. Schoettler and et al., “Meta-reinforcement learning for robotic industrial insertion tasks,” in *IROS*. IEEE, 2020.
- [9] S. Kozlovsky, E. Newman, and M. Zackenhouse, “Reinforcement Learning of Impedance Policies for Peg-in-Hole Tasks: Role of Asymmetric Matrices,” *IEEE RA-L*, 2022.
- [10] 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.
- [11] S. Chhatpar and M. Branicky, “Search strategies for peg-in-hole assemblies with position uncertainty,” in *IROS*. IEEE, 2001.
- [12] S. Jin, X. Zhu, C. Wang, and M. Tomizuka, “Contact Pose Identification for Peg-in-Hole Assembly under Uncertainties,” in *ACC*. IEEE, 2021.
- [13] K. Van Wyk, M. Culleton, J. Falco, and K. Kelly, “Comparative peg-in-hole testing of a force-based manipulation controlled robotic hand,” *IEEE T-RO*, 2018.
- [14] 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.
- [15] X. Zhang, M. Tomizuka, and H. Li, “Bridging the Sim-to-Real Gap with Dynamic Compliance Tuning for Industrial Insertion,” in *ICRA*. IEEE, 2024.
- [16] K. Huang, E. Hu, and D. Jayaraman, “Training Robots to Evaluate Robots: Example-Based Interactive Reward Functions for Policy Learning,” in *CORL*, 2022.
- [17] D. Son, H. Yang, and D. Lee, “Sim-to-Real Transfer of Bolting Tasks with Tight Tolerance,” in *IROS*. IEEE, 2020.
- [18] L. Kaelbling, M. Littman, and A. Cassandra, “Planning and acting in partially observable stochastic domains,” *Artificial intelligence*, 1998.
- [19] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, “TRANSIC: Sim-to-Real Policy Transfer by Learning from Online Correction,” *arXiv:2405.10315*, 2024.
- [20] 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.
- [21] V. Makoviychuk *et al.*, “Isaac gym: High performance gpu-based physics simulation for robot learning,” *arXiv:2108.10470*, 2021.
- [22] A. Allshire *et al.*, “Transferring dexterous manipulation from gpu simulation to a remote real-world trifinger,” in *IROS*. IEEE, 2022.
- [23] L. Ljung, “System identification,” in *Signal analysis and prediction*. Springer, 1998, pp. 163–173.
- [24] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, “Sim-to-Real Transfer of Robotic Control with Dynamics Randomization,” in *ICRA*. IEEE, 2018.
- [25] O. Spector and M. Zackenhouse, “Learning Contact-Rich Assembly Skills Using Residual Admittance Policy,” in *IROS*. IEEE, 2021.
- [26] A. Apolinarska *et al.*, “Robotic assembly of timber joints using reinforcement learning,” *Automation in Construction*, 2021.
- [27] M. Hebecker, J. Lambrecht, and M. Schmitz, “Towards Real-World Force-Sensitive Robotic Assembly through Deep Reinforcement Learning in Simulations,” in *AIM*. IEEE, 2021.
- [28] R. Petrea, M. Bertoni, and R. Oboe, “On the Interaction Force Sensing Accuracy Of Franka Emika Panda Robot,” in *IECON*. IEEE, 2021.
- [29] K. Zhang, M. Sharma, J. Liang, and O. Kroemer, “A modular robotic arm control stack for research,” *arXiv:2011.02398*, 2020.
- [30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv:1707.06347*, 2017.
- [31] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, “Asymmetric actor critic for image-based robot learning,” in *RSS*, 2018.
- [32] J. Xu, Z. Hou, Z. Liu, and H. Qiao, “Compare contact model-based control and contact model-free learning,” *arXiv:1904.05240*, 2019.
- [33] Z. Jia, A. Bhatia, R. Aronson, D. Bourne, and M. Mason, “A survey of automated threaded fastening,” *IEEE T-ASE*, 2018.
- [34] S. H. Drake, “Using compliance in lieu of sensory feedback for automatic assembly.” Ph.D. dissertation, MIT, 1978.
- [35] T. Lozano-Perez, M. Mason, and R. Taylor, “Automatic synthesis of fine-motion strategies for robots,” *IJRR*, 1984.
- [36] W. Newman, Y. Zhao, and Y. Pao, “Interpretation of force and moment signals for compliant peg-in-hole assembly,” in *ICRA*. IEEE, 2001.
- [37] N. Vuong, H. Pham, and Q. Pham, “Learning Sequences of Manipulation Primitives for Robotic Assembly,” in *ICRA*. IEEE, 2021.
- [38] L. Pinto and A. Gupta, “Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours,” in *ICRA*. IEEE, 2016.
- [39] B. Wen, W. Lian, K. Bekris, and S. Schaal, “You only demonstrate once: Category-level manipulation from single visual demonstration,” *RSS*, 2022.
- [40] Z. Su, O. Kroemer, G. Loeb, G. Sukhatme, and S. Schaal, “Learning manipulation graphs from demonstrations using multimodal sensory signals,” in *ICRA*. IEEE, 2018.
- [41] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine, “Variational inverse control with events: A general framework for data-driven reward definition,” *NeurIPS*, 2018.
- [42] A. Rodriguez and et al., “Failure detection in assembly: Force signature analysis,” in *IEEE CASE*, 2010.
- [43] O. Kroemer, S. Niekum, and G. Konidaris, “A review of robot learning for manipulation,” *JMLR*, 2021.