2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Refinery: Active Fine-tuning and Deployment-time Optimization for Contact-Rich Policies

Bingjie Tang[1], Iretiayo Akinola[2], Jie Xu[2], Bowen Wen[2], Dieter Fox[3], Gaurav S. Sukhatme[1], Fabio Ramos[2,4], Abhishek Gupta[3], Yashraj Narang[2]
[1]University of Southern California, [2]NVIDIA Corporation, [3]University of Washington, [4]University of Sydney

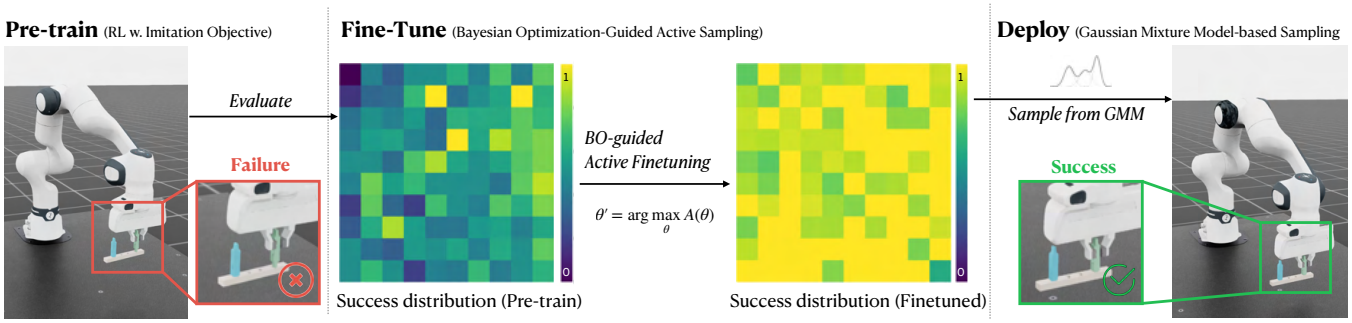

Fig. 1: **Method Overview**. We pretrain RL policies for assembly and evaluate them over initial conditions. (Here, a success map is shown over initial x- and y-coordinates). We leverage Bayesian Optimization-guided fine-tuning to improve policy performance. We then propose Gaussian Mixture Model-based sampling during deployment to select initializations that maximize execution success.

***Abstract*— Simulation-based learning has enabled policies for precise, contact-rich tasks (e.g., robotic assembly) to reach high success rates ($\sim$80%) under high levels of observation noise and control error. Although such performance may be sufficient for research applications, it falls short of industry standards and makes policy chaining exceptionally brittle. A key limitation is the high variance in individual policy performance across diverse initial conditions. We introduce** Refinery**, an effective framework that bridges this performance gap, robustifying policy performance across initial conditions. We propose Bayesian Optimization-guided fine-tuning to improve individual policies, and Gaussian Mixture Model-based sampling during deployment to select initializations that maximize execution success. Using** Refinery**, we improve mean success rates by** 10.98% **over state-of-the-art methods in simulation-based learning for robotic assembly, reaching** 91.51% **in simulation and comparable performance in the real world. Furthermore, we demonstrate that these fine-tuned policies can be chained to accomplish long-horizon, multi-part assembly—successfully assembling up to 8 parts without requiring explicit multi-step training. See our project website for more details.**

I. INTRODUCTION

Robotic assembly is a longstanding challenge, requiring high-precision, high-accuracy contact-rich manipulation, often over long horizons. Simulation and sim-to-real transfer have emerged as powerful strategies for tackling these challenges. Recent advances in simulation-based learning have enabled the development of robust assembly policies that achieve strong performance in both simulated and real environments [1–4]. Notably, these methods have achieved success rates of up to 80% on challenging benchmarks involving 2-part assemblies [5, 6]. Despite this progress, these results fall short of the requirements in industrial settings, where success rates of 95% or higher are typically expected, and 90% is considered a minimum threshold [7]. These standards are even more critical in multi-part assembly, as failure rates compound across sequential stages.

This leads to a natural question: how do the current state-of-the-art assembly policies fail? Our key observation is that a policy with a given average success rate in simulation does not fail with constant probability across all initial conditions (see Figure 1.); rather, it succeeds from certain initial states and fails from others, with high spatial variance. This observation suggests that minimizing the success rate variance among all initial conditions during learning and prioritizing high-performing ones during deployment could substantially improve overall task performance, potentially enabling long-horizon multi-part assembly as well.

Motivated by this insight, we introduce **Refinery**, a framework designed to enhance the fine-tuning and deployment of learned contact-rich policies. Our core idea is simple but effective: (1) actively identify and fine-tune policies on initial states with high uncertainty, and (2) prioritize high-success initializations during deployment. This approach significantly improves both individual and sequence-level success, moving research-grade policies closer to industry-grade reliability.

Specifically, our main contributions are: (1) **Active Sampling Method**: We introduce a Bayesian Optimization (BO)-guided fine-tuning strategy that significantly improves policy performance by focusing on high-uncertainty initial states; (2) **Deployment-time Optimization Method**: We propose a Gaussian Mixture Model (GMM)-based sampling strategy at deployment time that maximizes the success rate of already-trained or fine-tuned policies by prioritizing effective initial conditions; (3) **2-Part Assembly Results**: Using our approach, we achieve a 10.98% improvement in success rate on the simulated benchmark of 100 two-part assemblies from

[5], and we demonstrate zero-shot sim-to-real transfer with 97% success over 10 assemblies—exceeding prior state-of-the-art; (4) **Multi-part Assembly Results**: We provide 5 multi-part assemblies (derived from [8]) and corresponding simulation environments, and train assembly policies to solve these tasks. To our knowledge, this is the first work to demonstrate zero-shot sim-to-real for multiple long-horizon multi-part assemblies.

II. RELATED WORK

A. Robotic Assembly

Robotic assembly has historically been addressed with analytical methods [9–13]. Recent efforts have shifted toward learning-based approaches for greater adaptability and generalization. Model-based methods like GPS and iLQG offer sample efficiency but struggle with discontinuous contact dynamics [14–16]. Model-free, off-policy reinforcement learning (RL) methods (e.g., DQN, SAC, DDPG) improve sample reuse but suffer from convergence instability [17–19]. On-policy RL methods (e.g., PPO, TRPO, A3C) are more stable but sample-inefficient [20, 21]. Several works leverage demonstrations to improve data efficiency using residual learning, guided DDPG, DDPGfD, or offline RL [22–25]. However, collecting reliable demonstrations is often costly and time-consuming. Other non-RL methods include self-supervised learning from multimodal inputs or video-based imitation [26, 27]. While many methods achieve high success rates, they often rely on human input, long training times, or real-world resets, limiting scalability.

Over the past few years, advances in simulation accuracy and speed have enabled large-scale policy training for robotics, including contact-rich tasks like assembly [28–31]. There are few directly-comparable works for simulation-based assembly policies [5, 6, 32, 33]. Most focus on two-part assemblies or rely on human demonstrations, whereas our work leverages active sampling for diverse two-part and multi-part assembly tasks.

B. Improving Subtask Policies for Long-Horizon Tasks

Multi-part assembly can be decomposed into a sequence of two-part assembly subtasks, enabling multi-step policy chaining. Prior works address such chaining by adapting each sub-policy to the terminal states of its predecessor [34–37]. Extensions of this idea include backpropagation through the chain using goal regression in symbolic planning [38, 39] and bidirectional fine-tuning for long-horizon tasks [40, 41]. These approaches are most effective when terminal state distributions are broad. However, in tight-tolerance assembly, transitions between subtasks are narrowly distributed, making such chaining brittle and sequential fine-tuning costly.

Instead, our approach improves individual policy performance without explicitly optimizing inter-policy transitions. While on-policy RL suffers from high-variance gradient estimates due to random sampling, we mitigate this by leveraging active sampling during fine-tuning. Unlike exploration-focused strategies (e.g., entropy-regularized RL [42], GPS in high-reward regions [43], or Bayesian optimization for sample efficiency [44, 45]), our objective is to directly maximize task success.

III. PROBLEM DESCRIPTION

Goal: Our goal is to improve the overall performance of sequences of policies that solve challenging, long-horizon, contact-rich assembly tasks. Policy performance is highly sensitive to task initialization, as the tasks require high precision and accuracy. Hence, we study the problem of identifying and expanding the subsets of initial states for each individual policy that maximize the overall success rate for the sequence.

Assumptions: We make the following assumptions: (1) The assembly sequence is given, as is typical for industrial assembly workflows. (2) A CAD or mesh file is available for each part, as is typical for industrial assets. (3) All parts have size and initial pose such that at least one grasp is feasible and sufficient to allow subsequent insertion.

Problem Setup: Given a multi-part assembly problem consisting of N parts, the task is decomposed into $N-1$ sequential stages, where at each stage $i \in 1, ..., N-1$, a new part (*plug*) is inserted into the current partially-assembled structure. We first train a specialist policy π_i for each stage using the approach from [5], which uses on-policy RL with an imitation objective. We then apply the proposed fine-tuning and deployment-time optimization for each policy. [1]

Definitions: Each stage is formulated as a Markov decision process (MDP), where the agent is a simulated robot operating in an environment containing the parts to be assembled. We define a state space $\mathcal{S}$, observation space $\mathcal{O}$, and action space $\mathcal{A}$. The observation space includes robot arm joint angles, end-effector pose, goal pose, and the delta between the current and goal poses, and the robot action at each step is the delta between the current and next end-effector pose. Our state-transition dynamics is defined by $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, governed by rigid-body dynamics in our simulator. We define a randomized initial state distribution ρ_0, a reward function $R : \mathcal{S} \to \mathbb{R}$, and a discount factor $\gamma \in (0, 1]$. The episode consists of N timesteps, and the horizon is T, where $T \leq N$. The return G is defined as

$$G(T) = \mathbb{E}_\pi \left[\Sigma_{t=0}^{T-1} \gamma^t R(s_t) \right], \tag{1}$$

which represents the expected sum of discounted rewards over the horizon. The objective is to train a policy $\pi : \mathcal{O} \to \mathbb{P}(\mathcal{A})$ that maximizes this return.

For each stage i, we define a task-success evaluation function $J_i(\theta)$, where $\theta \in \mathbb{R}^3$ is the xyz position of the part relative to its goal and $J_i(\theta) \in [0, 1]$ represents the success probability of executing policy π_i starting from θ.

Dataset: We created a geometrically-diverse dataset of 5 multi-part assemblies based on [8] as shown in Figure 2. We design our selecting criteria to ensure sufficient complexity and compatibility with our setup. Specifically, each assembly must consist of 5-9 mechanical components and require 4-8

[1] We use the same hyperparameters, randomization ranges, observation noise levels and network structures for training policies with PPO as [5].

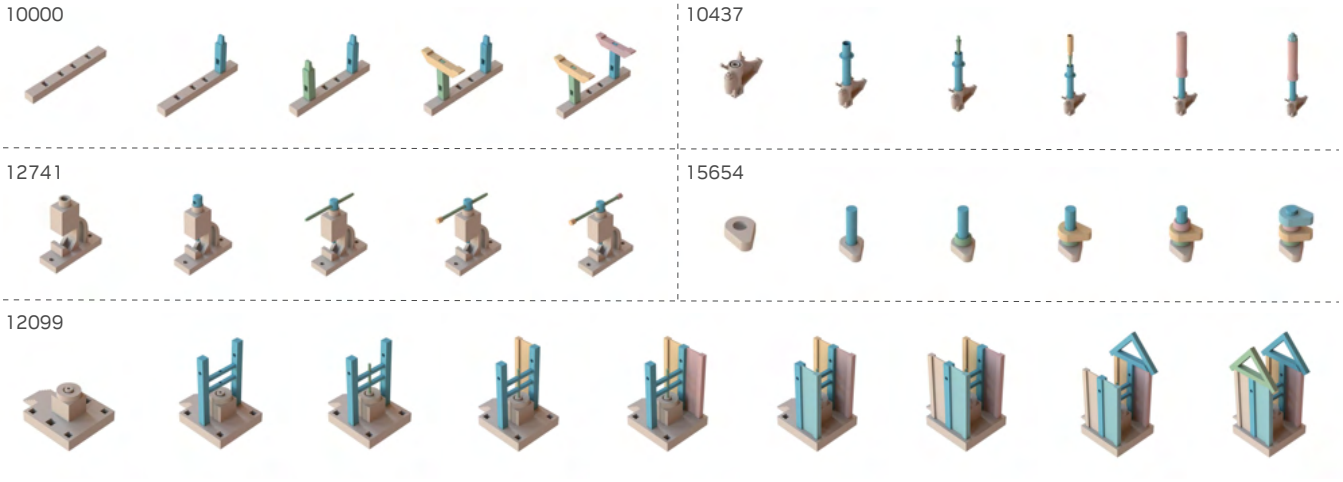

Fig. 2: **Multi-part Assembly Dataset.** We provide 5 multi-part assemblies derived from [8].

discrete insertion-based steps to complete. All components must be graspable using a parallel-jaw gripper. Furthermore, the complete assembly must be executable in a sequential manner by a single arm, such that each part can be added atop the previously assembled parts placed in a static fixture. For each assembly, we manually define a valid assembly sequence and designate the first component in this sequence as the *base*.

For selected assemblies, we apply a multi-step procedure that involves (1) scaling all component meshes to fit within a 10 cm^3 bounding box; (2) reorienting the assembly such that the *base* component is aligned with the z-axis in an upright configuration; (3) translating the assembly to ensure the bottom surface of the *base* is coplanar with the global origin; and (4) applying a depenetration step to enforce a 0.5 mm clearance between all parts. The resulting dataset is fully compatible with simulators that enforce non-penetration constraints [28] and is 3D-printable in the real world.

IV. Method

We present **Refinery**, a method that leverages the policy success rate distribution as a prior to achieve two main objectives. First, it enables efficient sampling of high-uncertainty regions in the task initialization space during policy fine-tuning, aiming to *improve every individual policy performance* (subsection IV-A). Second, at deployment time, it prioritizes promising initialization conditions to *maximize the overall success probability* of completing the full assembly task (subsection IV-B).

A. Bayesian Optimization-Guided Fine-Tuning

Given a sequence of policies π_i for each stage i of a multi-part assembly, our goal is to improve the overall sequence success rate by enhancing the robustness of individual policies through active fine-tuning. Before fine-tuning, we first train a specialist policy π_i for each stage with RL [5]. During fine-tuning, we aim to actively identify and prioritize task initial states with high uncertainty of success, in contrast with the uniform random sampling when training from scratch. However, the underlying success-rate distribution across initial states is typically unknown, non-differentiable, non-convex, and expensive to evaluate exhaustively. Moreover, this distribution evolves as policy learning progresses. To address these challenges, we adopt Bayesian Optimization (BO), a sample-efficient and gradient-free strategy for optimizing black-box functions. BO is particularly well-suited for our setting, as it enables informed exploration of the initialization space without requiring exhaustive evaluation.

In our formulation, for each stage i of a multi-part assembly, we model the success rate $\hat{J}_i$ of policy π_i as a function of θ, the initial position of the *plug* (i.e., the part to be inserted) using a Gaussian Process (GP):

$$\hat{J}_i(\theta) \sim \mathcal{GP}(\mu(\theta), k(\theta, \theta')) \tag{2}$$

where $\mu(\theta)$ is the mean function, and $k(\theta, \theta')$ is the covariance function that captures correlations between different initial states. To model $\hat{J}_i$, we execute the policy π_i from different randomly-sampled θ in simulation. [2]

After evaluation, BO then proposes the next sample θ' for fine-tuning by optimizing an acquisition function:

$$\theta' = \arg\max_{\theta} A(\theta) \tag{3}$$

where $A(\theta)$ is the selected acquisition function.

As described in Algorithm 1, this process is repeated iteratively to guide fine-tuning toward initializations most likely to improve individual policy robustness. In each iteration, we re-evaluate the current policy and update $\hat{J}_i$; this updated distribution accounts for policy shifts and guides BO-based fine-tuning. We continue this process until each policy reaches convergence, defined as $< 5\%$ variation in success rate over 5 consecutive epochs during evaluation.

In this work, we explored three acquisition functions:

1) **Upper Confidence Bound (UCB)**, which encourages exploration by selecting samples with high uncertainty:

$$A_{UCB} = \mu(\theta) + \beta\sigma(\theta)$$

[2]Sampling is parallelized and conducted entirely in simulation: executing 1000 rollouts takes only seconds on a single GPU.

Algorithm 1 Bayesian Optimization (BO)-Guided Fine-Tuning

Input: Assembly steps $\{A_1, A_2, ..., A_N\}$
for each assembly step A_i **do**
  Train specialist policy π_i using [5]
  while fine-tuning not converged **do**
    Evaluate success rate distribution $J_i(\theta)$ at different plug initializations θ
    Use Active Sampling (Equation 3) to propose samples θ'
    Fine-tune each policy by initializing at proposed θ'
  end while
end for

where is a hyperparameter that controls the trade-off between exploitation and exploration, and $\mu(\theta)$ and $\sigma(\theta)$ are the predicted mean and standard deviation from the GP. [3]

2) **Probability of Improvement (PI)**, which selects samples based on the likelihood of exceeding the best observed success rate J^+:

$$A_{PI}(\theta) = \Phi\left(\frac{\mu(\theta) - J^+}{\sigma(\theta)}\right)$$

where $J^+ = \max J_i(\theta)$ is the best observed success rate, and $\Phi(\cdot)$ is the cumulative distribution function (CDF) of the standard normal distribution.

3) **Expected Improvement (EI)**, which selects the samples that are expected to improve over the best observed success rate J^+:

$$A_{EI}(\theta) = (\mu(\theta) - J^+)\Phi(Z) + \sigma(\theta)\phi(Z)$$

where $Z = \frac{\mu(\theta) - J^+}{\sigma(\theta)}$ is the standardized improvement, and $\Phi(Z)$ and $\phi(Z)$ are the CDF and the probability density function of the standard normal distribution.

B. GMM Sampling for Deployment-Time Optimization

While policy fine-tuning focuses on improving each policy's performance across different plug initializations, the focus at *deployment* time is on identifying high-performing initial positions to maximize success rates. More specifically, we want to sample plug initializations with high probability of execution success given the success rate distribution approximated from evaluation rollouts.

To model the success distribution, we first collect a dataset by uniformly sampling $(N = 1000)$ initial plug poses $(\{x_i\}_{i=1}^N)$ from the predefined initialization space $(\mathcal{X})$ and executing the fine-tuned policy (π) from each initialization [4]. Note that x_i includes both initial plug position θ_i and orientation. For each rollout, we record the binary success outcome $(y_i \in \{0, 1\})$. We then extract the subset $\mathcal{D}_{\text{succ}} = \{x_i \mid y_i = 1\}$ from $(\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N)$ corresponding to successful executions and fit a Gaussian Mixture Model (GMM) to this set to estimate the underlying distribution. At deployment time, we sample $(M = 1000)$ candidate initializations $(\{\tilde{x}_j\}_{j=1}^M)$ from the GMM and compute the GMM probability density $(p(\tilde{x}_j))$ for each sample. The final initialization is selected as $(x^* = \arg\max_j p(\tilde{x}_j))$, i.e., the sample with the highest probability density under the GMM. This strategy enables efficient initialization selection by leveraging learned success distributions, avoiding brute-force search; a detailed description of the procedure is in Algorithm 2.

[3]We evaluated several β values and empirically choose $\beta = 2.0$ in our experiments; however, our approach is not particularly sensitive to β.

[4]Empirically, we found $N = 1000$ rollouts to enable stable mixture fitting without overfitting.

Algorithm 2 Deployment-time Optimization via GMM Sampling

// Offline Phase: Data Collection and GMM Fitting
Sample N plug initializations $\{x_i\}_{i=1}^N \sim \mathcal{U}(\mathcal{X})$
Execute π for each x_i, record binary success $y_i \in \{0, 1\}$, and collect a dataset of successful samples: $\mathcal{D}_{\text{succ}} = \{x_i \mid y_i = 1\}$
Fit a Gaussian Mixture Model with K components to $\mathcal{D}_{\text{succ}}$:

$$p_{\text{GMM}}(x) = \sum_{k=1}^{K} \xi_k \, \mathcal{N}(x \mid \mu_k, \Sigma_k)$$

// Online Phase: Deployment-time Plug Initialization Selection
Sample M candidates from the GMM: $\{x'_j\}_{j=1}^M \sim p_{\text{GMM}}(x)$
Compute GMM probability density $s_j = p_{\text{GMM}}(x'_j)$ for each x'_j
Select $x^* = \arg\max_{x'_j} s_j$ and initialize the plug at x^*
Execute policy π

V. Experimental Results

We present a detailed evaluation of our trained policies in both 2-part assembly and multi-part assembly settings. The key takeaway is that our proposed approach can substantially improve individual policy performance, as well as the execution success of multi-step assembly sequences.

A. Simulation-based Evaluation

1) BO-Guided Fine-Tuning: We now present the results of our proposed approach that uses Bayesian Optimization (BO) to propose plug initialization samples for fine-tuning.

Our first evaluation question is, **which acquisition function is the most effective for proposing plug initialization samples during fine-tuning for multi-part assembly?** We evaluate our 3 acquisition functions on all 5 multi-part assemblies, which consist of 26 total assembly steps. For each assembly step in each multi-part assembly, we train a specialist policy from scratch over 5 random seeds using [5] until convergence, and we use these policies as our baseline. All baselines converge within 200 epochs, beyond which performance does not improve. Then, for each acquisition function, we follow the procedure described in Algorithm 1 to fine-tune all 5 seeds of each specialist policy. The fine-tuning procedure requires only 50 epochs, which is significantly fewer than training-from-scratch. We evaluate each seed for 1000 trials and report the average success rate for each policy over the 5 seeds (Figure 3). The average success rate is 83.61±9.27% for baseline policies, 91.82±10.05% for policies fine-tuned with PI as acquisition function, 91.12±10.61% for policies fine-tuned with EI, and 94.41±7.51% for policies fine-tuned with UCB.

Our second evaluation question is, **does our proposed fine-tuning approach also significantly improve policy performance compared to the state-of-the-art for 2-part assembly?** We now use UCB as acquisition function and

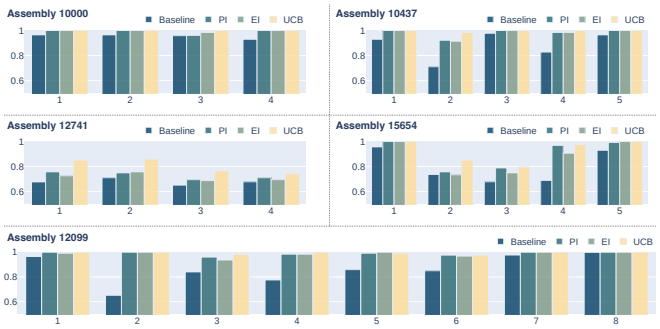

Fig. 3: Comparison of success rate by using different acquisition function for fine-tuning in 5 multi-part assemblies (see Figure 2 for asset IDs). X-axis corresponds to step index within each assembly.

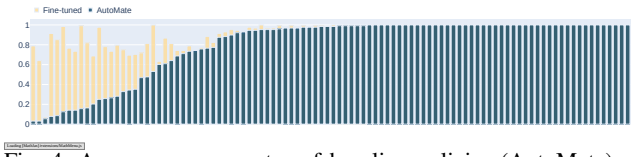

Fig. 4: Average success rates of baseline policies (AutoMate) and fine-tuned policies across 100 2-part assembly tasks. Policies are sorted by baseline success rate. BO-guided fine-tuning significantly improves the average success rate from 80.53% to 91.51%, with greatest improvements occurring for the most challenging tasks.

again follow the procedure described in Algorithm 1 to fine-tune specialist policies, but now over 100 2-part assembly tasks from an established baseline [5] [5]. We evaluate using the same procedure as for the prior evaluation. The baseline policies, trained using [5] without any fine-tuning, achieve an average success rate of 80.53±31.43%. After applying our proposed fine-tuning procedure, the average success rate increases to 91.51±14.27%, demonstrating a clear performance gain (10.98% increase in mean success rate and 17.16% decrease in success rate variance). Improvement is consistently observed across the majority of tasks, with a number of fine-tuned policies approaching or reaching a 100% success rate (Figure 4).

2) GMM-based Sampling for Deployment-Time Optimization: We now present the results of our proposed approach that uses GMM-based sampling to determine plug initializations during policy deployment, as opposed to uniform sampling from the initialization space. Our evaluation question is, **does our proposed deployment-time optimization approach significantly improve performance of baseline and fine-tuned policies for multi-part assembly?**

We compare the following 4 approaches for training and deploying individual policies: (1) *Baseline:* Train policy from scratch using [5] and uniformly sample plug initialization, (2) *Deployment:* Train policy from scratch using [5] and sample plug initialization with GMM, (3) *Fine-tune:* Fine-tune baseline policy with BO and uniformly sample plug initialization, (4) ***Refinery*** (Ours): Fine-tune baseline policy with BO and sample plug initialization with GMM.

The policies are derived from Section V-A.1, which

[5]Visualization of the dataset can be found on their project website.

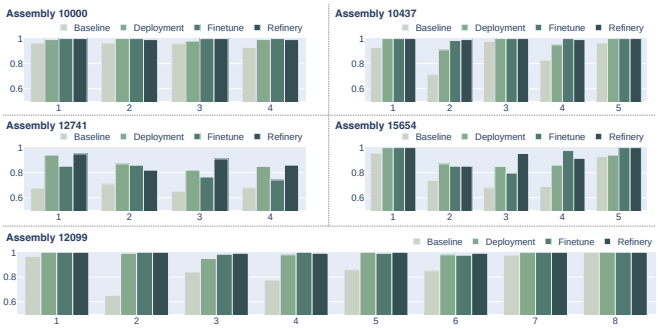

Fig. 5: Comparison of success rate under different fine-tuning and deployment-time sampling strategies in 5 multi-part assemblies.

Asset ID	Baseline	Deployment	Fine-tune	Refinery (Ours)
10000	83.20	96.47 (+13.27)	**99.22 (+16.02)**	98.45 (+15.25)
12099	31.26	91.55 (+60.29)	91.52 (+60.26)	**97.68 (+66.42)**
12741	21.10	56.78 (+35.68)	48.65 (+27.55)	**60.62 (+39.52)**
10437	49.11	86.45 (+37.34)	98.44 (+49.33)	**98.44 (+49.33)**
15654	31.98	56.48 (+24.50)	66.22 (+34.24)	**74.19 (+42.21)**

TABLE I: Success rates for full assembly sequences.

produced trained-from-scratch and UCB-fine-tuned policies for each assembly step over 5 random seeds. For each of the 4 approaches listed above, we evaluate all 5 seeds for every assembly step over 1000 trials and calculate the average success rate (Figure 5). The initializations of all previously assembled parts are randomly sampled (i.e., the goal configuration is randomized), and we apply uniformly-sampled observation noise (within ±2mm on position, ±10° on orientation) on goal poses during training and evaluation.

The average success rate is 83.61±9.27% for *Baseline*, 94.41±7.51% for *Fine-tune*, 94.50±4.90% for *Deployment*, and 96.35±4.45% for ***Refinery***. Specifically, we observe that GMM-based initialization alone (*Deployment*) improves average success rates over *Baseline* without requiring additional online adaptation. This highlights the effectiveness of modeling success rate distributions using GMM and exploiting them for informed sampling during policy execution. However, combining BO-guided fine-tuning with GMM-based deployment consistently achieves the highest success rates across all assemblies, outperforming both individual components. This demonstrates that while GMM-based sampling during deployment reduces the reliance on online fine-tuning, it does not eliminate the benefits of fine-tuning. These results collectively suggest that to fully maximize success rates (critical for industrial settings), both a BO-based fine-tuning approach and GMM-based sampling strategy should be employed.

3) Full Assembly Sequence: To demonstrate the effectiveness of the proposed combination of BO-guided fine-tuning and GMM-based sampling during deployment, we evaluate the success rate of the full assembly sequence across all 5 multi-part assemblies (Table I). In these experiments, a sequence is only considered as successful if all steps are executed successfully. Our proposed method consistently outperforms the baseline across all evaluated assemblies,

Asset ID	Baseline		Fine-tune		Refinery (Ours)	
	Simulation (%)	Reality	Simulation (%)	Reality	Simulation (%)	Reality
01053	56.25	2/10	91.41	7/10	97.66	9/10
00190	54.69	4/10	87.34	10/10	99.22	10/10
00514	76.56	6/10	88.28	8/10	89.84	8/10
00614	84.37	8/10	89.69	9/10	91.86	10/10
00681	94.53	8/10	96.87	8/10	100.00	10/10
00553	92.18	9/10	94.53	10/10	97.66	10/10
00768	96.87	9/10	98.44	10/10	99.22	10/10
00346	97.66	10/10	100.00	10/10	100.00	10/10
01036	100.00	10/10	100.00	10/10	100.00	10/10
01129	96.88	10/10	100.00	10/10	100.00	10/10
Total	84.99	76/100	94.66 (+9.67)	92/100 (+16)	97.54 (+12.55)	97/100 (+21)

TABLE II: Real world evaluation. Comparison of policy success rates under different fine-tuning and deployment-time sampling strategies on 10 two-part assemblies, over a total of 300 real-world trials.

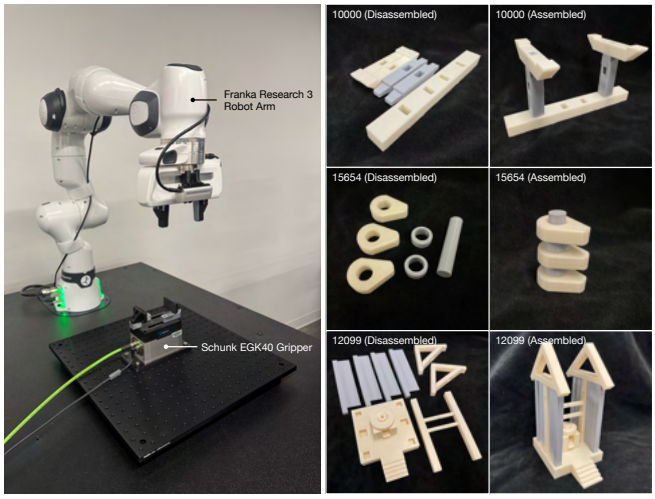

Fig. 6: Real-world experiment setup. The Franka Research 3 (FR3) robot and a table-mounted Schunk EGK40 gripper is used to perform multi-part assembly tasks. 3D-printed 10000, 12099, and 15654 are shown in disassembled and assembled configurations.

demonstrating its general applicability and robustness. Notably, assemblies with lower baseline performance, such as 12099 (31.26%) and 15654 (31.98%), exhibit substantial improvements under our approach, achieving success rates of 97.68% and 74.19%, respectively. Even in assemblies with relatively high baseline performance, such as 10000, our method yields a meaningful absolute gain of 15.25%. Overall, these findings underscore the synergistic benefit of combining uncertainty-aware fine-tuning with success-driven sampling for improving sequential task performance.

B. Real-world Evaluation

To assess the real-world effectiveness of our approach, we evaluate performance across a diverse set of two-part and multi-part assemblies in the real world. Our real-world system (Figure 6) consists of a robot arm with a parallel-jaw gripper, another parallel-jaw gripper mounted to the tabletop, and 3D-printed assemblies from our dataset (Figure 2).

Our first evaluation question is, **does the proposed approach for fine-tuning and deployment also improve real-world policy performance?** We evaluate the *Baseline, Fine-tune, and* ***Refinery*** approaches listed in subsubsection V-A.2 on 10 two-part assemblies. In these experiments, only ***Refinery*** uses GMM-based initialization during deployment, and the GMM is directly modeled from data collected in simulation. For each of 10 assemblies, we deploy the corresponding policy for each approach 10 times, for a total of 300 trials. Our results are shown in Table II. For assemblies where baseline policies already achieve high success rates (e.g., 00346, 01036, 01129), our method maintains near-perfect performance. In cases with moderate baseline success (e.g., 00614, 00681, 00553, 00768), the approach effectively eliminates residual failure cases, reducing the total number of failures from 3/40 to 0/40. For more challenging assemblies with low baseline success rates (e.g., 01053, 00190, 00514), the improvement is substantial, increasing the real-world success count from 12/30 to 27/30. The results demonstrate that our proposed approach consistently improves real-world policy performance without additional adaptation.

Asset ID	Step	Fully Auto.	1 Intervention	2 Interventions
10000	37/40	8/10	10/10	10/10
12099	65/70	6/10	10/10	10/10
15654	44/50	5/10	9/10	10/10

TABLE III: Success rates for full assembly and individual assembly in the real world across 3 multi-part assemblies.

Our second evaluation question is, **can we complete full sequences of multi-part assembly in the real world?** We evaluate our approach on three multi-part assemblies executed by a physical robot. For each assembly, the robot sequentially executes the learned policy for each subtask. If an intermediate step fails, a human operator intervenes to correct the failure before continuing. Each assembly is evaluated over 10 full-sequence trials, corresponding to 160 individual subtask executions. We report fully autonomous success rates and the success rate with 1 or 2 human interventions from the same experiments. As shown in Table III, the average fully-autonomous success rate across individual subtasks is 90.6% (146/160), indicating strong per-step reliability. However, execution of the entire assembly sequence is more challenging: for 10000, a notable 80% success rate was achieved over the sequence; for 15654,

though, a 50% success was achieved, primarily due to a single failing step, which is reflected in the success rates with only 1 or 2 human interventions.

C. Limitations

Across both simulation and real-world experiments, we identified the key causes of remaining failure cases. One cause was grasp instability, where the parallel-jaw gripper slipped or introduced unintended rotations during part transport. In addition, in multi-step assembly, the robot could apply large forces during policy execution, which destabilized or displaced previously assembled parts. Additionally, in real-world trials, failures arose from structural limitations of 3D-printed components, such as flexing or tolerance mismatches with the 3D mesh due to printer settings, which could not be corrected by the policy. These observations suggest that addressing robustness in grasping, integrating force control, and ensuring part stability may further advance long-horizon assembly.

VI. Conclusion

We presented **Refinery**, a framework for improving the fine-tuning and deployment of learned policies in multi-part robotic assembly. By identifying high-uncertainty initial states with Bayesian Optimization and leveraging Gaussian Mixture Models to prioritize robust initializations at deployment, Refinery enhances both individual policy performance and long-horizon execution success. Our experiments show substantial gains on two-part and multi-part assembly tasks, achieving state-of-the-art results in both simulation and the real world. Several promising directions emerge from this work. Expanding the framework to more geometrically-diverse assembly tasks would test its generality beyond the current benchmark set. Incorporating perception-driven initialization could enable more autonomous deployment in unstructured environments. Additionally, exploring joint optimization strategies across policy chains may further improve sequence-level reliability in long-horizon assembly. Finally, future research will investigate how to bridge the gap towards additional industrial requirements, such as stricter tolerances and time constraints.

References

[1] X. Zhang, M. Tomizuka, and H. Li, "Bridging the sim-to-real gap with dynamic compliance tuning for industrial insertion," *arXiv preprint arXiv:2311.07499*, 2023.

[2] 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 *Conference on Robot Learning*, 2023.

[3] B. Tang et al., "IndustReal: Transferring contact-rich assembly tasks from simulation to reality," in *Robotics: Science and Systems*, 2023.

[4] M. Noseworthy et al., "FORGE: Force-guided exploration for robust contact-rich manipulation under uncertainty," *IEEE Robotics and Automation Letters*, 2025.

[5] B. Tang et al., "AutoMate: Specialist and Generalist Assembly Policies over Diverse Geometries," in *Robotics: Science and Systems*, 2024.

[6] Y. Guo, B. Tang, I. Akinola, D. Fox, A. Gupta, and Y. Narang, "SRSA: Skill Retrieval and Adaptation for Robotic Assembly Tasks," *arXiv preprint arXiv:2503.04538*, 2025.

[7] G. Gencer, *First Pass Yield: Calculation, Examples and Improvement Strategies*, Accessed: 2025-04-30, Sep. 2024.

[8] Y. Tian et al., "ASAP: Automated Sequence Planning for Complex Robotic Assembly with Physical Feasibility," *arXiv preprint arXiv:2309.16909*, 2023.

[9] D. E. Whitney, *Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development*. Oxford University Press, 2004.

[10] M. T. Mason, *Mechanics of Robotic Manipulation*. MIT Press, 2001.

[11] S. H. Drake, "Using compliance in lieu of sensory feedback for automatic assembly," Ph.D. dissertation, Massachusetts Institute of Technology, 1978.

[12] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *The International Journal of Robotics Research*, 1984.

[13] Y. Xia, Y. Yin, and Z. Chen, "Dynamic analysis for peg-in-hole assembly with contact deformation," *The International Journal of Advanced Manufacturing Technology*, 2006.

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

[15] J. Luo et al., "Reinforcement learning on variable impedance controller for high-precision robotic assembly," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[16] O. Spector and M. Zacksenhouse, "Deep reinforcement learning for contact-rich skills using compliant movement primitives," *arXiv:2008.13223 [cs]*, 2020.

[17] X. Zhang, S. Jin, C. Wang, X. Zhu, and M. Tomizuka, "Learning insertion primitives with discrete-continuous hybrid action space for robotic assembly tasks," *arXiv:2110.12618 [cs]*, 2021.

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

[19] J. Luo et al., "Robust multi-modal policies for industrial assembly via reinforcement learning and demonstrations: A large-scale study," *arXiv preprint arXiv:2103.11512*, 2021.

[20] M. Hebecker, J. Lambrecht, and M. Schmitz, "Towards real-world force-sensitive robotic assembly through deep reinforcement learning in simulations," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, 2021.

[21] L. Shao, T. Migimatsu, and J. Bohg, "Learning to scaffold the development of robotic manipulation skills," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[22] T. Johannink et al., "Residual reinforcement learning for robot control," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[23] Y. Fan, J. Luo, and M. Tomizuka, "A learning framework for high precision industrial assembly," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[24] M. Vecerik et al., "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv:1707.08817 [cs]*, 2018.

[25] T. Z. Zhao et al., "Offline meta-reinforcement learning for industrial insertion," in *International Conference on Robotics and Automation (ICRA)*, 2022.

[26] O. Spector and D. Di Castro, "InsertionNet: A scalable solution for insertion," *IEEE Robotics and Automation Letters*, 2021.

[27] B. Wen, W. Lian, K. Bekris, and S. Schaal, "You only demonstrate once: Category-level manipulation from single visual demonstration," *arXiv preprint arXiv:2201.12716*, 2022.

[28] NVIDIA, *Isaac Lab*, 2025.

[29] Y. Narang et al., "Factory: Fast contact for robotic assembly," in *Robotics: Science and Systems*, 2022.

[30] Z. Xian, Y. Qiao, X. Zhenjia, T.-H. Wang, Z. Chen, J. Zheng, et al., *Genesis: A Universal and Generative Physics Engine for Robotics and Beyond*, Dec. 2024.

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

[32] Y. Jiang, C. Wang, R. Zhang, J. Wu, and L. Fei-Fei, "TRANSIC: Sim-to-Real Policy Transfer by Learning from Online Correction," in *Conference on Robot Learning*, 2024.

[33] L. Ankile, A. Simeonov, I. Shenfeld, and P. Agrawal, "JUICER: Data-Efficient Imitation Learning for Robotic Assembly," *arXiv*, 2024.

[34] G. Konidaris and A. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," *Advances in neural information processing systems*, vol. 22, 2009.

[35] A. Bagaria and G. Konidaris, "Option discovery using deep skill chaining," in *International Conference on Learning Representations*, 2019.

[36] A. Clegg, W. Yu, J. Tan, C. K. Liu, and G. Turk, "Learning to dress: Synthesizing human dressing motion via deep reinforcement learning," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 6, pp. 1–10, 2018.

[37] Y. Lee, J. J. Lim, A. Anandkumar, and Y. Zhu, "Adversarial skill chaining for long-horizon robot manipulation via terminal state regularization," *arXiv preprint arXiv:2111.07999*, 2021.

[38] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *2011 IEEE International Conference on Robotics and Automation*, IEEE, 2011, pp. 1470–1477.

[39] L. P. Kaelbling and T. Lozano-Pérez, "Pre-image backchaining in belief space for mobile manipulation," in *Robotics Research: The 15th International Symposium ISRR*, Springer, 2016, pp. 383–400.

[40] Y. Chen, C. Wang, L. Fei-Fei, and C. K. Liu, "Sequential dexterity: Chaining dexterous policies for long-horizon manipulation," *arXiv preprint arXiv:2309.00987*, 2023.

[41] Z. Chen, Z. Ji, J. Huo, and Y. Gao, "Scar: Refining skill chaining for long-horizon robotic manipulation via dual regularization," *Advances in Neural Information Processing Systems*, vol. 37, pp. 111 679–111 714, 2024.

[42] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, Pmlr, 2018, pp. 1861–1870.

[43] S. Levine and V. Koltun, "Guided policy search," in *International conference on machine learning*, PMLR, 2013, pp. 1–9.

[44] S. Müller, A. von Rohr, and S. Trimpe, "Local policy search with Bayesian optimization," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20 708–20 720, 2021.

[45] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, pp. 455–492, 1998.