DynamicManip: Enabling Dynamic Manipulation from a Single Static Demonstration

Anonymous Author(s)
Affiliation
Address
email

Figure 1: **DynamicManip** is able to augment diverse dynamic manipulation data from a single static task demonstration in the real world. Leveraging the augmented data, our dynamic-aware adaptive policy enables generalized dynamic manipulation with efficient inference.

Abstract

Dynamic manipulation is a critical capability for robots operating in complex and dynamic environments, where robots must interact with objects that are moving or require rapid adjustments. However, learning models for dynamic manipulation tasks faces two major challenges: 1) the combinatorial complexity of dynamic scenarios leads to substantial data requirements, 2) rapid variations in dynamics require real-time and accurate policy execution. In this paper, we propose DynamicManip to address these challenges through an efficient data augmentation pipeline and a low-latency imitation policy. We first propose a static-to-dynamic augmentation pipeline that synthesizes diverse dynamic manipulation demonstrations from a single static demonstration. Second, we introduce a dynamic-aware adaptive policy that adaptively adjusts its inference frequency according to task dynamics, enabling responsive and effective dynamic manipulation. Third, we build a dynamic task manipulation benchmark, which includes diverse dynamic tasks with an automatic evaluation system for scalable and consistent assessment. Extensive experiments in both simulation and the real world demonstrate that DynamicManip not only provides significant improvements in data efficiency but also achieves superior performance in dynamic manipulation tasks, with higher success rates and better dynamic responsibility.

1 Introduction

A long-term goal in robotics is to deploy robots in everyday human environments[**? ? ?**], where they can assist people and integrate into our daily workflows. To realize this vision, robots must handle dynamic tasks in which the target objects are in motion, such as catching a flying ball or stopping a glass about to slip off the table[**?**]. These tasks require the ability of a robot to continuously perceive, track, and respond to objects in motion through closed-loop perception-action cycles.

However, dynamic manipulation remains significantly challenging compared to static tasks. Two fundamental challenges stand in the way[**?**]. First, collecting data for dynamic tasks is inherently difficult, owing to the combinatorial explosion of object velocities, trajectories, and robot response timings [**? ?**]. Second, dynamic manipulation demands real-time model inference to cope with rapid state variations [**? ? ? ? ?**]. Traditional methods typically trade inference speed for precision and train on quasi-static datasets, making them incompetent for dynamic tasks [**?**].

To overcome these limitations, we present **DynamicManip**, which addresses the challenges of dynamic manipulation learning through (1) a static-to-dynamic data augmentation pipeline to synthesize diverse dynamic data from a single static data while preserving underlying physical consistency

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

and robot behavior patterns. (2) a dynamic-aware adaptive policy that accelerates inference according to dynamic manipulation situations while preserving precision.

For data augmentation, the augmentation pipeline enables flexible modification of object dynamics and robot motion. Specifically, the pipeline first builds the geometric reconstruction of the manipulation environment, allowing point cloud rendering of both objects and the robot under arbitrary configurations. It then edits object trajectories and performs motion planning to synthesize corresponding closed-loop robot actions, ensuring physically consistent dynamic interactions.

Based on the augmented data, the proposed policy enables efficient inference by adaptively adjust the policys inference strategy. Specifically, we construct a dynamic-stage awareness that indicates whether high or low computational effort is required. This stage awareness is heuristically and automatically annotated by our augmentation pipeline, precisely derived from the underlying objectrobot interaction dynamics. Then we train a lightweight auxiliary head to predict the dynamic-stage awareness, which is used during inference to reduce diffusion denoising steps or increasing chunk-based execution adaptively.

To support our framework, we introduce the DynamicManip Benchmark, comprising diverse dynamic manipulation tasks with an automatic evaluation pipeline for comprehensive experiments. Our benchmark is built on the RoboTwin 2.0 platform [**?**], where we provide highly configurable dynamic task environments for data collection and an automatic evaluation system that reliably measures task success rates, response times, and robustness to variations in object motion.

Comprehensive experiments in both simulation and the real world demonstrate that our approach systematically scales a single static human demonstration into a diverse dynamic dataset, drastically reducing both the time consumption and human labor required by traditional continuous teleoperation. Moreover, the proposed policy achieves substantially faster execution speeds while maintaining high precision, enabling highly responsive closed-loop control in dynamic scenarios with significantly higher success rates.

2 Related Work

2.1 Robot Manipulation

Robot manipulation has been a long-standing area of research and a central pillar of the robotics community, widely regarded as one of the most tangible pathways toward general-purpose embodied intelligence[**? ? ?**]. Recent methods such as Imitation Learning (IL) [**? ? ? ? ?**] and Vision-Language-Action (VLA) [**? ? ? ? ? ?**] have achieved impressive results on learning the underlying distribution from large-scale expert demonstrations and predicting action sequences conditioned on observations. However, these methods primarily focus on static tasks where the target objects remain stationary during manipulation. In these scenarios, the demand for data scale is relatively modest and the model inference speed is not held to stringent requirements.

2.2 Dynamic Tasks

Dynamic manipulation is pervasive in everyday and industrial environments, from catching a slipping glass to sorting products on moving conveyors. Such tasks require a robot to continuously track and respond to objects whose state evolves unpredictably, demanding rapid perception, predictive planning, and low-latency action generation in closed-loop perception–action cycles. Despite its importance, general dynamic manipulation under uncertain motion and fine contact constraints remains underexplored. Existing methods largely focus on static settings or structured scenarios with predictable motion, such as DBC-TFP [**?**] and GEM [**?**] in conveyor-like environments. Concurrent VLA approaches, including RDT-2 [**?**], RTVLA [**?**], and VLASH [**?**], demonstrate real-time interaction with fast-moving targets but operate with large contact margins and lack precise manipulation. Although some work like DynamicVLA[**? ? ?**] try to address this problem, a significant gap remains between current capabilities and the demands of real-world dynamic manipulation.

2.3 Data Collection and Augmentation

High-quality and diverse demonstration data is critical for learning robust manipulation policies[**? ? ? ? ? ?**], yet collecting such data remains expensive[**? ? ? ?**], especially for dynamic tasks

Figure 2: The pipeline of our static-to-dynamic data augmentation. From a single static demonstration, we localize the manipulation object and target and reconstruct their geometry. We then extract action-based keyframes, where labels such as start, grasp, interact, and recover only serve as intuitive annotations for segmenting the source motion into reusable pieces. Dynamic trajectory editing then organizes stages such as Dynamic Target Alignment (Φ_{dyn}) to synthesize diverse trajectories, which are generated through motion planning and replaying local interaction segments.

with large combinatorial variation[**? ? ?**]. Data augmentation has therefore emerged as a promising direction to scale limited demonstrations[**?**]. Existing approaches can be broadly categorized into several groups. 2D visual augmentation methods (e.g., GenAug [**? ?**]) modify appearance through image-level transformations but do not enforce 3D consistency. Implicit 3D reconstruction methods (e.g., RoboSplat [**?**]) improve visual fidelity via neural scene representations, yet typically do not adapt action trajectories accordingly. Geometric augmentation approaches (e.g., RoCoDA [**?**]) apply SE(3)-equivariant transformations in state space but are largely limited to rigid global transformations. Simulation-based methods (e.g., FAR-Dex [**?**]) can generate physically plausible trajectories but may suffer from sim-to-real discrepancies. More recent work such as DemoGen [**?**] explores trajectory editing for tabletop tasks, but only generates quasi-static variations and does not address the temporal dynamics critical for dynamic manipulation. In contrast, our method generates diverse dynamic trajectories only using minimal static demonstrations.

3 Method

3.1 Overview

We study dynamic manipulation from extremely limited demonstrations, where the robot must learn to interact with moving objects while maintaining low perception–action latency. DynamicManip addresses this problem with two coupled components: a static-to-dynamic augmentation pipeline that converts a single static demonstration into diverse dynamic episodes, and a dynamic-aware imitation policy that adjusts its inference behavior according to the current interaction phase.

3.2 Efficient Data Augmentation

The core insight behind our augmentation pipeline is that a static demonstration already specifies *how* the robot should interact with the object, while the dynamic variations mainly come from *where* and *when* the interaction happens. Therefore, instead of collecting dynamic demonstrations directly, we preserve the local interaction patterns in the source episode and edit the object motions, phase anchors, and transition targets to synthesize new dynamic episodes. Given a single source episode $\mathcal{E}_{\text{src}} = \{(\mathbf{p}_t, \mathbf{e}_t, \mathbf{q}_t)\}_{t=0}^{T}$, where $\mathbf{p}_t \in \mathbb{R}^{N\times 3}$ is the scene point cloud, $\mathbf{e}_t \in SE(3)$ is the end-effector pose, and $\mathbf{q}_t \in \mathbb{R}^d$ is the robot state, our pipeline generates K augmented dynamic episodes through geometric reconstruction, keyframe extraction, and dynamic trajectory editing.

Geometric reconstruction. A key challenge in editing single-view demonstrations is that raw point clouds are incomplete and view-dependent: after spatial transformation, previously occluded object surfaces are partly missing from the recorded observation. To avoid such geometric artifacts, we reconstruct both the manipulated object and the robot geometry before trajectory editing. For the object, we generate a CAD mesh from a single RGB image and align it to the observed object point cloud $\mathcal{P}_o$ using OBB-based coarse initialization followed by multi-resolution ICP refinement [**?**]. The aligned mesh is then resampled into a canonical object point cloud, which can be transformed coherently under arbitrary spatial edits. For the robot, instead of relying on noisy and partially observed sensor points, we render the robot point cloud from its URDF model and joint configuration $\mathbf{q}_t$ via forward kinematics: $\mathcal{P}_t^{\text{robot}} = \mathrm{RenderRobot}(\mathrm{URDF}, \mathbf{q}_t)$. Together, the reconstructed object point cloud and the rendered robot point cloud provide a complete and consistent scene representation for subsequent trajectory editing.

Keyframe Extraction. A single static demonstration contains the local interaction priors needed for dynamic task synthesis, but these priors must be separated into reusable segments. We therefore ask the user to annotate only a small set of keyframes, such as object-contact moments, post-contact states, and task-specific transition points. These keyframes define the boundaries of reusable phases and provide anchor frames for later trajectory editing.

Figure 3: **Overview of the dynamic-aware adaptive policy.** During training, stage labels automatically extracted from augmented trajectories provide auxiliary supervision for stage prediction. During deployment, the predicted stage is used to adaptively regulate the policy inference frequency according to the current task dynamics.

Dynamic trajectory editing. After geometric reconstruction and keyframe extraction, we augment the single static demonstration by editing, planning, and recomposing its annotated phase segments. For the k-th augmented episode, we sample a set of task-specific augmentation parameters $\xi_k \sim \mathcal{D}$, which control variations in object poses, object motions, phase anchors, and transition goals. Guided by ξ_k, we synthesize new dynamic trajectories by instantiating and composing the following four distinct phases:

Static Object Acquisition ($\mathbf{\Phi}_{\text{static}}$) This phase corresponds to the trajectory segment where the robot manipulates a static object in the single demonstration, such as grasping a hammer or picking up a basket. Since the object is static, this segment can be augmented by applying a rigid transformation $T_k^{\text{static}} \in SE(3)$ to the entire phase: $\mathbf{p}'_t = T_k^{\text{static}}\mathbf{p}_t$, $\mathbf{e}'_t = T_k^{\text{static}}\mathbf{e}_t$. Such translations and rotations provide strong spatial generalization for static object handling.

Dynamic Target Alignment ($\mathbf{\Phi}_{\text{dyn}}$). Dynamic object manipulation is difficult to demonstrate through teleoperation, as it requires precise timing and smooth tracking of moving objects. Instead of requiring dynamic demonstrations, we only use a static terminal keyframe that specifies the desired object-frame relative pose T^*_{rel} of the end-effector. Given a synthesized object motion $T^k_{\text{obj}}(t)$, the desired end-effector pose is defined as $\mathbf{e}^k_{\text{goal}}(t) = T^k_{\text{obj}}(t)T^*_{\text{rel}}$. A motion planner then generates a smooth trajectory to reach these object-conditioned poses while satisfying robot constraints. Compared with teleoperated data, the planned trajectory avoids human-induced jitter and better utilizes the robot's reachable workspace.

Contact Interaction Execution ($\mathbf{\Phi}_{\mathbf{inter}}$). This phase contains the local contact behavior between the end-effector and the object, such as closing the gripper during dynamic graspingr or striking a target with a hammer. We represent the interaction segment relative to an anchor frame A and replay it under a new anchor A_k: $\mathbf{e}'_t = A_k A^{-1}\mathbf{e}_t$. By reusing this local interaction pattern, the robot can perform dynamic interactions with objects at different positions and orientations.

Pose-Conditioned Transition ($\mathbf{\Phi}_{\mathbf{trans}}$). This phase moves the robot to a recorded keyframe pose under a new spatial configuration. Given a recorded pose $\mathbf{e}_{\text{key}}$ and a pose offset $T_k^{\text{trans}} \in SE(3)$, we define the transformed target pose as $\mathbf{e}^k_{\text{key}} = T_k^{\text{trans}}\mathbf{e}_{\text{key}}$, and use motion planning to generate a trajectory from the current state to $\mathbf{e}^k_{\text{key}}$. This phase is used for rigid object transport, intermediate transitions, and returning to the reset configuration.

Flexible Composition. The four phase operators can be flexibly composed according to the interaction logic of each task. For example, a typical dynamic manipulation trajectory can be generated as $\Phi_{\text{static}} \rightarrow (\Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter}} \rightarrow \Phi_{\text{trans}})^L$, where L denotes the number of interaction cycles. Other tasks can be instantiated by reordering, skipping, or repeating different phase operators. By varying ξ_k during composition, our pipeline generates diverse dynamic trajectories from only one keyframe-annotated static demonstration. Importantly, because every timestep is produced by a known phase operator, the augmented dataset also provides stage labels that can be directly used to train a dynamic-aware policy.

3.3 Dynamic-aware Imitation Learning Policy

Based on the augmented data, our goal is to train a policy that is not only accurate but also responsive enough for dynamic manipulation. Specifically, we introduce a dynamic interaction stage awareness mechanism, which flexibly apply inference strategies based on the objectrobot interaction stage.

Dynamic-stage Awareness Learning To enable the policy aware of its inference stage, we leverage the generated pseudo-labels in our augmentation pipeline. Specifically, every augmented trajectory carries a stage label $\ell_t \in \{\Phi_{\text{static}}, \Phi_{\text{dyn}}, \Phi_{\text{inter}}, \Phi_{\text{trans}}\}$ for each timestep t, reflecting the interaction logic that produced it. We attach a lightweight classification head h_ϕ to the shared observation encoder. Given encoded features $\mathbf{f}_t$, the head predicts a stage distribution $\hat{\mathbf{p}}_t = h_\phi(\mathbf{f}_t)$ over

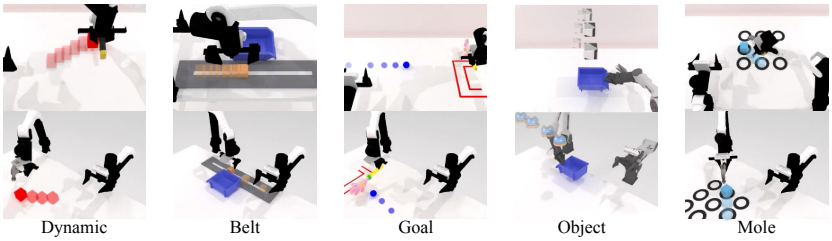

Figure 4: The visualization of the tasks in DynamicManip Simulation Benchmark.

the four stages, supervised by a cross-entropy loss:

$$\mathcal{L}_{\text{stage}} = -\frac{1}{T}\sum_{t=1}^{T} \log \hat{p}_t(\ell_t). \tag{1}$$

The total training objective combines the diffusion loss with this auxiliary term:

$$\mathcal{L} = \mathcal{L}_{\text{diff}} + \lambda\, \mathcal{L}_{\text{stage}}, \tag{2}$$

where λ balances action fidelity and stage awareness. This auxiliary objective encourages the encoder to aware of the current stage, which helps the policy adaptively adjust its inference strategy.

Dynamic-adaptive inference. At deployment, the trained stage classifier produces a per-timestep prediction $\hat{\ell}_t = \arg\max_k \hat{p}_t(k)$. Conditioned on this stage label, the policy dynamically adjust its computational allocation including the denoising schedule and the action-chunk execution depth.

For the denoising schedule, we adopt a DDIM[**?**] sampling[**? ?**] acceleration mechanism that treats each residual block output $\mathbf{y}(s)$ as a smooth function of the diffusion step index s and approximate it via a Taylor expansion around a recently computed anchor step s_0:

$$\mathbf{y}(s) \approx \sum_{k=0}^{K} \frac{1}{k!}\, \mathbf{y}^{(k)}(s_0)\, (s - s_0)^k, \tag{3}$$

where the derivatives $\mathbf{y}^{(k)}(s_0)$ are estimated via backward finite differences between consecutive full-computation steps (see Appendix for details). By skipping expensive forward passes at intermediate denoising steps, this approximation reduces inference latency with negligible accuracy.

We let the predicted stage $\hat{\ell}_t$ determine three scheduling parameters: the TaylorSeer computation interval τ (how often to perform full recomputation), the expansion order K, and the number of action-chunk steps n_{exec} to deploy per cycle. Different stages have fundamentally different latency–consistency trade-offs:

In Φ_{static}, we use a conservative schedule with frequent full recomputation and short execution chunks, since inaccurate early actions may lead to irreversible grasp failure.

In Φ_{dyn}, tracking fast-moving targets demands minimal closed-loop latency. We adopt an aggressive Taylor configuration with a large interval τ_{follow} and short action-chunk execution n_{short}, re-planning frequently so that corrections are applied at every cycle.

In Φ_{inter}, maintaining contact quality and motion smoothness requires temporal coherence. We switch to a conservative setting with a smaller interval τ_{interact} and a longer action-chunk prefix $n_{\text{long}} > n_{\text{short}}$, prioritizing consistency over raw speed.

In Φ_{trans}, returning has the least stringent timing requirements. We apply the most aggressive acceleration (largest interval, higher-order expansion) with full action-chunk execution, minimizing compute overhead during non-critical motions.

This stage-awared scheduling mechanism ensures reasonable computational resources allocation, which balances responsiveness , consistency , and efficiency.

Figure 5: The visualization of real-world robot setup and the objects used in our experiments.

4 DynamicManip Benchmark

To support the evaluation of dynamic manipulation capabilities, we introduce the DynamicManip Benchmark based on the RoboTwin 2.0 platform. We design 6 dynamic tasks environments for convenient data collection and policy evaluation, as illustrated in Figure 4.

Data Collection. We extend the RoboTwin 2.0 pipeline with two key modifications tailored for dynamic manipulation. First, a dynamic-aware heuristic planner continuously tracks moving targets, predicts future positions, and triggers grasps at optimal moments. Second, a contact-release mechanism transfers control from scripted trajectories to the physics engine upon physical contact. These additions enable the robot to actually track and interact with moving objects, ensuring high-quality data collection. Finally, we record expert demonstrations capturing head-camera RGB, proprioceptive states, and scene step deltas at 30 FPS.

Automatic Evaluation. We design a task-specific automatic evaluation system that determines success based on distinctive physical signatures (e.g., state-machine-based contact detection for tapping). After each evaluation, it generates a detailed diagnostic report including per-stage success flags, trajectory residuals, and specific failure reasons. To ensure accuracy and reliability, we rigorously validated the system through manual review. By inspecting 100 evaluation episodes per task, we confirmed the automatic evaluation achieves perfect agreement with human judgment.

5 Experiments

We comprehensively evaluate the proposed DynamicManip framework in both simulated and real-world dynamic environments. Our experiments are designed to answer the following key questions:

Q1: Regarding data efficiency, **can our augmented data match or exceed human demonstrations** while significantly reducing the collection time?

Q2: In dynamic scenarios, **how does our dynamic-aware policy perform** in terms of reaction speeds and task success rates compared to the baseline?

Q3: For the ablation study, **how much do mesh reconstruction and robot rendering contribute** to the final success of the policy?

5.1 Experiment Settings

Simulation Setup. We conduct simulation experiments to evaluate the performance of DynamicManip on the DynamicManip Benchmark. We use two ARX-X5 robotic arms with a pair of parallel grippers, a RealSense D435 head camera for visual perception. More details on the simulation environment can be found in the supplementary material.

Real-World Hardware Setup. We also perform real-world experiments to validate the effectiveness of our approach. We use a pair of AgileX Piper robots with parallel grippers and a RealSense L515 depth camera for visual perception.

Evaluation Metrics. We use the *Success Rate* (Suc.) to measure the performance, which measures the proportion of trials in which the robot successfully completes the dynamic task. For example, hitting a moving target, catching an object, or intercepting a goal-bound item;

Table 1: Comparison of models trained on human demonstrations vs. our synthetic data in simulation.

Setting	Dynamic Tapping	Belt Picking	Object Catching	Mole Whacking	Goal Blocking
1 demo	27	0	8	3	12
10 demos	15	5	54	21	46
50 demos	19	31	37	63	94
200 demos	55	85	50	83	87
1 demo $\xrightarrow{\text{aug.}}$ 50 eps	45	44	70	63	88
1 demo $\xrightarrow{\text{aug.}}$ 200 eps	52	82	68	80	87

Table 2: Comparison of models trained on human demonstrations vs. our synthetic data in real world.

Setting	Dynamic Tapping	Belt Picking	Boat Loading	Bottle Catching
1 demo	6.67	30	0.00	0.00
50 demos	68.9	70.3	3.33	73.3
1 demo $\xrightarrow{\text{aug.}}$ 50 eps	61.1	56.7	23.3	76.7
1 demo $\xrightarrow{\text{aug.}}$ 200 eps	87.8	73.3	66.7	83.3

Table 3: Time cost comparison for data collection and augmentation.

Method	Time Cost
Human Collection	
Teleop. (50 dyn. demos, 2 persons)	∼100 mins
Teleop. (200 dyn. demos, 2 persons)	∼400 mins
Teleop. (**1 static demo, 1 person**)	∼30 s
Automated Augmentation	
Aug. (50 eps, **human-free**)	∼7 mins
Aug. (200 eps, **human-free**)	∼30 mins

Training and Inference Details. Our framework adopts DDIM as the diffusion sampler, configured with 100 steps during training. At test time, we use dynamic-aware diffusion steps. We train for 300 epochs with a batch size of 128. Optimization uses an initial learning rate of 2.0×10^{-4}, decayed via a cosine schedule. All experiments are implemented in PyTorch on a single RTX 4090 GPU for training, while real-world inference is performed on a single RTX 4070 GPU. Additional details are available in the supplementary material.

Baseline. We compare our method against the state-of-the-art imitation learning policy, 3D-Diffusion-Policy [**?**], reproduced following the official RoboTwin defaults across every experiment, trained on the identical demonstration set, and evaluated under the same benchmark protocol, ensuring a fair comparison that isolates the contribution of our augmentation and inference components.

5.2 Can Augmented Data Match or Exceed Human Demonstrations (Q1)

To evaluate our augmentation pipeline's ability to overcome data scarcity, we compare the policy performance and collection cost of our generated data against manual demonstrations, using the DP3 baseline in both our simulated DynamicManip Benchmark and real-world physical setups.

Simulation Analysis. A single human demonstration inherently encapsulates rich physical priors and interaction logic. However, conventional direct training methods struggle to extract this knowledge efficiently, relying instead on massive data scaling to achieve satisfactory performance. As shown in Table 1, our augmentation pipeline breaks this paradigm. By fully unlocking the potential of just one static demonstration, our synthesized dynamic episodes enable the policy to achieve success rates matching or even exceeding those of models trained on extensive human datasets (e.g., 200 manual demos). This demonstrates that our geometric reconstruction and trajectory editing not only generate data as effective as human demonstrations, but frequently yield data of higher quality and broader dynamic coverage.

Real-World Validation and Time Efficiency. The superiority of our framework extends robustly to complex real-world environments (Table 2). Using only a single static demonstration, scaling the number of augmented episodes steadily boosts the policy's performance, easily surpassing the success rates of models trained on 50 expensive, teleoperated dynamic demonstrations. This proves that our augmentation strategy can effectively solve highly challenging dynamic tasks in the physical world. Furthermore, we achieve these higher success rates with drastically reduced collection costs (Table 3). While recording 50 manual dynamic demonstrations requires two synchronized operators and an hour of continuous teleoperationwhich is highly prone to human error and physical hardware damageour method demands only about *one minute* of human teaching. Because the subsequent augmentation process is fully automated and human-free, our approach virtually eliminates manual

Table 4: Latency and success rate comparison between baseline and adaptive dynamic-aware policy.

Method	Dynamic Tapping		Belt Picking		Object Catching		Mole Whacking		Goal Blocking	
	T.	Suc.	T.	Suc.	T.	Suc.	T.	Suc.	T.	Suc.
Baseline [**?**]	60	44.8	45.0	78	61.4	28	44.4	38	44.6	58
Ours	**56**	**32.5**	**32.9**	**80**	**36.0**	**26**	**30.6**	**44**	**33.9**	**88**

Table 5: Ablation study on key components of our dynamic-aware adaptive policy.

Object Mesh Recon.	Robot Recon.	Avg. Success Rate (%)
✓	✓	76.7
×	✓	10.0
×	×	6.67

labor. This allows the overall end-to-end time consumption to remain far below that of manual data collection, offering a highly efficient and scalable solution.

Summary. These findings decisively answer **Q1**: DynamicManip successfully synthesizes highly effective training data at a fraction of the cost, completely overcoming the data scarcity bottleneck. Beyond data quantity, our framework offers qualitative advantages: motion-planned trajectories yield smoother, safer, and jitter-free policy execution, while our diverse augmentation strategies endow the model with superior spatial generalization compared to raw human teleoperation. Ultimately, we deliver superior dynamic manipulation capabilities while reducing human effort from hours of tedious operation to a single minute of teaching.

5.3 Real-Time Performance of the Dynamic-Aware Policy (Q2)

Dynamic manipulation fundamentally hinges on minimizing perception-action latency to keep pace with unpredictable moving targets. Traditional diffusion-based models often sacrifice execution speed for generative precision, leading to critical tracking delays. As shown in Table 4, our dynamic-aware policy effectively breaks this trade-off, achieving a substantial and consistent reduction in inference latency across all benchmark tasks compared to the DP3 baseline. This accelerated closed-loop cycle empowers the robot to update its visual perception and correct trajectory deviations with significantly higher frequency, providing a foundation for robust reactive control.

This aggressive inference acceleration actively enhances dynamic tracking while largely preserving accuracy. Driven by low-latency responsiveness, our policy largely matches or even exceeds the success rates of the baseline, particularly in time-critical tasks like Goal Blocking and Belt Picking. This synergy of speed and accuracy is directly attributed to our stage-conditioned TaylorSeer mechanism, which intelligently maximizes inference speed during tracking phases while preserving precision during contact interaction. These results definitively answer **Q2**: our framework delivers significantly faster reaction speeds and superior closed-loop performance, ensuring effective dynamic manipulation in fast-paced scenarios.

5.4 Contribution of Mesh Reconstruction and Robot Rendering (Q3)

Spatial editing directly on raw, single-view point clouds is prone to severe geometric distortions due to occlusion and view dependency. As shown in Table 5, removing object mesh reconstruction substantially degrades the average success rate to 10.0%, as editing partial point clouds introduces a critical mismatch between augmented training data and inference observations. Further removing kinematic robot rendering worsens the performance to 6.67% by causing spatial inconsistencies in the robot's own embodiment. Conversely, our full pipeline yields a 76.7% success rate, demonstrating that both mesh-based object reconstruction and kinematic robot rendering are essential for maintaining the geometric coherence required for robust single-view dynamic augmentation.

6 Conclusion

In this paper, we present *DynamicManip* to address the critical bottlenecks of data scarcity and inference latency in dynamic robot manipulation. We introduce an efficient static-to-dynamic augmentation pipeline that synthesizes diverse, physically plausible demonstrations from only a single static demonstration, alongside a dynamic-aware adaptive policy that intelligently regulates inference frequency for responsive closed-loop control. Furthermore, we establish a comprehensive dynamic manipulation benchmark with an automatic evaluation system to provide a scalable assessment standard. Extensive experiments in both simulation and real-world settings demonstrate that our approach significantly improves data efficiency and task success rates while achieving faster inference than existing baselines. By effectively bridging the gap between minimal static data and complex reactive execution, our framework represents a robust and scalable step toward deploying robots in unstructured, fast-paced human environments.

Appendix

A Network Architecture & Training Details

A.1 Network Architecture

Our model adopts a diffusion-based framework that incorporates dynamic-aware perception and action generation capabilities. The architecture is designed to process multi-modal observations and generate precise robot actions through a denoising diffusion process.

A.1.1 Input modalities.

The model takes three types of inputs at each timestep:

`Point clouds:` Single-view point clouds serve as high-overload visual observations, denoted as $\mathbf{o}_t^h \in \mathbb{R}^{N\times 3}$, where $N = 2048$ points are obtained via Farthest Point Sampling (FPS). We define the observation horizon as t_o and in our experiments, we set $t_o = 6$.

`Proprioception:` Low-overload robot state observations $\mathbf{o}_t^l \in \mathbb{R}^M$ capture joint positions and gripper configurations across a temporal window. The proprioception shares the observation horizon as t_o.

Point Cloud Encoder. We first crop the point cloud using a 3D bounding box and downsample to $N = 2048$ points via FPS. We employ a PointNet encoder to independently encode each of the K_{high} sampled frames, then concatenate the features along the channel dimension to obtain the final visual representation $\mathbf{f}^h$.

State Encoder. Proprioceptive observations are encoded *per frame* with a lightweight MLP. For each observation step, the robot state vector is projected into a state feature token via the MLP. sThe state feature is then concatenated with the point-cloud feature at the same timestep.

A.1.2 Loss Function

We employ FiLM [**?**] conditioning to inject the conditional feature f_c into the denoising network, which predicts the noise associated with the robot action $\mathbf{a} \in \mathbb{R}^{k_a}$, where k_a denotes the action dimension specific to the task. The primary training objective minimizes the denoising score matching loss:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\mathbf{a}_0,\boldsymbol{\epsilon}\sim\mathcal{N}(0,\mathbf{I})}\left[\|\boldsymbol{\epsilon} - \epsilon_\theta(\mathbf{a}_t, t)\|^2\right], \tag{4}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ is the Gaussian noise, and the network ϵ_θ is trained to predict the added noise given the noisy action $\mathbf{a}_t$ and the diffusion timestep t. We employ DDIM [**?**] for inference sampling.

Dynamic-Stage Awareness Learning Every augmented trajectory carries a stage label $\ell_t \in \{\Phi_{\text{static}}, \Phi_{\text{dyn}}, \Phi_{\text{inter}}, \Phi_{\text{trans}}\}$ at each timestep t, reflecting the interaction logic that produced it. We leverage these labels as free supervision signals by attaching a lightweight classification head h_ϕ to the shared observation encoder. Given the fused feature f_{lh} processed through a dense fusion network ϕ_{dense}, the head predicts a stage distribution $\hat{\mathbf{p}}_t = h_\phi(\phi_{\text{dense}}(f_{lh}))$ over the four stages, supervised by a cross-entropy loss:

$$\mathcal{L}_{\text{stage}} = -\frac{1}{T}\sum_{t=1}^{T} \log \hat{p}_t(\ell_t). \tag{5}$$

The dense fusion network ϕ_{dense} enhances feature representations that are subsequently used by the diffusion decoder, while h_ϕ is intentionally kept as a lightweight single-layer MLP to mitigate the risk of overfitting the auxiliary head. The total training objective combines both losses:

$$\mathcal{L} = \mathcal{L}_{\text{diff}} + \lambda\mathcal{L}_{\text{stage}}, \tag{6}$$

where $\lambda = 0.1$ balances action fidelity and stage awareness. This auxiliary objective encourages the encoder to learn representations that disambiguate interaction phases, which are later exploited at inference time without any additional computation.

A.2 Training Configuration

We train all models using the AdamW optimizer with learning rate 1×10^{-4}, betas $[0.95, 0.999]$, weight decay 1×10^{-6}, and batch size 128. Training proceeds for 1500 epochs with a cosine learning rate schedule and 500 warmup steps. We employ Exponential Moving Average (EMA) with momentum 0.9999 on model parameters, activated after the first training step.

The noise scheduler follows a squared cosine schedule with 100 diffusion timesteps during training, with $\beta_{\text{start}} = 0.0001$ and $\beta_{\text{end}} = 0.02$. For inference, we use DDIM sampling with 10 denoising steps to efficiently generate action sequences.

Data augmentation is minimal to preserve the precise geometric relationships required for manipulation: we only apply random translation within $\pm$2cm in the horizontal plane during point cloud preprocessing. All models are trained on a single NVIDIA RTX 4090 GPU.

A.3 Inference

A.3.1 Inference Sampling.

We employ DDIM for efficient inference sampling. The reverse diffusion process is formulated as:

$$\mathbf{a}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{\mathbf{a}_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(\mathbf{a}_t, t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\epsilon_\theta(\mathbf{a}_t, t), \tag{7}$$

where $\bar{\alpha}_{t-1}$ and $\bar{\alpha}_t$ are the cumulative noise schedule coefficients at timesteps $t-1$ and t, respectively. We use 10 denoising steps during inference for efficient real-time deployment.

A.3.2 TaylorSeer Acceleration

During inference, the diffusion model must solve the reverse denoising process iteratively, which requires multiple forward passes through the 1D U-Net. Each DDIM step involves computing activations for all residual blocks across three resolution levels, making inference computationally expensive and limiting the real-time control frequency of the policy. To address this bottleneck, we introduce a Taylor-based acceleration mechanism that replaces selective U-Net forward passes with Taylor series extrapolation during the denoising trajectory.

Principle. The core observation is that adjacent DDIM steps produce highly correlated intermediate features — the U-Net's residual block outputs vary smoothly as the noise level changes incrementally along the reverse trajectory. Let $f^{(l)}(t_i)$ denote the output of residual block l at denoising step t_i. We maintain a cache of Taylor coefficients computed via backward finite differences over consecutive steps:

$$c_0 = f^{(l)}(t_i), \quad c_1 = \frac{f^{(l)}(t_i) - f^{(l)}(t_{i-1})}{t_i - t_{i-1}}, \quad c_2 = \frac{c_1^{(i)} - c_1^{(i-1)}}{t_i - t_{i-1}}, \tag{8}$$

and so on up to the maximum order K. For a future step $t_{i+\delta}$ where we skip the full forward pass, the Taylor prediction is:

$$\hat{f}^{(l)}(t_{i+\delta}) = \sum_{j=0}^{K} \frac{c_j}{j!} (t_{i+\delta} - t_i)^j. \tag{9}$$

Effect. TaylorSeer reduces inference latency by avoiding redundant full U-Net evaluations at selected DDIM steps. In our implementation on a single NVIDIA RTX 4090 GPU, one standard DDIM denoising step with a full 1D U-Net forward pass takes approximately 46 ms, whereas one TaylorSeer extrapolation step only takes approximately 22 ms. This gives a per-step speedup of

$$\frac{46}{22} \approx 2.09\times, \tag{10}$$

corresponding to a 52.2% reduction in latency for each approximated denoising step.

For the 10-step DDIM sampler used during deployment, let m denote the number of denoising steps replaced by Taylor extrapolation. The total inference latency can be estimated as

$$T(m) = (10 - m) \cdot 46 + m \cdot 22 = 460 - 24m \text{ ms}. \tag{11}$$

Therefore, each Taylor step saves approximately 24 ms. The resulting speedup over the vanilla 10-step DDIM sampler is

$$S(m) = \frac{460}{460 - 24m}. \tag{12}$$

This acceleration is especially beneficial for real-time robot control, where the policy must repeatedly sample action trajectories under a strict control-frequency budget. Since Taylor operates only on intermediate U-Net features during inference, it does not modify the diffusion training objective, the DDIM update rule, or the policy output representation, making it a plug-and-play acceleration module for the deployed diffusion policy.

Computation Scheduling We do not apply Taylor approximation at every step. Instead, we adopt a periodic recomputation strategy that interleaves full forward passes with Taylor predictions. Specifically:

- **First step:** always computed fully to establish the initial anchor.
- **First k steps (default $k = 3$):** always computed fully, as the early denoising steps contain the most informative signal about the target action distribution.
- **Every τ steps (default $\tau = 5$):** computed fully to refresh the Taylor coefficients and prevent extrapolation drift.
- **All other steps:** replaced by Taylor prediction of order K (default $K = 1$, i.e., linear extrapolation).

With $\tau = 5$, $K = 1$, and $k = 3$ under $T = 10$ DDIM inference steps, 6 out of 10 steps use Taylor prediction while only 4 steps require full U-Net evaluation, yielding approximately $1.45\times$ reduction in forward passes per inference call.

Per-Stage Adaptive Scheduling Different interaction stages have distinct requirements for inference accuracy and re-planning frequency. We define four task stages corresponding to the data augmentation phases: Φ_{static} (static object acquisition), Φ_{dyn} (dynamic target alignment), Φ_{inter} (contact interaction execution), and Φ_{trans} (pose-conditioned transition). For each stage, the predicted stage classifier output $\hat{\ell}_t$ determines three scheduling parameters: the TaylorSeer computation interval τ, the Taylor expansion order K, and the number of executed action steps n_{exec} from the predicted action chunk. The scheduling parameters are listed in Table 6.

Stage	$\boldsymbol{\tau}$ (interval)	$\boldsymbol{K}$ (order)	$\boldsymbol{n}_{\text{exec}}$
Φ_{static}	5	1	6
Φ_{dyn}	7	1	2
Φ_{inter}	3	1	8
Φ_{trans}	8	2	8

Table 6: Stage-specific TaylorSeer and execution step scheduling. τ is the recomputation interval, K is the Taylor polynomial order, and n_{exec} is the number of action steps executed per inference call.

A.3.3 Execution Step Configuration

The policy operates in a receding horizon control loop. At each inference call, the model predicts an action chunk of length $H = 8$ (the *action horizon*) via DDIM sampling with $T = 10$ denoising steps. Only a subset of n_{exec} actions from this chunk are actually executed before the next inference call is triggered. The remaining actions are discarded, and a fresh prediction is computed from the updated observations.

In our framework, n_{exec} is adjusted per Table 6.

B Efficient Data Augmentation Pipeline

In this section, we provide a detailed description of the data augmentation pipeline used in our framework. The process begins with a single **static** demonstration, from which we systematically

synthesize massive and diverse augmented episodes featuring complex dynamic target movements. To guarantee seamlessly scale up the training dataset, our augmentation engine is parallelized across a cluster of four NVIDIA RTX 4090 GPUs. This fully automated pipeline encompasses several critical computational stepsincluding mesh generation, precise mesh alignment, dynamic trajectory editing, and point cloud post-processingwhich we describe in detail below.

B.1 Mesh Generation

To ensure the precise geometric reconstruction of the manipulation scene, we capture an RGB image of the manipulation object and generate a 3D mesh using a 3D generative model [**?**]. This generated mesh is then utilized in the subsequent mesh alignment step, where it is integrated to form a complete 3D representation of the object. This step is essential for accurate trajectory editing and the generation of realistic augmented episodes.

B.2 Mesh Alignment and Geometric Reconstruction

A crucial part of our data augmentation pipeline is the alignment of known CAD meshes to the observed point cloud. Since point clouds captured from a single viewpoint typically only contain visible surfaces, this step helps in reconstructing complete 3D assets, which are then used in subsequent stages.

B.2.1 Coarse-to-Fine Mesh Alignment

The mesh alignment is performed using a coarse-to-fine approach, where an initial alignment is followed by a refinement step to improve accuracy. The steps are as follows:

- **Coarse Alignment (OBB-based Initialization):** We begin by aligning the mesh to the observed point cloud using an Oriented Bounding Box (OBB). The OBB is computed to estimate the initial transformation between the mesh and the point cloud.
- **Fine Alignment (Iterative Closest Point - ICP):** After the coarse alignment, we apply a multi-resolution ICP algorithm to iteratively refine the alignment. ICP minimizes the distance between the points in the point cloud and the vertices of the mesh. This is achieved by solving for the transformation that best matches the observed points to the mesh at different resolutions.

The mathematical formulation of the ICP step is as follows:

$$\min_{\mathbf{T}} \sum_{i=1}^{N} \|\mathbf{p_i} - \mathbf{T}\mathbf{m_i}\|^2$$

where $\mathbf{p_i}$ is the point from the observed point cloud, $\mathbf{m_i}$ is the corresponding mesh vertex, and $\mathbf{T}$ is the transformation matrix that minimizes the point-to-point distance.

The visualization of the mesh alignment process is shown in Figure 6.

B.2.2 Canonical Point Cloud Generation

Once the mesh is aligned, the aligned mesh is resampled into a canonical point cloud that is spatially invariant. This canonical point cloud is used in subsequent steps and allows for geometrically coherent transformations during editing.

B.3 Dynamic Trajectory Editing

The core of our data augmentation pipeline lies in the *Dynamic Trajectory Editing* engine, which systematically decomposes a single static demonstration into four distinct physical phases: Static Object Acquisition (Φ_{static}), Dynamic Target Alignment (Φ_{dyn}), Contact Interaction Execution (Φ_{inter}), and Pose-Conditioned Transition (Φ_{trans}). Rather than relying on simple linear interpolations, our engine dynamically synthesizes and warps trajectories by coupling geometric registration, physical priors, and real-time motion planning. This ensures that the generated trajectories are not only physically feasible but also exhibit robust spatial generalization.

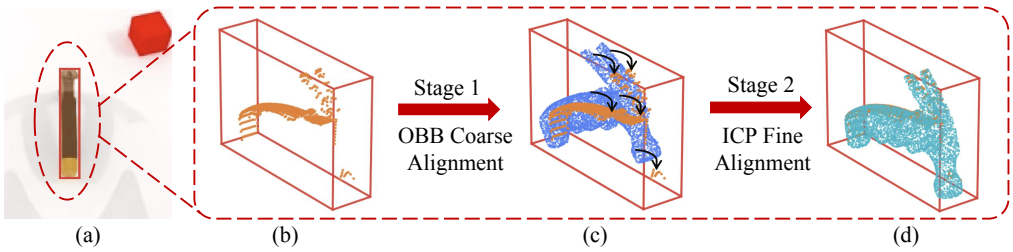

Figure 6: Visualization of the mesh alignment process. (a) shows the ego view image of a manipulation object, (b) shows the initial point cloud captured from the RGB-D sensor, (c) shows the coarse alignment using OBB, and (d) shows the refined alignment after applying ICP. The aligned mesh is then used for geometric reconstruction in the augmentation pipeline.

B.3.1 Static Object Acquisition (Φ_{static})

In tasks involving auxiliary tools (e.g., grasping a hammer or picking up a basket), the initial segment of the demonstration involves grasping or acquiring a static object. Because the object remains stationary during this phase, we parameterize this segment by annotating its start and end keyframes, denoted as $t_{\text{start}}^{\text{static}}$ and $t_{\text{end}}^{\text{static}}$, respectively.

To generalize this acquisition behavior to arbitrary initial poses in the augmented environment, we apply a rigid transformation $T_k^{\text{static}} \in \mathbb{SE}(3)$ to the recorded trajectory. The modified end-effector pose $\mathbf{e}'_t$ and object trajectory $\mathbf{p}'_t$ at any time step $t \in [t_{\text{start}}^{\text{static}}, t_{\text{end}}^{\text{static}}]$ are generated by replaying the source trajectory under this transformation:

$$\mathbf{p}'_t = T_k^{\text{static}} \mathbf{p}_t, \quad \mathbf{e}'_t = T_k^{\text{static}} \mathbf{e}_t \tag{13}$$

This rigid formulation effortlessly ensures spatial generalization, allowing the policy to reliably acquire tools or objects under diverse spatial displacements without requiring additional manual demonstrations.

B.3.2 Dynamic Target Alignment (Φ_{dyn})

Manipulating dynamic targets via manual teleoperation is notoriously difficult due to human cognitive limits, which frequently induce control latency and undesirable hand jitter. We elegantly circumvent the need for any dynamic demonstrations by formulating this phase as an autonomous, target-conditioned tracking problem.

This phase is defined using only a single terminal keyframe from the static demonstration to extract the target relative transformation T_{rel}^* between the end-effector and the object. Crucially, the target frame of the end-effector is not restricted to its physical center but can be customized with a task-specific offset T_{offset} (e.g., the tip of a hammer in a leverage task), which is extracted directly from the contact keyframe. Given the synthesized trajectory of the moving object $T_{\text{obj}}^k(t)$, the desired end-effector pose at any time step t is formulated as:

$$\mathbf{e}_{\text{goal}}^k(t) = T_{\text{obj}}^k(t) T_{\text{offset}} T_{\text{rel}}^* \tag{14}$$

To track this dynamic target, a reactive motion planner dynamically generates smooth trajectories under a maximum velocity constraint $v_{\max}$. To handle rapid variations in dynamics, the planner executes a high-frequency re-planning mechanism (e.g., every few steps) to continuously correct trajectory deviations. Furthermore, we map the planned trajectory to the robots reachable workspace, automatically cropping segments that violate physical kinematic limits. The tracking phase automatically terminates once the distance between the end-effector and the target falls below a predefined threshold:

$$\|\mathbf{e}(t) - \mathbf{e}_{\text{goal}}^k(t)\| < \epsilon \tag{15}$$

This design ensures that the generated trajectories are completely devoid of human-induced noise, offering superior tracking performance and maximizing workspace utilization.

B.3.3 Contact Interaction Execution (Φ_{inter})

The contact phase captures the precise, high-frequency interaction behaviors between the robot and the object, such as striking a target or executing a dynamic tap. Since these local contact patterns are highly sensitive to physical constraints and difficult to plan from scratch, we preserve the exact interaction dynamics from the source demonstration.

This phase is bounded by annotated start and end keyframes $[t_{\text{start}}^{\text{inter}}, t_{\text{end}}^{\text{inter}}]$. We model the local trajectory relative to a source anchor frame A, which represents the contact state in the demonstration. During augmentation, we warp this segment to a new target anchor A_k (determined by the terminal pose of the preceding Φ_{dyn} phase), translating and rotating the entire interaction sequence:

$$\mathbf{e}'_t = A_k A^{-1} \mathbf{e}_t, \quad \forall t \in [t_{\text{start}}^{\text{inter}}, t_{\text{end}}^{\text{inter}}] \tag{16}$$

By shifting the anchor frame, our engine replays the high-fidelity contact behavior at arbitrary spatial coordinates, seamlessly matching the physical constraints of different task locations while maintaining the original interaction dynamics.

B.3.4 Pose-Conditioned Transition (Φ_{trans})

After completing an interaction, the robot must safely transition to a target spatial configuration, such as returning to a neutral home position or moving to an intermediate waypoint (e.g., returning to the center in a whack-a-mole task).

Rather than replaying a rigid, collision-prone path, we leverage our motion planner to generate collision-free trajectories to a transformed keyframe pose. This phase requires only a single keyframe annotation to identify the target pose $\mathbf{e}_{\text{key}}$. To introduce spatial variation and prevent policy overfitting, we apply a stochastic pose offset $T_k^{\text{trans}} \in \mathbb{SE}(3)$ to the target configuration:

$$\mathbf{e}_{\text{key}}^k = T_k^{\text{trans}} \mathbf{e}_{\text{key}} \tag{17}$$

The motion planner then computes a smooth trajectory from the robot's current state to $\mathbf{e}_{\text{key}}^k$. The integration of the motion planner ensures that the robot can robustly transition across diverse configurations while completely avoiding obstacles and joint limits.

B.4 Task-Specific Phase Composition Logic

Unlike monolithic trajectory generation paradigms that frequently suffer from compounding errors during long-horizon execution, the core strength of *DynamicManip* lies in its ability to decouple complex manipulation into a precise sequence of modular phase operators $\{\Phi_{\text{static}}, \Phi_{\text{dyn}}, \Phi_{\text{inter}}, \Phi_{\text{trans}}\}$. By strategically recomposing these operators, our framework effortlessly instantiates diverse, long-horizon interaction patterns while maintaining strict physical plausibility. This section details the precise composition formulas and the underlying tracking logic used across both simulation and real-world benchmarks.

B.4.1 Simulation Benchmark Tasks

The interaction logic for simulation tasks is designed to robustly handle both infinite cyclic behaviors and high-precision spatial interceptions, ensuring zero-shot generalization to varying target velocities.

- **Dynamic Tapping:** To synthesize continuous striking motions without accumulating temporal drift, we employ a tightly coupled cyclic structure: $\Xi_{\text{tap}} = \Phi_{\text{static}} \rightarrow (\Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter}})^L \rightarrow \Phi_{\text{trans}}$. The policy first stably acquires the tool (Φ_{static}). Subsequently, the motion planner continuously guarantees real-time target alignment (Φ_{dyn}) before strictly executing the high-frequency striking template (Φ_{inter}). The final Φ_{trans} ensures a safe return to the reset configuration.
- **Mole Whacking:** Because the target's reappearance location is highly unpredictable, this task requires a full spatial reset after every interaction to prevent out-of-distribution tracking angles: $\Xi_{\text{mole}} = \Phi_{\text{static}} \rightarrow (\Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter}} \rightarrow \Phi_{\text{trans}})^L$. By forcing a Φ_{trans} step within the loop, the robot consistently restores an optimal central vantage point.

- **Object Catching:** To maximize dynamic responsiveness, this task strips away non-essential interactions: $\Xi_{\text{catch}} = \Phi_{\text{static}} \rightarrow \Phi_{\text{dyn}}$. The robot strictly focuses on grasping the container and aggressively tracking the predicted parabolic trajectory of the incoming object.

- **Goal Blocking:** The robot pre-positions the paddle and performs a rapid, reactive interception: $\Xi_{\text{block}} = \Phi_{\text{static}} \rightarrow \Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter}}$.

- **Belt Picking:** We decompose this sequential, long-horizon task to strictly separate transport from interaction, cleanly avoiding the compounding inaccuracies typical of continuous teleoperation: $\Xi_{\text{belt}} = \Phi_{\text{trans},1} \rightarrow \Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter},1} \rightarrow \Phi_{\text{trans},2} \rightarrow \Phi_{\text{inter},2}$. Crucially, $\Phi_{\text{trans},1}$ enforces a geometrically optimal pre-grasp posture (e.g., orienting the end-effector vertically downward) to facilitate stable visual tracking. After reactive alignment (Φ_{dyn}) and grasping ($\Phi_{\text{inter},1}$), $\Phi_{\text{trans},2}$ explicitly handles collision-free transport to the target bin, culminating in a precise release ($\Phi_{\text{inter},2}$).

B.4.2 Real-World Benchmark Tasks

In real-world deployment, the composition logic is strictly refined to enforce hardware safety, boundary reachability, and cycle-bounded execution.

- **Boat Loading (Reachability-Aware Tracking):** Dynamic targets often exceed the robot's strict kinematic limits. Traditional methods catastrophically fail when targets leave the workspace. To counter this, we introduce an augmented tracking phase (Φ^*_{dyn}) utilizing negative/boundary demonstrations: $\Xi_{\text{boat}} = \Phi_{\text{static}} \rightarrow \Phi^*_{\text{dyn}} \rightarrow \Phi_{\text{inter}} \rightarrow \Phi_{\text{trans}}$. Even when the boat temporarily moves out of the end-effector's reachable zone (e.g., points A or B), Φ^*_{dyn} actively projects the target to the nearest feasible boundary points (e.g., A^* or B^*). This reachability-aware extrapolation ensures the policy aggressively pursues the target within safe kinematic bounds until a valid interaction state naturally emerges, after which it executes the placement (Φ_{inter}).

- **Real-World Dynamic Tapping:** To prevent physical degradation of the hardware during real-world continuous testing, the cyclic striking is strictly bounded to a finite execution ($L = 3$): $\Xi_{\text{tap-real}} = \Phi_{\text{static}} \rightarrow (\Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter}})^3 \rightarrow \Phi_{\text{trans}}$.

- **Bottle Catching:** We utilize a highly streamlined tracking-to-grasp sequence to easily surpass human reaction limitations, ensuring minimal latency before the bottle falls: $\Xi_{\text{bottle}} = \Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter}}$.

- **Real-World Belt Picking:** To rigidly guarantee hardware safety and system repeatability between consecutive real-world trials, a mandatory terminal reset is appended: $\Xi_{\text{belt-real}} = \Phi_{\text{trans},1} \rightarrow \Phi_{\text{dyn}} \rightarrow \Phi_{\text{inter},1} \rightarrow \Phi_{\text{trans},2} \rightarrow \Phi_{\text{inter},2} \rightarrow \Phi_{\text{trans},3}$. Here, $\Phi_{\text{trans},3}$ effectively neutralizes any residual kinematic momentum, safely returning the arm to its strict zero-position.

B.5 Point Cloud Post-Processing

Once the trajectory is edited and the augmented episode is generated, we perform post-processing on the point cloud. The primary goal is to merge point clouds from various stages, ensuring that the final scene representation is cohesive and complete.

B.5.1 Point Cloud Merging and Cleanup

Point clouds from different stages (e.g., Grasp, Follow, Interact, Recover) are merged to form the final scene. We ensure that redundant points are removed, and the points are uniformly distributed in the final representation.

We use a k-d tree for efficient nearest-neighbor search during the cleanup process. For each point in the scene, we check if it is too close to an object or part of the robotic arm and remove it if necessary. The cleaning is based on a threshold distance δ, which is set to 2 cm experientially.

B.5.2 Final Point Cloud Formatting

After cleanup, the point cloud is formatted to comply with the HDF5 data structure, where each augmented episode is stored with its associated metadata (e.g., end-effector poses, joint actions, etc.). The final point cloud is stored as a 2048-point sample for each frame, ensuring consistency across the augmented dataset.

$$\text{Final Point Cloud} = \{\mathbf{p_1}, \mathbf{p_2}, \ldots, \mathbf{p_{2048}}\}$$

where each point $\mathbf{p_i} = [x_i, y_i, z_i, r_i, g_i, b_i]$ represents the 3D coordinates and color of a point in the scene.

Once the point clouds have been processed, the final augmented episodes are saved in HDF5 format. This format allows for efficient storage and retrieval of large datasets, and each augmented episode contains all the relevant data for training robotic systems.

- **Episode Data:** For each episode, we store the point cloud, end-effector pose, joint action, and episode boundaries.
- **Metadata:** Additional metadata includes task-specific parameters and transformation matrices used during augmentation.

C Benchmark Details

Overview

We construct a comprehensive benchmark for dynamic manipulation tasks, encompassing both simulation and real-world environments. The simulation benchmark covers 5 dynamic tasks with 200 expert demonstrations each, built on the RoboTwin 2.0 platform using the SAPIEN physics simulator. The real-world benchmark includes 4 dynamic tasks with 50 demonstrations each, collected using AgileX Piper robotic arms and an Intel RealSense L515 RGB-D camera.

C.1 Simulation Benchmark

Our simulation benchmark is built upon the RoboTwin 2.0 platform, using the SAPIEN physics simulator as backend.

C.1.1 Environment Setup

The simulation environment consists of serveral key components:

3D Assets

We use a diverse set of 3D assets for the manipulation objects, lerveraging both existing object datasets from RoboTwin and custom-designed assets. For the costmized assets, we created them using Hyper3D from a single RGB image, and then manually adjust the scale of generated meshes to ensure they are of appropriate size for the tasks. Then, we annotated each mesh with *contact points* and *function points* that are relevant for the specific manipulation tasks.

Automatic data collection system. The RoboTwin 2.0 platform provides a built-in data collection system that allows us to record expert demonstrations in a structured format. For each manipulation task, we disign a environment script that sets up the scene with the relevant objects and initializes the robot in a predefined starting configuration and discribes the task-specific interaction logic. During trajectory planning, the system plan the expert demonstration trajectories using CUROBO, which generates smooth and feasible trajectories that achieve the task objectives. During grasping, the system iteratively trys the annotated contact points and select the one that results in a successful grasp.

In data collection, we record multi-modal observations including RGB-D images from the head-mounted camera with 2048 points sampled from the point cloud, proprioceptive states including joint positions, end-effector poses, and gripper states, and language instructions describing the task from the task-specific templates. All data is recorded at 30 Hz control and observation frequency and saved in HDF5 format for efficient storage and retrieval.

Table 7: Sim data confidence intervals

Setting		Dynamic Tapping	Belt Picking	Object Catching	Mole Whacking	Goal Blocking
1 demo		[19%, 36%]	[0%, 4%]	[4%, 15%]	[1%, 8%]	[7%, 20%]
10 demos		[9%, 23%]	[2%, 11%]	[44%, 63%]	[14%, 30%]	[37%, 56%]
50 demos		[13%, 28%]	[23%, 41%]	[28%, 47%]	[53%, 72%]	[88%, 97%]
200 demos		[45%, 64%]	[77%, 91%]	[40%, 60%]	[74%, 89%]	[79%, 92%]
1 + aug	(50)	[36%, 55%]	[35%, 54%]	[60%, 78%]	[53%, 72%]	[80%, 93%]
	(200)	[42%, 62%]	[73%, 88%]	[58%, 76%]	[71%, 87%]	[79%, 92%]

C.1.2 Task descriptions.

Dynamic Tapping. The robot needs to continuously track a moving object and execute a timely tapping motion to contact the target while it is in motion.

Belt Picking. The robot needs to track an object moving on a conveyor belt, grasp it within the reachable workspace, and place it into a basket.

Object Catching. The robot needs to grasp a blanket and estimate the trajectory of a thrown object and move a basket to the predicted landing position to catch it.

Mole Whacking. The robot needs to interact with a 3×3 whack-a-mole setup, where a mole appears from one hole and moves randomly toward another while leaving a visible motion trace. The robot must anticipate the mole's future position and strike it at the appropriate time.

Goal Blocking. The robot needs to block an incoming ball from entering the goal by moving a handheld stick or paddle to intercept the ball's trajectory.

C.1.3 Evaluating Evaluation Metrics.

We develop task-specific automatic evaluation criteria for each scenario to ensure objective and reproducible performance measurement. For Dynamic Tapping, success is determined by detecting physical contact between the robot and the moving target. For Belt Picking, we evaluate whether the object is successfully transported and placed within a predefined target region. For Object Catching, we measure whether the object is correctly captured inside the basket based on spatial inclusion. For Mole Whacking, success is defined by whether the robot strikes the correct target location corresponding to the mole's appearance or predicted trajectory. For Goal Blocking, we verify whether the ball is successfully intercepted before entering the goal region.

All evaluations are conducted within the RoboTwin simulation environment, allowing efficient, scalable, and fully automated benchmarking. In addition, we record the robot's motion trajectories during task execution for further analysis of motion smoothness and dynamic tracking performance.

C.1.4 Confidence Intervals for Simulation Evaluation

We report 95% confidence intervals to quantify the uncertainty of success-rate estimates from finite simulation evaluations. For each setting, we evaluate the policy over $n = 100$ trials and compute the empirical success rate $\hat{p} = x/n$, where x is the number of successful trials. We use the Wilson score interval:

$$\mathrm{CI}_{95\%} = \frac{\hat{p} + \frac{z^2}{2n} \pm z\sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}}, \quad z = 1.96.$$

As shown in Table 8, increasing the number of training episodes generally improves task success, though the effect varies across tasks. The augmented-data settings show that augmentation from a single expert demonstration can provide useful additional training data.

Table 8: Real-world data confidence intervals

Setting		Dynamic Tapping	Belt Picking	Boat Loading	Bottle Catching
1 demo		[2%, 21%]	[17%, 48%]	[0%, 11%]	[0%, 11%]
50 demos		[51%, 83%]	[52%, 84%]	[1%, 17%]	[56%, 86%]
1 + aug	(50)	[43%, 76%]	[39%, 73%]	[12%, 41%]	[59%, 88%]
	(200)	[72%, 95%]	[55%, 86%]	[49%, 81%]	[66%, 93%]

C.2 Real-world Benchmark

Our real-world benchmark is designed to validate the effectiveness of the proposed data augmentation strategy in practical robotic scenarios. To this end, we construct a set of dynamic manipulation tasks and evaluate policies trained with our augmented data against strong baselines. Compared to simulation, real-world dynamic tasks introduce system latency and sensing noise, making precise timing and tracking more challenging. This places stricter requirements on the robustness and responsiveness of the learned policy, providing a more convincing evaluation of our approach.

C.2.1 Environment Setup

The real-world platform consists of two Piper robotic arms equipped with parallel grippers. Data collection is performed using a master–slave teleoperation setup with two GELLO devices. All demonstrations are recorded in the same format as in simulation to ensure consistency between training and deployment. The dataset is stored in HDF5 format and collected at 20 Hz.

For each timestep, we record the robot joint positions and velocities, end-effector poses, and gripper states. In addition, we capture synchronized RGB-D observations, which are further processed into point clouds with 2048 sampled points. Camera intrinsic and extrinsic parameters are also stored to support geometric reasoning.

C.2.2 Task Descriptions

We evaluate four representative dynamic manipulation tasks in the real-world setting:

Dynamic Tapping. The robot tracks a moving object and executes a precisely timed tapping motion to make contact.

Belt Picking. The robot tracks an object moving on a conveyor belt, grasps it at the appropriate moment, and places it into a basket.

Boat Loading. The robot grasps an object and places it onto a motorized boat moving on a water surface, requiring coordination with the boat's motion.

Bottle Catching. The robot intercepts and grasps a rolling bottle before it falls off the edge of the table.

C.2.3 Evaluation Metrics

For each task, we perform 30 trials and report the average success rate. Task-specific automatic evaluation criteria are defined as follows: for Dynamic Tapping, success is determined by whether the robot successfully contacts the moving target; for Belt Picking, whether the object is grasped and placed into the basket; for Boat Loading, whether the object is successfully placed onto the moving boat; and for Bottle Catching, whether the bottle is successfully intercepted and grasped before falling.

These metrics enable consistent and objective evaluation across all tasks.

C.2.4 Confidence Intervals for Real-world Evaluation

As shown in Table 8, we report 95% confidence intervals to quantify the uncertainty of success-rate estimates from finite real-world evaluations. For each setting, we evaluate the policy over $n = 30$ trials and compute the empirical success rate $\hat{p} = x/n$, where x is the number of successful trials.

D Limitations

Despite the promising results demonstrated by DynamicManip, our framework has several limitations that point to directions for future work.

Deformable objects. Our data augmentation pipeline relies on mesh-enhanced geometric reconstruction that assumes rigid objects with stable geometries. For deformable or articulation-enabled objects such as cloth, ropes, or spring-loaded mechanisms, the ICP-based alignment and rigid transformation assumptions break down. Editing the trajectory of a deformable object would require modeling its evolving shape and physical parameters, which remains an open challenge.

Rigid grasp assumption. The current pipeline attaches the grasped object to the end-effector via a fixed relative transform $\mathbf{T}_{\text{attach}}$ after contact. This assumption holds for tasks where the object does not slip or reorient relative to the gripper, but fails in scenarios requiring in-hand manipulation, rolling, or sliding contacts. Extending our method to handle non-rigid attachments or dynamic grasp maintenance would be a valuable extension.

Lack of dexterous tasks. Our benchmark tasks primarily involve parallel-jaw grippers and gross manipulation primitives such as tapping, catching, and blocking. We have not explored dexterous, multi-fingered tasks that require fine finger coordination or adaptive grasping strategies. Applying DynamicManip to dexterous manipulation would likely require augmenting the action space and rethinking the trajectory editing pipeline for multi-contact interactions.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the paper's contributions and scope, which are consistent with the theoretical and experimental results presented throughout the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The paper discusses limitations in a dedicated section in the appendix.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include theoretical results; it proposes a method and validates its effectiveness through experiments.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides sufficient details including training configurations, hyperparameters, data augmentation methods, and evaluation protocols needed to reproduce the main experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code and data are not yet publicly available, as they require further organization before release. And we will release the data and code after publication.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.

- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: All training and test details, including data splits, hyperparameters, and optimizer settings, are provided in the main text and the implementation details section in the appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Each experiment is run for 50 trials, and 95% confidence intervals are computed and reported as error bars for the main claims.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

Justification: The paper specifies the GPU types, memory requirements, and execution time for training and evaluation, sufficient to reproduce the experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conforms to the NeurIPS Code of Ethics. The work involves robotic manipulation in controlled environments with no human subject or data privacy concerns.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The paper discusses the positive societal impact of enabling robots to assist humans in everyday environments through dynamic manipulation in both the introduction and conclusion.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not pose such risks; all data is self-collected through robotic simulation and real-world experiments, and the models are domain-specific manipulation policies.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets including DP3 and RoboTwin 2.0 are properly cited with their original papers and used in accordance with their licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [No]

    Justification: The DynamicManip benchmark is introduced in the paper but not yet released alongside it. The code and data require further organization and will be open-sourced in a future release.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for writing assistance, editing, and formatting, and do not impact the core methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.