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

Michael Noseworthy[1] **Bingjie Tang**[2] **Bowen Wen**[3] **Ankur Handa**[3] **Chad Kessens**[4]

Nicholas Roy[1] **Dieter Fox**[3] **Fabio Ramos**[3] **Yashraj Narang**[3] **Iretiayo Akinola**[3]

Abstract: We present FORGE, a method that enables sim-to-real transfer of contact-rich manipulation policies in the presence of significant pose uncertainty. FORGE combines a *force threshold* mechanism with a *dynamics randomization* scheme during policy learning in simulation to enable the robust transfer of the learned policies to the real robot. At deployment, FORGE policies adaptively perform contact-rich tasks while respecting the specified force threshold, regardless of the controller gains. Additionally, FORGE autonomously predicts a termination action once the task has succeeded. We demonstrate that FORGE can be used to learn a variety of robust contact-rich policies (nut-threading, insertion, and gear meshing), enabling multi-stage assembly of a planetary gear system.Project website: https://noseworm.github.io/forge/

Keywords: Assembly, Sim-to-Real, Force Sensing

1 Introduction

We are interested in *sim-to-real* techniques for learning assembly primitives (e.g., low-clearance insertion or nut-threading). Over the past decade, work in simulation and sim-to-real has led to advances in challenging areas such as dexterous manipulation and legged locomotion [1, 2, 3, 4]. However, similar results have only recently been achieved for assembly, which requires efficient and accurate simulation of both the robot and the detailed, low-clearance parts [5, 6, 7, 8, 9, 10].

Even with these advances, successfully deploying sim-to-real policies for assembly remains challenging. Previous approaches typically consider small amounts of perceptual noise. This assumption aligns with industrial robot workcells where uncertainty is typically engineered away. Strategies to deal with uncertainty include mechanical design of fixtures and adapters, extensive calibration processes, and the use of high-precision sensing. We aim to develop control methods that are robust to higher levels of pose estimation error, which is unavoidable in less structured environments.

When there is pose uncertainty, behaviours that search for and rely on contact can be used to ensure success [11, 12]. However, the required contact between the parts can lead to undesirable outcomes if the force is too high. Parts can slip or become damaged, making the task difficult or impossible to complete. Heuristic approaches, such as spiral search [11, 13] are task-specific and can be inefficient. RL offers a general paradigm for developing more flexible search behaviours. However, the sim-to-real gap makes it difficult to transfer policies learned in simulation to the real world. Even if the simulator has an accurate robot model (itself a time-consuming calibration procedure), it is difficult to know *a priori* the material and inertial properties of the parts the robot will interact with.

In this work, we propose FORGE: a framework for developing sim-to-real policies that safely and efficiently perform assembly tasks in the presence of significant pose uncertainty. FORGE trains policies in simulation that are *robust* to a wide range of contact interactions. Additionally, policies are trained without precise knowledge of part poses, leading to emergent search behaviours.

Correspondence: mnosew@mit.edu
[1]MIT [2]USC [3]NVIDIA [4]DEVCOM Army Research Laboratory (ARL)

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).

As assembly policies become more performant, we emphasize the importance of reporting metrics beyond *success rate*, such as *mean force* or *time to success*. Standard practice in *sim-to-real* assembly is to execute policies for a fixed duration [7]. However, due to task variation, this will often lead to premature termination or leave a robot "waiting" after the task is finished. Instead, the policy can determine when to terminate. This is itself a difficult task that can benefit from contact (e.g., a successfully inserted peg cannot move laterally). FORGE proposes a method for early termination that expands the action space so that the policy learns to predict task success. We show that early termination, trained in simulation, robustly transfers to the real world.

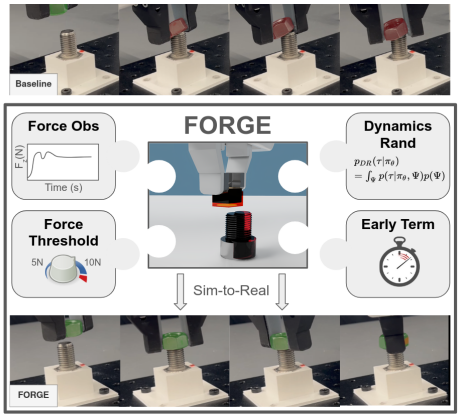

Figure 1: FORGE uses force feedback to learn search behaviours for contact-rich tasks with pose estimation uncertainty. It combines *dynamics randomization*, a *force threshold*, and *early termination* for robust sim-to-real transfer. The resulting policies are *safer* (bottom) compared to aggressive baseline policies that cause parts to slip (top).

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 pose estimation error (up to $5mm$). **(2) A dynamics randomization scheme** that enables robust sim-to-real transfer, and minimizes the need to tune controller gains. **(3) A method for early termination prediction** that allows efficient policy execution. **(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 over > 500 real-world trials.

2 RL for Contact-Rich Assembly

We are interested in tasks with tight tolerances and detailed geometry. Each task involves mating a part held in the gripper to a part fixed to the workspace. 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).

Peg Insertion: A small round peg with $8mm$ diameter needs to be inserted into a corresponding socket with $0.5mm$ diametrical clearance. There is position uncertainty such that a successful search behaviour requires lateral exploration.

Gear Meshing: Gears need to be inserted onto pegs with $0.5mm$ clearance. Other gears are present and the teeth of adjacent gears must be aligned for successful meshing. In addition to lateral exploration, rotational exploration may be necessary to mesh the teeth.

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 oriented[1] such that success can be achieved with a single revolution of the wrist joint. We consider nuts with a relatively small size (M16) compared to previous sim-to-real work (M48) [14]. A successful behaviour will resolve lateral uncertainty and place the nut on the bolt before rotating the wrist (otherwise the threads may not catch).

[1] We leave the more challenging scenario involving completely unobserved thread orientation to future work.

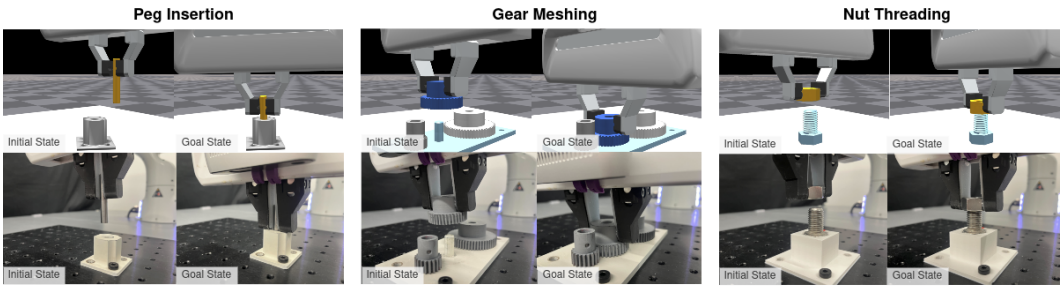

Figure 2: FORGE is evaluated on three tasks proposed in *Factory* [5]: Peg Insertion, Gear Meshing, and Nut Threading. Each task is trained in simulation and transferred directly to the real robot.

2.1 POMDP Formulation

We formulate our problem as a *Partially Observable Markov Decision Process* (*POMDP*) [15, 16] to reflect the partial observability of most contact-rich manipulation setups. The goal is to learn a parameterized policy, $\pi_\theta(a_t|o_1, \ldots, o_t)$, that maximizes the expected return:

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

where $\tau = (s_0, a_0, o_0, s_1, a_1, o_1, \ldots)$ is the trajectory of states, actions, and observations resulting from the robot following policy π_θ.

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): It is difficult to accurately estimate the full state of small parts. Instead, policies use the following observations: 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 without tactile sensing. Likewise, we do not observe Ψ, but include the previous action, a_{t-1}, to help infer unknown dynamics.

Actions ($\mathcal{A}$): Control targets for a task-space impedance controller [17, 7]. 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): T is parameterized by the dynamics parameters, Ψ: $T_\Psi : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ and specified using a simulator (in our case *IsaacGym* [18]) with the corresponding set of simulation parameters, Ψ^{sim}. The sim-to-real gap comes from the mismatch between Ψ^{sim} and Ψ^{real}.

Observation Function (O): The observation function generates noisy observations from state: $O : \mathcal{S} \times \mathcal{A} \to \Omega$. The position of the fixed part is assumed to have up to $5mm$ error. Independent Gaussian noise is added to each of the other observations at every timestep (see Appendix A).

Reward (R): The reward function, $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, uses a keypoint formulation as its main component: $R_{kp}(p^{fixed}, p_t^{held})$. The target keypoints, k^{targ}, represent the desired position of the held part, while k_t^{held} represent its current position. We modify the keypoint reward from previous work [5, 19] to account for small, threaded geometries (see Appendix B for more details).

3 FORGE: Robust Search under Uncertainty

FORGE uses on-policy RL to learn exploratory behaviours in simulation. A *force threshold* (Section 3.1) and *dynamics randomization* (Section 3.2) are introduced to achieve robust search behaviours. FORGE also introduces an *early termination* procedure (Section 3.3) for efficient execution.

3.1 Force Threshold

During policy execution, excessive force can cause parts to slip or become damaged (e.g., electronic components with fragile pins). 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.

To develop *safe* policies, 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}, experienced by the arm exceeds the threshold. Concretely, we add an additional term to the reward function:

$$R_{contact\_pen}(F_t^{ee}) = -\beta * \max(0, ||F_t^{ee}|| - F_{th}). \tag{2}$$

In simulation, the true contact force can be measured. Note that this penalty can be used during training whether the policy has access to the force observation or not.

3.2 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*. To gain insight into why minimizing the gap is important, specifically for contact-rich tasks, we can look more deeply into how trajectories are sampled:

$$\tau \sim p(\tau|\pi, \Psi) = p(s_0) \prod_{t=1}^{T} \Big[\pi(a_t|o_{1:t}) p(o_t|s_t, a_{t-1}, \Psi) p(s_t|s_{t-1}, a_{t-1}, \Psi)\Big]. \tag{3}$$

From this equation, we see the dynamics parameters can impact both the next-state and observation distributions. For the same action, different dynamics parameters can lead to parts being in different locations. Further, similar actions may lead to different observed contact forces.

The *sim-to-real* gap is usually handled by (1) system identification (Sys-ID) [20] or (2) dynamics randomization (DR) [21, 10]. 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:

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

The integral is approximated with samples from a randomization distribution (see Appendix A).

Controller Randomization: The controller has a large impact on what force will be experienced. This work uses impedance-control where applied forces are computed as:

$$p_t^{targ} = clip(combine(a_t, p^{fixed}), \lambda) \qquad F^{targ} = k_p(p_t^{targ} - p_t^{ee}) - k_d v_t^{ee}. \tag{5}$$

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, 22, 23]. 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 [23] can be costly and complex. Randomization has the benefit that the policy is robust to a range of control parameters, greatly simplifying deployment.

Part Friction Randomization: As parts slide against each other, the material friction determines how much lateral force the sensor will experience. To ensure policies can work across a range of materials, we randomize part friction.

	8mm Peg		Medium Gear		M16 Nut	
	Success Rate ↑	Duration (s) ↓	Success Rate ↑	Duration (s) ↓	Success Rate ↑	Duration (s) ↓
FORGE	**0.84** (0.05)	5.01 (0.17)	**0.98** (0.02)	**6.34** (0.42)	0.44 (0.07)	24.50 (1.35)
FORGE (No Force)	**0.82** (0.06)	7.30 (0.42)	**0.93** (0.04)	9.02 (0.79)	**0.69** (0.07)	**13.16** (0.66)
No FP (400kp)	0.64 (0.07)	6.06 (0.40)	0.82 (0.06)	**6.44** (0.23)	N/A	N/A
No FP (600kp)	0.71 (0.07)	5.28 (0.35)	0.73 (0.07)	**6.99** (0.46)	N/A	N/A
Baseline	0.64 (0.07)	5.09 (0.30)	0.69 (0.07)	7.57 (0.50)	0.20 (0.06)	27.89 (2.22)

Table 1: **Baseline Comparison** FORGE (with and without force observations) is compared to baselines that do not include robust sim-to-real components. It is additionally compared to ablations that do not use an excessive-force penalty. Evaluations are performed over a total of 585 trials on the real robot (45 per row). Standard errors are included in parentheses.

Robot Dynamics Randomization: Due to phenomena such as joint friction, 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.

3.3 Early Termination

Ideally, we want the policy to terminate as soon as the task has succeeded and no sooner. Although success is clearly defined in simulation where we have access to the positions of each part, it is difficult to reliably predict success in the real world [24]. Consider the nut-threading task, where the distance between a successfully threaded nut and a loose nut may be as small as a millimeter.

We propose to train a success predictor which can robustly transfer from sim-to-real and be used to make early termination decisions. 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 reward, R_t^{ET}, which penalizes incorrect success predictions: $R_t^{ET}(a_t, y_t) = -|a_t^{ET} - y_t|$, where y_t is the true success label at time t. During training, episodes are always executed for the maximum length.

At deployment time, a confidence threshold, p_{term}, can be used to terminate the episode as soon as the policy believes it has succeeded: $a_t^{ET} > p_{term}$. This allows us to behave efficiently, a desirable property for industrial applications where it is important to reduce cycle times.

4 Results and Discussion

This section focuses on evaluating sim-to-real transfer. We trained all our policies in *IssacGym* and deploy them on a real Franka Panda robot. Additional details on the setup are in Appendix D. Additional analysis on the proposed early-termination procedure is in Appendix E.

4.1 Baseline Comparisons

We first compare FORGE to a baseline method that does not include any FORGE components; however, it was trained with the early termination procedure so meaningful episode durations could be reported. We also considered a version of FORGE that does not use force observations. The questions we seek to answer are: **(Q1) Does FORGE lead to more robust sim-to-real transfer? (Q2) Does FORGE lead to policies with more desirable behavioural properties?**

Each reported metric represents 45 trials spread across 5 workspace locations for the fixed part, and 3 pose-estimation error levels ranging from $0 - 5mm$ (see Fig. 3). Similar randomization ranges

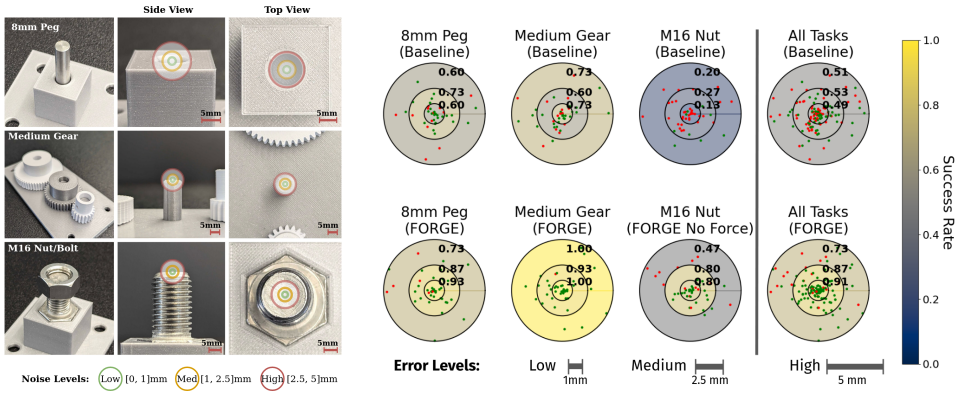

Figure 3: **Noise Analysis [Left]:** For each task, we visualize what the different pose estimation errors look like overlaid on the fixed part. **[Right]:** Performance broken down by level of pose error. Each subplot is a planar representation of the error levels where each ring corresponds to low (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.

were used as in simulation except for the in-hand part randomization where the part was placed centrally in the gripper. Results are reported in Table 1.

One conclusion for Q1 is that FORGE outperformed the *Baseline* method for all tasks whether force is included or excluded from the observation space. This suggests that the primary benefits of FORGE come from the dynamics randomization and excessive-force penalty. For FORGE, comparison to the policy without force observation shows that although using force sensing was useful for the easier insertion and gear meshing tasks, it harmed performance for the nut-threading task. We hypothesize that this is because nut-threading policies rely more on force observations in simulation and would therefore be more sensitive to any sim-to-real gap for this observation modality.

We include the analysis for Q2 in Appendix E which shows that FORGE used less force than the baseline and had minor improvements in trial duration.

4.2 Noise Analysis

We next aim to answer **(Q3): How is policy performance, in terms of success rate, affected by pose-estimation error?** We use the same trials from the previous section, but show a breakdown of the results across different error levels. During each trial, artificial perception error was added to the fixed part's pose.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 3 visualizes the performance of the baseline policy vs. FORGE 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. The color of the ring represents the success rate at the corresponding error levels (increasing outwards). For the M16 Nut task, we include results for the *No Force* ablation as this was the most robust policy that used dynamics randomization and a force threshold.

FORGE achieved high success rates (> 0.8) for all tasks at low and medium error levels, even for the M16 nut. Although performance degraded with error $> 2.5mm$, FORGE still significantly outperformed the baseline. With high error, the effects of contact are more pronounced because the robot may need to search longer before the task is complete.

4.3 Force Analysis

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

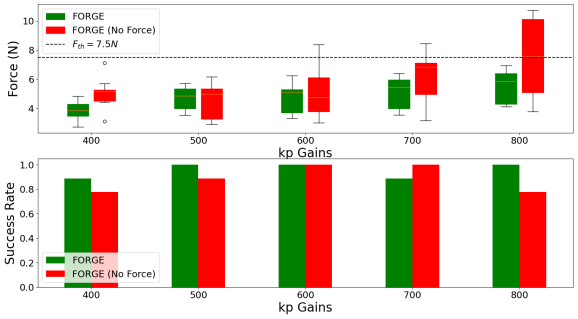

Figure 4: **Gains Analysis** (90 trials, 8mm Peg) With force sensing (+Force), FORGE can achieve robust success rates (bottom) across varying controller gains at deployment time. Even with different gains, force sensing allows the policy to modulate its actions to achieve low contact forces (top).

Gains Robustness (Q4): 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 compared FORGE and the *No Force* ablation to gain insight into how important force sensing is to limiting applied forces. The experiment was carried out for the $8mm$ peg task at a single workspace location, with medium pose 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. 4 (bottom), we see that FORGE achieves high success rates across a wide range of controller gains. However, performance is less consistent without force observations. In Fig. 4 (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: $F_{th} = 7.5N$. 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. This highlights the importance of force sensing to enable the policy to effectively modulate the contact force.

We answer Q5 in Appendix E which shows that policies without the excessive-force penalty led to higher contact forces and lower success rates.

4.4 Multi-Stage Assembly

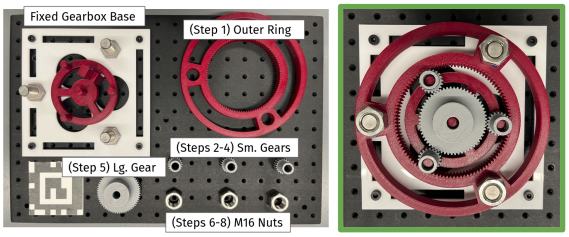

Figure 5: FORGE policies enable a robot to complete long-horizon tasks such as assembling a planetary gearbox (from initial state [left] to goal state [right]).

To culminate this work, we show that FORGE enables the multi-stage assembly of a planetary gearbox using a simple perception system (see Fig. 5 for the initial and final states). We assume the assembly sequence is known *a priori* and 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 Appendix C for more experimental details.

5 Related Work

Assembly tasks typically involve mating parts with tight clearances and detailed geometries [25, 26]. Various approaches have been proposed to handle pose uncertainty in such tasks. Mechanically, *remote centers of compliance* [27] or chamfers can mitigate small misalignments. Compliant control [28] and strategies such as spiral search [11, 29] have also be used for insertion. These strategies typically consider low noise levels and are task-specific.

Real World Reinforcement Learning Learning 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 [30, 31, 32, 33, 34] or using model-based approaches [35, 36, 37, 38]. To ensure excessive forces are not exceeded during training, these papers typically use control methods designed to be safe [34, 39, 40].

Sim-to-Real Transfer: 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 [41, 42, 43, 18], there has been much interest in *sim-to-real* transfer for complex control problems. Of note include legged locomotion [4, 44, 45, 46] and in-hand manipulation [19, 1, 2]. Recent advances in contact-rich simulation has enabled efficient simulation of assembly tasks [47, 48, 5, 6].

Although *system identification* is a principled approach to minimize the *sim-to-real* gap [20], it is often time-consuming and difficult to apply to contact-rich tasks [49, 50]. Instead, *dynamics randomization* randomizes parameters such as part friction/stiffness [10, 21, 22, 51, 52], controller gains [12, 21], or F/T observation scale [12, 53]. 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, 54, 55]. Gains can also be adapted online via optimization [23] or an explicit *gain-tuning* model [53].

Similar to FORGE, other works have proposed to use a force-threshold [10, 17, 52]. 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.

Most prior works 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 [14] 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 early termination procedures.

Early-Termination: 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 [56, 57]. For others, a classifier can be learned from visual data [58, 59]. However, for contact-rich tasks, visual and proprioceptive data alone may be insufficient to determine success [60]. In such cases, the robot can execute actions to verify success [61]. Previous work learns a separate policy to check success *after* task execution [24]. Instead, we jointly trained a policy to predict success *during* task execution.

6 Conclusion

In conclusion, we present FORGE, a method to train sim-to-real policies for robust execution 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 early termination, allowing efficient policy execution. In future work, we plan to investigate torque sensing to help develop more efficient search strategies. We also believe research in *real-to-sim* will help automatically tune simulation models for robust transfer in complicated tasks such as nut-threading.

Acknowledgments

We would like to thank the Seattle Robotics Lab and the Robust Robotics Group for valuable feedback and discussion.

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

[15] L. Kaelbling, M. Littman, and A. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 1998.

[16] 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.

[17] 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.

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

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

[20] L. Ljung. System identification. In *Signal analysis and prediction*, pages 163–173. Springer, 1998.

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

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

[23] 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.

[24] K. Huang, E. Hu, and D. Jayaraman. Training Robots to Evaluate Robots: Example-Based Interactive Reward Functions for Policy Learning. In *CORL*, 2022.

[25] J. Xu, Z. Hou, Z. Liu, and H. Qiao. Compare contact model-based control and contact model-free learning. *arXiv:1904.05240*, 2019.

[26] Z. Jia, A. Bhatia, R. Aronson, D. Bourne, and M. Mason. A survey of automated threaded fastening. *IEEE T-ASE*, 2018.

[27] S. H. Drake. *Using compliance in lieu of sensory feedback for automatic assembly.* PhD thesis, MIT, 1978.

[28] T. Lozano-Perez, M. Mason, and R. Taylor. Automatic synthesis of fine-motion strategies for robots. *IJRR*, 1984.

[29] W. Newman, Y. Zhao, and Y. Pao. Interpretation of force and moment signals for compliant peg-in-hole assembly. In *ICRA*. IEEE, 2001.

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

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

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

[37] 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.

[38] 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.

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

[40] 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.

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

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

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

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

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

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

[47] 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.

[48] 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.

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

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

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

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

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

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

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

[56] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *ICRA*. IEEE, 2016.

[57] B. Wen and et al. You only demonstrate once: Category-level manipulation from single visual demonstration. *RSS*, 2022.

[58] Z. Su, O. Kroemer, G. Loeb, G. Sukhatme, and S. Schaal. Learning manipulation graphs from demonstrations using multimodal sensory signals. In *ICRA*. IEEE, 2018.

[59] 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.

[60] A. Rodriguez and et al. Failure detection in assembly: Force signature analysis. In *IEEE CASE*, 2010.

[61] O. Kroemer, S. Niekum, and G. Konidaris. A review of robot learning for manipulation. *JMLR*, 2021.

[62] R. Petrea, M. Bertoni, and R. Oboe. On the Interaction Force Sensing Accuracy Of Franka Emika Panda Robot. In *IECON*. IEEE, 2021. doi:10.1109/IECON48115.2021.9589424.

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

[64] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel. Asymmetric actor critic for image-based robot learning. In *RSS*, 2018.

A Randomization

All randomization ranges are reported in Table 2. 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).

B Reward

B.1 Keypoint Reward

Here we describe the keypoint reward in more details. Keypoint distance is calculated as: $d_t^{kp}(p_t^{held}, p^{fixed}) = ||k_t^{held} - k^{targ}||$. We use a logistic kernel as in [19] 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 2).

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 $< 1mm$. 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 the part. These are implemented using different parameters for the logistic kernel,

$$R_{kp}(p^{fixed}, p_t^{held}) = \mathcal{K}_{a^c,b^c}^{coarse}(d_t^{kp}) + \mathcal{K}_{a^f,b^f}^{fine}(d_t^{kp}). \tag{6}$$

Parameters for each task can be found in Table 2.

B.2 Task Success

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: $R_{bonus}(p^{fixed}, p_t^{held}) = \mathbb{I}_{place} + \mathbb{I}_{success}$. The relative z-position of bottom of the held part to the top of the fixed part is used to check each condition. We found the bonuses led to more robust learning when there is significant pose uncertainty.

Each task defines success based on the relative positions between the held and fixed parts (Table 2 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.

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 2: Simulation parameters used to train FORGE policies.

C 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 small and large gear meshing tasks had one abutting gear in simulation. 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 ($3/5$).

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. 6) and trained a policy using the FORGE framework. We assume there is small orientation error on the ring ($< 5^\circ$) 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: The M16 policy was chosen as the best policy from our main evaluation (FORGE No Force). All other policies were trained using the FORGE framework including force observations. We trained one policy per task without any additional checkpoint selection procedure. 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.

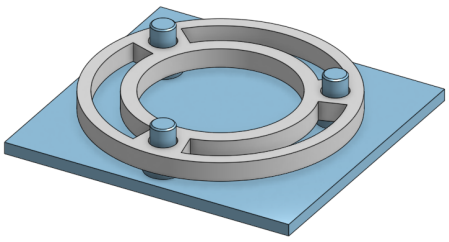

Figure 6: Simulated assets for the ring insertion task. The ring gear (grey) is inserted onto the gearbox plate (blue).

	Episode		Force		Early Termination	
8mm Peg	Success Rate ↑	Duration (s) ↓	F_{mean} (N) ↓	F_{max} (N) ↓	Precision ↑	Recall ↑
FORGE	**0.84** (0.05)	5.01 (0.17)	**5.51** (0.24)	12.84 (0.37)	**1.00** (0.0)	**1.00** (0.0)
FORGE (No Force)	**0.82** (0.06)	7.30 (0.42)	7.09 (0.35)	14.16 (0.39)	0.59 (0.08)	0.81 (0.07)
No FP (400kp)	0.64 (0.07)	6.06 (0.40)	6.94 (0.13)	**11.94** (0.24)	0.83 (0.07)	0.92 (0.05)
No FP (600kp)	0.71 (0.07)	5.28 (0.35)	10.66 (0.15)	16.58 (0.32)	0.91 (0.05)	**0.97** (0.03)
Baseline	0.64 (0.07)	5.09 (0.30)	11.81 (0.21)	17.93 (0.41)	**0.97** (0.03)	**1.00** (0.0)
Medium Gear	Success Rate ↑	Duration (s) ↓	F_{mean} (N) ↓	F_{max} (N) ↓	Precision ↑	Recall ↑
FORGE	**0.98** (0.02)	**6.34** (0.42)	7.95 (0.11)	15.10 (0.45)	0.95 (0.03)	**1.00** (0.0)
FORGE (No Force)	**0.93** (0.04)	9.02 (0.79)	8.49 (0.23)	14.68 (0.39)	0.60 (0.08)	**1.00** (0.0)
No FP (400kp)	0.82 (0.06)	**6.44** (0.23)	**6.52** (0.14)	**10.97** (0.24)	**1.00** (0.0)	**1.00** (0.0)
No FP (600kp)	0.73 (0.07)	**6.99** (0.46)	9.48 (0.23)	15.73 (0.30)	0.94 (0.04)	**0.97** (0.03)
Baseline	0.69 (0.07)	7.57 (0.50)	11.67 (0.45)	18.29 (0.40)	0.90 (0.05)	**0.97** (0.03)
M16 Nut	Success Rate ↑	Duration (s) ↓	F_{mean} (N) ↓	F_{max} (N) ↓	Precision ↑	Recall ↑
FORGE	0.44 (0.07)	24.50 (1.35)	**6.88** (0.14)	**13.34** (0.17)	0.50 (0.11)	**1.00** (0.00)
FORGE (No Force)	**0.69** (0.07)	**13.16** (0.66)	7.78 (0.17)	14.41 (0.28)	**1.00** (0.0)	1.00 (0.0)
Baseline	0.20 (0.06)	27.89 (2.22)	7.32 (0.23)	16.73 (0.29)	0.11 (0.10)	**1.00** (0.0)

Table 3: **Baseline Comparison** FORGE (with and without force observations) is compared to baselines that do not include robust sim-to-real components. It is additionally compared to ablations that do not use an excessive-force penalty. Evaluations are performed over a total of 585 trials on the real robot (45 per row). Standard errors are included in parentheses.

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 [7]. 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.

D Experimental Setup

Robot System: We use a *Franka Panda* robot and the *FrankaPy* [54] library for the impedance controller. Policies send targets to the controller at $15Hz$ while the controller operates at $1000Hz$. The Panda has joint-torque sensing, which can be projected to EE-frame force values [62].

For most experiments, we calibrate the poses of each fixed object and artificially add noise. This allows us to analyze performance under known levels of pose estimation error. Real experiments use the same initial state randomization as in simulation (see Appendix A). For our last experiment, we assemble a planetary gear box (Section 4.4) using the perception system from *IndustReal* [7] (retrained using data we collected). The perception errors in this system are largely caused by extrinsics calibration errors and minor bounding box prediction errors.

Simulator: All policies are trained using the *Factory* simulation methods within IsaacGym [5]. Noisy sensor values are used as policy input, whereas ground-truth sensor values are used to compute the excessive-force penalty. For all RL, we use recurrent PPO [63] with asymmetric actor-critic [64] to handle partial observability. More training details can be found in Appendix A.

Checkpoint Selection: For all tasks and models, we train three policies with separate random seeds. For the peg and gear tasks, all policies are deployed on the real-robot and reported results are averaged across the three policies. For the nut task, we found that not all policies transferred reliably to the real world, even when high success rates are achieved in simulation. As such, for this task, we report results for the *best* of the three checkpoints (determined using 18 test runs each).

Observation and Action Frames: For generalization across the workspace, we assume actions and observations are defined relative to the tip of the fixed part. The policy outputs a $4D$ relative transform from the tip of the fixed part (we assume upright parts). The policy output is bounded, which limits the operational volume of the end-effector (we allow targets to be up to $5cm$ away).

E Additional Results

Behavioural Properties (Q2): Does FORGE lead to policies with more desirable behavioural properties?

Along with success rate, the following metrics are reported to answer Q2 in Table 3:

- Duration (s): Average trial length using an early termination threshold of $p_{term} = 0.9$.
- $F_{mean}, F_{max}(N)$: Forces experienced by the robot.
- Early Term. Precision: Fraction of early-terminated trials that were successful.
- Early Term. Recall: Fraction of successful trials which were terminated correctly with $a^{ET} > p_{term}$.

Examining the behavioural metrics for Q2, we notice that FORGE used less force than the baseline and had minor improvements in trial duration. 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.

Excessive-Force Penalty (Q5): How important is the excessive-force penalty for safe interactions?

In Table 1, 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). Note that ablation results are not reported for nut-threading as we found that the nut always slipped out of the gripper. 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.

Success Prediction Analysis:

To evaluate the early termination procedure, we ask: **(Q6) Does success prediction, trained in simulation, transfer to the real world? (Q7) How much efficiency is gained when the policy determines when to terminate?**

To answer Q6, results in Table 3 show that early termination prediction transferred well to the real world. While the termination method tended to correctly identify successes for all models (high and often perfect recall), we see that precision was best when using force observations for the gear and peg tasks. This shows the benefit of force for sensing task completion: when the bottom of the socket has been reached, or the gear has been fully meshed.

To answer Q7, 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. 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. 7 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

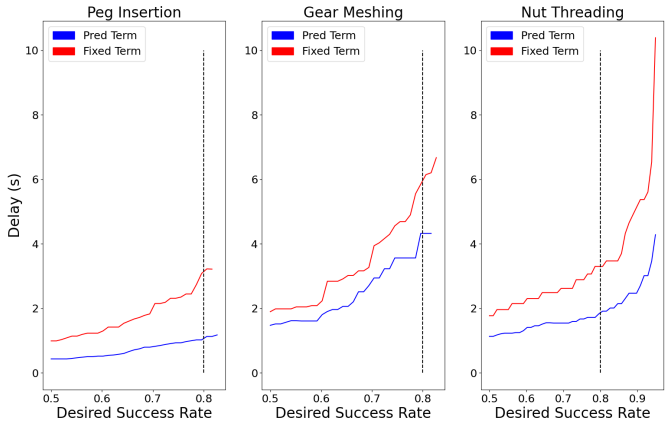

Figure 7: **Success Prediction Analysis** Relationship between Delay Time and Success Rate for two early termination methods (generated by varying each method's respective parameter: T or p_{term}). The *Pred Term* method leads to lower delays than the *Fixed Term* method, especially at higher success rates. The vertical line shows a 0.8 success rate.

by success rate and plotted.[2] 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.

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