Delta Action Model for Narrowing the Sim-to-Real Gap in Contact-Rich Robotic Manipulation

Zhongqi Huang
University of Michigan Email: iamdonkh@umich.edu

***Abstract*—Training reinforcement learning (RL) policies directly on physical robotic systems is often impractical due to limited parallelism, safety concerns, and the need for human supervision. As a result, policies are commonly trained in simulation and then transferred to the real world; however, discrepancies between simulated and real dynamics often cause severe performance degradation. This project investigates a delta action modeling approach to narrow the simulation-to-reality (sim-to-real) gap for contact-rich manipulation tasks. Building on the ASAP framework, a delta action model is trained to predict an additive correction to the policy action that compensates for unmodeled dynamic differences between simulation and reality. The method is evaluated on a peg insertion task, where a policy trained in a large-hole simulation environment experiences significant performance drop when deployed in a small-hole environment representing reality. Experimental results show that incorporating the delta action model yields trajectories that more closely match real reference trajectories and enables higher reward during policy fine-tuning in simulation. However, improvements in reward do not directly translate to higher success rates in the real-like environment, indicating sensitivity to noise and reward design. These results suggest that delta action modeling is a promising and data-efficient strategy for sim-to-real transfer, while also highlighting the need for improved regularization and reward alignment in future work.**

I. Introduction

Robotic manipulation tasks involving frequent contacts, such as peg insertion, are highly sensitive to modeling inaccuracies in dynamics, friction, and contact interactions. Reinforcement learning has shown strong performance in simulated environments, but transferring learned policies to real robotic systems remains challenging due to the simulation-to-reality (sim-to-real) gap. This gap arises from unavoidable discrepancies between simulated physics and real-world dynamics, including imperfect contact models, unmodeled compliance, and sensor noise.

Several approaches have been proposed to mitigate the sim-to-real gap. System identification methods aim to explicitly estimate physical parameters of the real system, but they require predefined parameter spaces and accurate ground-truth measurements. [2] Domain randomization exposes policies to a wide range of simulated parameters, improving robustness [4], [5], [7] but often resulting in overly conservative behavior. Learning full dynamics models has also been explored [2], [3], [8]; however, such approaches are typically data-intensive and has not been scaled to high-dimensional, contact-rich systems. [1]

An alternative strategy is to learn an action-level correction that directly compensates for discrepancies between simulation and reality rather than modeling the entire system dynamics. The ASAP framework introduced a delta action model for agile humanoid motion tracking, demonstrating that learning additive action corrections can effectively align simulated and real behaviors. [1] Motivated by this idea, this work extends delta action modeling to a contact-rich manipulation task.

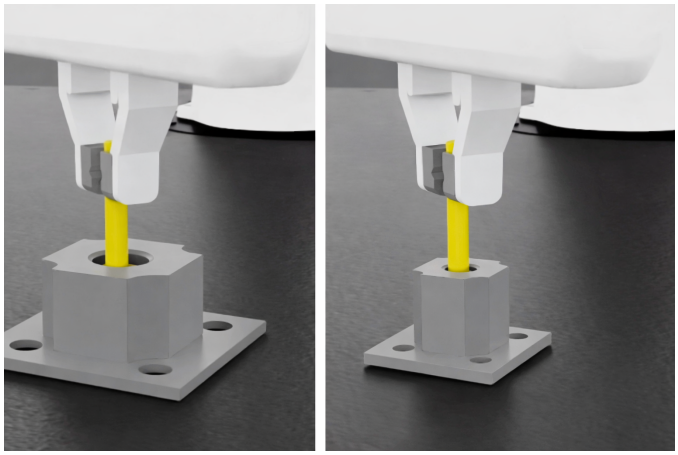

Fig. 1. Task Setup: a big peg hole is used for simulation and a small peg hole is used for ”reality”.

This report investigates whether a delta action model can narrow the sim-to-real gap for contact-rich tasks by learning dynamic discrepancies between simulated and real-like environments. Section II describes the proposed approach and training procedure. Section III presents experimental results. Section IV discusses implications and limitations, and Section V concludes the paper with directions for future work.

II. Approach

A. Problem Setup

The target task is a robotic peg insertion problem. A policy is first trained in simulation using a large peg hole, which provides a forgiving environment for learning. The task in ”reality” is represented by a small peg hole, where tighter tolerances and contact interactions cause frequent failures when deploying the simulation-trained policy. The related parameters are shown in Table. I.

B. Method Overview

To investigate whether a delta action model can reduce the sim-to-real gap in contact-rich manipulation tasks, a policy is first trained in simulation using a large peg hole. The

TABLE I
PEG HOLE GEOMETRY IN SIMULATION AND REAL ENVIRONMENTS

Environment	Hole Diameter (mm)	Description
Simulation (Large Hole)	16	Forgiving, low contact constraint
Real-like (Small Hole)	8	Tight tolerance, contact-rich

trained policy is then deployed in "reality," represented by a small peg hole, to collect reference trajectories that expose discrepancies in contact dynamics. These reference trajectories are subsequently used to train a delta action model that learns action-level corrections aimed at minimizing the sim-to-real gap. Finally, the policy is fine-tuned in a modified simulation environment that integrates the learned delta action model, and the success rate of the fine-tuned policy is evaluated in "reality."

C. Delta Action Model

Rather than learning a complete dynamics model, a delta action model is trained to output an additive correction to the dynamics in simulation. At each control step, the action executed by the environment is given by

$$\mathbf{a}_{\text{exec}} = \mathbf{a}_{\text{policy}} + \Delta\mathbf{a}, \tag{1}$$

where $\mathbf{a}_{\text{policy}}$ is the action produced by the RL policy and $\Delta\mathbf{a}$ is the correction predicted by the delta action model. The simulator then advances the system state using $\mathbf{a}_{\text{exec}}$, ensuring that the resulting state remains physically feasible.

The goal of training the delta action model and integrating it in a modified simulation environment is to minimize the sim-to-real gap, so that the policy trained in the modified environment can successfully transfer to reality. In other words, given the current state s and action a output by the policy, the action output by the delta action model Δa is aimed to achieve the equation:

$$f^{\text{sim}}\big(s, a + \pi^{\Delta}(s,a)\big) \;\approx\; f^{\text{real}}(s,a), \tag{2}$$

This formulation offers several advantages. First, the delta action model learns only the discrepancy between simulation and reality, making it simpler and more data-efficient than learning full dynamics, so that the model can be trained with a relatively small dataset of real trajectories. Second, because the simulator governs state transitions, physically infeasible states are avoided.

D. Training Policy in Simulation

A policy is first trained in simulation using a large peg hole. The observation space follows the default IsaacLab configuration and includes the robot joint positions and velocities, as well as the poses and velocities of the held object (peg) and the fixed object (peg hole). The action space is a 6-dimensional vector that controls the position and orientation of the held object. The reward function is inherited from the default IsaacLab task specification.

Algorithm 1 Training Delta Action Model

Require: Recorded trajectory $\{(s_t^{\text{ref}}, a_t^{\text{ref}})\}_{t=1}^{T}$
Require: Policy π_θ, environment $\mathcal{E}$
Initialize policy parameters θ
for each training episode **do**
  $s_0^{\text{ref}} \leftarrow \mathcal{E}.\text{reset}()$
  for $t = 0$ to $T-1$ **do**
    Retrieve next-step reference action a_{t+1}^{ref}
    Construct observation $o_t = (s_t, a_{t+1}^{\text{ref}})$
    Sample residual action $a_t^{\text{agent}} \sim \pi_\theta(a_t^{\text{agent}} \mid o_t)$
    Execute composed action $a_t = a_t^{\text{ref}} + a_t^{\text{agent}}$
    $s_{t+1} \leftarrow \mathcal{E}.\text{step}(a_t)$
    Compute reward $r_t = f(s_t, s_t^{\text{ref}})$
  end for
  Update policy parameters θ using collected rewards
end for
return trained policy π_θ

E. Training Delta Action Model

The delta action model is designed to output additive action corrections that reduce discrepancies between simulated and real-world dynamics. To achieve this, the model is trained using reference trajectories. These trajectories are generated in simulation with a large peg hole and subsequently replayed in "reality" with a smaller peg hole to expose differences in contact dynamics.

Compared to the observation space used for policy learning, which consists of robot and object state information, the observation space of the delta action model is augmented with the current reference action. This additional input explicitly encodes the intended control command and enables the model to learn residual, action-level corrections.

At the beginning of each episode, the environment is assigned a randomly selected reference trajectory and initialized to the same initial state as the reference. At each time step, the environment executes the composed action $\mathbf{a}_{\text{exec}}$ defined in (1) and returns a reward based on the deviation between the resulting next state and the corresponding next reference state. The reward function used for training the delta action model is defined as

$$r = \lambda_j e^{-\text{MSE}_j/\sigma_j} + \lambda_h e^{-\text{WMSE}_h/\sigma_h} + \lambda_f e^{-\text{MSE}_f/\sigma_f}. \tag{3}$$

where the subscripts j, h, and f denote the robot joint pose, the pose of the held object (peg), and the pose of the fixed object (peg hole), respectively. MSE denotes the mean squared error between the current pose and the corresponding reference pose and WMSE denotes the weighted mean square error. The coefficients λ_j, λ_h, and λ_f weight the relative importance of each term in the reward. The overall training procedure of the delta action model is summarized in Algorithm 1.

F. Fine-tuning policy

The policy is initially trained in simulation using a large peg hole. To facilitate successful transfer to the real-like environ-

ment with a small peg hole, the policy is subsequently fine-tuned in a modified environment that integrates the learned delta action model. During policy fine-tuning, the parameters of the delta action model are frozen, and only the policy parameters are updated.

At each control step, the environment executes the composed action $\mathbf{a}_{\text{exec}}$, defined as the sum of the policy action $\mathbf{a}_{\text{policy}}$ and the delta action Δa. Unlike the delta action training phase, the environment is randomly initialized at each episode reset rather than being initialized to a reference state, enabling the policy to learn robust behaviors under diverse initial conditions.

In contrast to the delta-action training reward, the policy fine-tuning stage uses the unmodified, task-native reward specified by IsaacLab for peg insertion.

G. Training Pipeline

All policies trained uses the defualt PPO agent in IsaacLab. The training pipeline consists of four stages:

1) **Policy Training in Simulation:** A policy is trained in the large-hole simulation environment.
2) **Reference Trajectory Collection:** The trained policy is executed in "reality", and 50 reference trajectories are recorded.
3) **Delta Action Model Training:** The delta action model is trained in simulation to minimize the discrepancy between trajectories generated by the reference action plus delta action and the recorded reference trajectories.
4) **Policy Fine-tuning:** The simulation environment augmented with delta actions is treated as a closer approximation of reality, and the policy is fine-tuned in this modified environment.

III. Results

A. Performance Drop Without Delta Action

A significant sim-to-real performance drop is observed when the policy trained in the large-hole simulation is evaluated in the small-hole environment. The success rate decreases from 99.2% (127/128 trials) in the large-hole environment to 43.75% (56/128 trials) in the small-hole environment. It is observed that failures frequently occur when the peg becomes stuck at the edge of the hole.

B. Training Delta Action Model

Using 50 reference trajectories, the delta action model converges after approximately 200 training episodes, achieving an average reward of 445.2. To verify that the delta action model effectively reduces the discrepancy between the reference trajectories and the trajectories generated by applying the delta action, a baseline evaluation is conducted. Specifically, the reward is computed over the same 50 reference trajectories using only the reference action without any delta correction (i.e., zero delta action), resulting in an average reward of 439.1.

Since the reward function in (3) is defined in terms of the mean squared error and weighted mean squared error between the reference trajectories and the generated trajectories, a higher reward corresponds to a smaller trajectory deviation. The observed improvement in reward therefore indicates that the delta action model successfully narrows the gap between the simulated trajectories and the reference trajectories. Figure 2 shows the training reward curve of the delta action model.

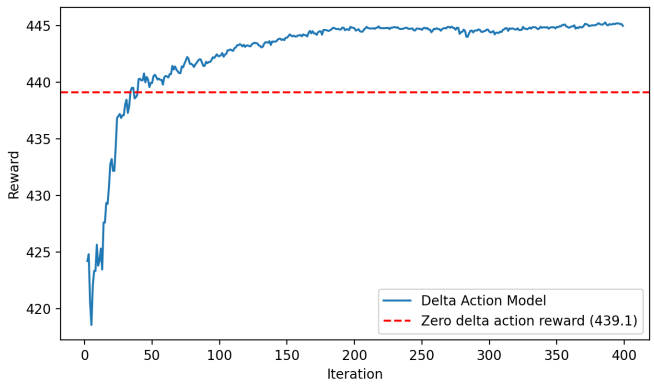

Fig. 2. Training reward curve of the delta action model. The red dashed line indicates the reward achieved by executing the reference action without delta correction (zero delta action).

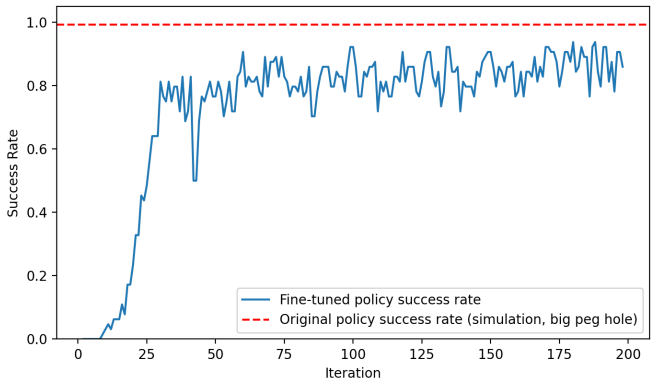

Fig. 3. Training success rate curve of the fine-tuned policy. The red dashed line indicates the success rate achieved by the original policy in simulation without delta action model.

C. Policy Fine-Tuning Results

The policy is fine-tuned in the large-hole simulation environment augmented with the delta action model. Figure 3 shows the success rate of the fine-tuned policy during fine-tuning, which is around 89% at convergence. The red dashed line indicates the success rate achieved by the original policy in simulation with a large peg hole. Figure 4 shows the reward curve of the policy during fine-tuning, which converges at around 50 episodes. While fine-tuning improves performance in the modified environment, the success rate does not reach the level achieved by the original policy (99.2%) under the more forgiving simulation setting.

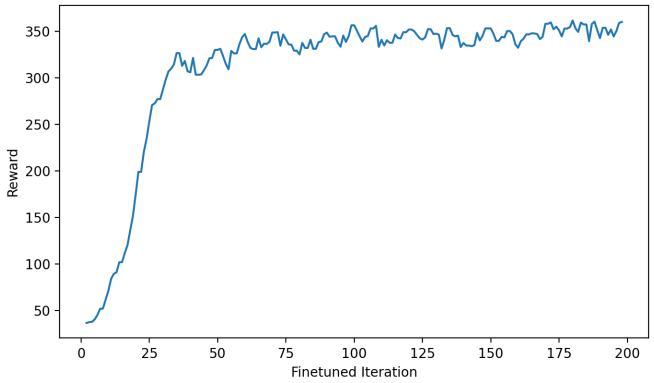

Fig. 4. Training reward curve of the fine-tuned policy.

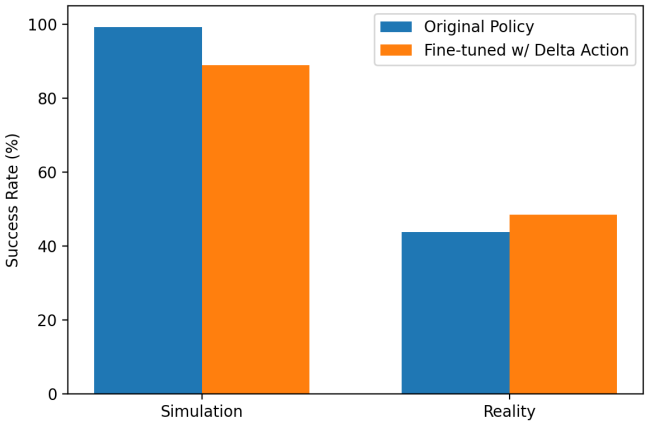

Fig. 5. Success Rate Comparison Before and After Delta Action Fine-Tuning

TABLE II
SUCCESS RATES OF FINE-TUNED POLICY EVALUATED IN "REALITY"

Fine-tuning Episodes	Success Rate (%)
20	48.4
50	39.1
100	38.3
200	40.1

D. Test of Fine-Tuned Policy in "Reality"

The fine-tuned policy is evaluated in "reality" (the small peg hole) at multiple fine-tuning stages. Specifically, policy checkpoints obtained after 20, 50, 100, and 200 fine-tuning episodes are each evaluated over 128 trials to estimate the corresponding success rates. Table. II summarizes the test success rates for the fine-tuned policy at different stages.

IV. DISCUSSION

Without delta action augmentation, a significant sim-to-real gap is observed when comparing the success rates of a policy trained and evaluated in simulation with those obtained when the same policy is tested in "reality." Specifically, the success rate drops from 99.2% in simulation to 43.75% in "reality." This result confirms that the proposed problem setup, which uses large and small peg holes to emulate the sim-to-real gap, is effective and meaningful.

From the training curve of the delta action model, it can be observed that the reward achieved when reproducing the reference states in simulation (large peg hole) is consistently higher than that obtained with zero delta action. Since the reward is inversely related to the discrepancy between the generated and reference trajectories, this improvement indicates that the delta action model successfully learns nontrivial information during training, specifically capturing part of the dynamic gap between simulation and "reality."

However, based on the policy fine-tuning results, which converge at approximately 50 episodes, the fine-tuned policy fails to achieve a near-100% success rate in the delta-action-augmented environment. This suggests that, while the delta action model compensates for the sim-to-real gap, it may also introduce additional noise into the control process. As a result, the augmented environment becomes dynamically more challenging, making successful peg insertion harder.

When evaluating the fine-tuned policy at different training stages in "reality" (small peg hole), the best performance is achieved at an early fine-tuning stage (20 episodes), whereas the success rate degrades after convergence (50 episodes). This behavior indicates that the learned delta action model is biased, and excessive fine-tuning leads to overfitting to the augmented simulation environment. Consequently, the converged policy fails to generalize well to "reality," despite achieving higher training rewards.

The results show that the delta action model captures non-trivial dynamic discrepancies between simulation and "reality," as evidenced by its higher trajectory reproduction reward compared to the zero-delta-action baseline. This indicates that the model learns meaningful action-level corrections that reduce the mismatch between generated and reference trajectories in the augmented simulation environment.

However, fine-tuning the policy in the delta-action-augmented environment does not consistently improve real-task performance. While early-stage fine-tuning achieves the highest success rate in "reality," continued fine-tuning leads to performance degradation, suggesting that the delta action model introduces additional bias and noise. As training progresses, the policy overfits to the augmented simulation dynamics, which are more challenging and differ subtly from the true target environment.

Overall, these findings indicate that delta action modeling alone is insufficient to guarantee improved sim-to-real transfer for contact-rich manipulation. Nevertheless, the observed gains during early-stage fine-tuning demonstrate its potential when combined with careful reward design and early stopping. Future work may investigate improved training strategies, such as regularization of delta actions, reward alignment, and adaptive fine-tuning schemes, to fully realize the promise of delta action models for sim-to-real transfer.

V. Conclusion

This work studies a delta action modeling approach for narrowing the sim-to-real gap in a contact-rich peg insertion task. The results demonstrate that the proposed method is able to capture nontrivial dynamic discrepancies between simulation and "reality." In particular, the delta action model consistently achieves higher trajectory reproduction rewards than the zero-delta-action baseline, indicating that it successfully learns meaningful action-level corrections that reduce the mismatch between generated and reference trajectories. These findings validate the effectiveness of delta action modeling as a mechanism for encoding sim-to-real dynamics differences in a data-efficient and physically consistent manner.

At the same time, policy fine-tuning in the delta-action-augmented environment does not consistently translate into improved performance in "reality." Although early-stage fine-tuning yields the highest success rate on the small peg hole, continued fine-tuning leads to performance degradation. This behavior suggests that the learned delta action model introduces additional bias and noise, and that excessive fine-tuning causes the policy to overfit to the augmented simulation dynamics rather than improving generalization. As a result, convergence in the augmented environment does not necessarily correspond to optimal performance in "reality."

Overall, these results indicate that delta action modeling alone is insufficient to guarantee robust sim-to-real transfer for contact-rich manipulation tasks. Nevertheless, the observed improvements at early fine-tuning stages highlight its clear potential when combined with appropriate training strategies. Future work will focus on improving the stability and generalization of delta action models through techniques such as regularization of delta action outputs, alignment of reward functions between delta action training and policy optimization, and adaptive fine-tuning schemes with early stopping. With these enhancements, delta action modeling may serve as a powerful component for improving sim-to-real transfer in challenging robotic manipulation scenarios.

References

[1] T. He et al., "Aligning Simulation and Real-World Physics for Learning Agile Humanoid Whole-Body Skills," 2025.

[2] J. Hwangbo et al., "Learning Agile and Dynamic Motor Skills for Legged Robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, 2019.

[3] J. Tan et al., "Sim-to-Real: Learning Agile Locomotion for Quadruped Robots," arXiv preprint arXiv:1804.10332, 2018.

[4] X. B. Peng et al., "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization," in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 3803–3810, 2018.

[5] F. Ramos et al., "BayesSim: Adaptive Domain Randomization via Probabilistic Inference for Robotics Simulators," arXiv preprint arXiv:1906.01728, 2019.

[6] G. B. Margolis et al., "Rapid Locomotion via Reinforcement Learning," arXiv preprint arXiv:2205.02824, 2022.

[7] J. Tobin et al., "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

[8] T. He et al., "Learning Human-to-Humanoid Real-Time Whole-Body Teleoperation," arXiv preprint arXiv:2403.04436, 2024.