SegDiff: Segmented Trajectory Diffusion for Consistent and Adaptive Robot Manipulation

Haidong Cao[1,2], Wenjun Cao[1,2], Quanhao Li[1,2], Sicheng Xie[1,2], Zhiying Du[1,2], Jiaqi Leng[1,2], Zuxuan Wu[1,2†], and Yu-Gang Jiang[1,2]

[1]Institute of Trustworthy Embodied AI, Fudan University, China
[2]Shanghai Key Laboratory of Multimodal Embodied AI, China

Abstract. Imitation learning enables robots to acquire manipulation skills from demonstrations by mapping observations to actions. Existing approaches predict either short-horizon continuous action sequences or discrete keyposes. However, continuous prediction methods suffer from compounding errors due to short prediction horizons and struggle with multi-modal action distributions, whereas keypose-based methods necessitate an external planner, constraining real-time applicability. To address these challenges, we introduce **SegDiff**, a closed-loop visuomotor policy that integrates the strengths of both paradigms. SegDiff decomposes demonstrations into motion segments between keyposes and learns to predict the continuous trajectory from the current state to the next keypose, enabling long-horizon prediction with real-time refinement. Furthermore, we leverage the capability of diffusion models and DDIM inversion to propose a Dynamic Temporal Ensembling mechanism, which allows the policy to efficiently respond to dynamic environments and mitigate discontinuities caused by inconsistent multi-modal sampling. SegDiff demonstrates significant performance gains over existing approaches across various simulated and real-world scenarios, indicating its strong ability to reason over extended temporal dependencies while maintaining real-time adaptability and control stability.

Keywords: Imitation Learning · Robotic Manipulation · Diffusion Model · Temporal Ensembling

1 Introduction

Robotic manipulation is a critical skill for intelligent agents, yet developing robust policies remains challenging due to the complexity of the physical world. Imitation learning (IL) [2,32,35,36,47], which learns directly from expert demonstrations, has thus emerged as a powerful and pragmatic paradigm. Offline imitation learning (OIL) [1, 2, 7, 27] further improves data efficiency and safety by learning from fixed expert datasets without costly real-world interaction. Among OIL approaches, behavior cloning (BC) [2] remains the most fundamental and widely used paradigm. Traditional BC predicts the immediate action from the

† Corresponding author.

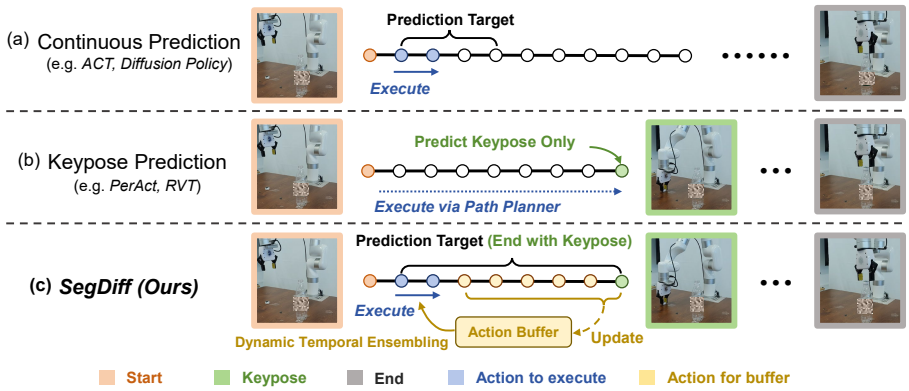

Fig. 1: Comparison of conventional continuous prediction methods, keypose prediction methods, and **SegDiff**. SegDiff combines the advantages of both paradigms by enforcing keypose constraints within continuous trajectory generation via **Segmented Trajectory Modeling**, while improving action consistency and real-time responsiveness through **Dynamic Temporal Ensembling**.

current observation, but even small deviations between the predicted and expert actions caused by perception noise or imperfect policy approximation can quickly accumulate and drive the robot into out-of-distribution (OOD) states.

Recent approaches [4, 5, 9, 24, 41, 48, 50, 53] extend the prediction horizon by predicting short action sequences as shown in Fig. 1(a), with some employing temporal ensembling [5, 53] between consecutive predictions to smooth motions. Although these methods improve short-term consistency, they still suffer from compounding errors over extended durations. Another major challenge arises from the multi-modal nature of the action distribution in robotic manipulation tasks. The same observation can correspond to multiple valid action modes such as grasping an object from different directions. Averaging across such modes can yield ambiguous actions and cause unsafe behaviors.

Keypose-based prediction methods [11, 13, 14, 18, 23, 38, 49] mitigate compounding errors by focusing on discrete salient points (*i.e.*, keyposes) and applying an external planner to generate motions toward the next keypose in an open-loop manner, as shown in Fig. 1(b). However, this open-loop design limits the adaptability to dynamic environments and constrains real-time applicability. While there are approaches that attempt to combine continuous action prediction and keypose prediction [42, 46, 51], they often rely on complex multi-stage architectures that increase inference latency and suffer from error propagation between modules. Furthermore, they typically struggle to handle the multi-modal nature of robotic actions, making them susceptible to mode switching and limiting their responsiveness in dynamic scenarios.

To address these challenges, we propose **SegDiff**, a closed-loop robot manipulation policy that seamlessly integrates the strengths of continuous prediction and keypose prediction. The core idea is to leverage keyposes as **prediction**

anchors and learn continuous motion to reach the next keypose. Specifically, we formulate the problem as predicting the complete trajectory from the current state to the next keypose, which we term **Segmented Trajectory Modeling** (as shown in Fig. 1(c)). This formulation directs learning toward task-critical states and alleviates the compounding errors common in continuous prediction.

During inference, SegDiff follows the Receding Horizon Control (RHC) [31] scheme, executing only the initial portion of each predicted trajectory and storing the remainder in an action buffer. Benefiting from Segmented Trajectory Modeling, the buffer and the newly predicted trajectory share the same initial state and both terminate at the next keypose. This inherent alignment improves the accuracy of next keypose prediction and enables detection of action mode deviations. Leveraging this, SegDiff employs a **Dynamic Temporal Ensembling** mechanism using DDIM inversion to validate and refine the action buffer. Outdated buffers are discarded and replaced by trajectories generated conditioned on the latest observation. Conversely, when the buffer remains valid, new trajectories are generated from the latest observation while preserving the original action mode, which facilitates more effective temporal ensembling with the existing buffer. This entire design enables SegDiff to adapt to dynamic environments while ensuring modal consistency across consecutive predictions. In summary, our contributions are as follows:

1) We propose **SegDiff** (Segmented Trajectory Diffusion), featuring a novel **Segmented Trajectory Modeling** approach that integrates keyposes as prediction anchors. This formulation not only serves as a training objective to guide the model toward task-critical states, but also acts as a crucial architectural enabler, laying the structural foundation for real-time action buffer refinement via DDIM inversion.

2) Building upon this structural design, we introduce a novel **Dynamic Temporal Ensembling** mechanism. By leveraging DDIM inversion, this mechanism enables the policy to seamlessly adapt to dynamic environments while strictly maintaining action mode consistency across consecutive predictions.

3) We demonstrate the superior and robust performance of SegDiff on two simulated benchmarks and five challenging real-world manipulation tasks, showing that SegDiff consistently outperforms existing advanced approaches.

2 Related Work

Keypose-Based Manipulation. Keypose-based policies reformulate the robotic manipulation task into predicting the next end-effector pose. Early methods [18, 20, 38] relied on pixel-level or voxel-level representations to locate the keypose, but incurred high computational cost and lacked real-time performance. Subsequent methods [6, 11–13, 23, 49] adopt representations such as neural fields or point clouds for improved spatial precision, yet still depend on external planners [21, 22, 25, 26] and struggle in dynamic environments. ChainedDiffuser [44] utilizes a diffusion model to generate intermediate trajectories, but its two-stage inference pipeline limits real-time adaptability. In contrast, our method predicts

the complete trajectory from the current pose to the next keypose at every inference step, enabling real-time adaptation while preserving the keypose constraint.

Continuous Action Prediction. Continuous prediction methods [4, 9, 17, 24, 28, 29, 33, 41, 45, 50, 53] operate under a sliding-window paradigm, predicting short-horizon action sequences at each step while paying limited attention to task-critical states, which makes them prone to compounding errors. Waypoint-based and stage-conditioned methods [3, 37, 40, 42, 46, 51] partially address this issue, but require extra annotations or incur multi-stage inference overhead. In contrast, our approach segments demonstrations using keyposes and predicts continuous trajectories toward the next keypose within a single end-to-end model.

Action Consistency and Temporal Ensembling. Temporal ensembling improves the smoothness of continuous predictions by aggregating outputs over time; however, abrupt mode switches can still produce unsafe actions. Recent approaches address this issue by adjusting denoising schedules [8, 10, 16] or incorporating optimization for consistency [5]. However, these methods typically assume that previous predictions remain reliable, which may not hold in dynamic environments. SegDiff addresses this challenge with Dynamic Temporal Ensembling, which leverages DDIM inversion to explicitly validate and refine the action buffer in real time, improving both mode consistency and responsiveness.

3 Method

3.1 Preliminaries

Offline Imitation Learning. We focus on offline imitation learning (OIL), where the objective is to learn a visuomotor policy solely from expert demonstrations without interaction with the environment. In this setting, the demonstration dataset is denoted as $\mathcal{D} = \{(o_t, s_t, a_t)\}_{t=1}^{N}$, where o_t represents the visual observation (*e.g.*, RGB image) at time step t, s_t denotes the robot state (*e.g.*, end-effector pose), and a_t is the expert action. The policy $\pi_\theta(a_t|o_t, s_t)$ is typically trained with the objective:

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{(o_t,s_t,a_t)\sim\mathcal{D}} \left[\|a_t - \pi_\theta(o_t, s_t)\|^2 \right]. \tag{1}$$

OIL methods can be broadly categorized into continuous prediction and keypose prediction. Continuous prediction methods replace the target a_t with a future action sequence $a_{t:t+L}$, improving long-horizon reasoning. During inference, they typically follow the Receding Horizon Control (RHC) scheme [31], executing part of the predicted sequence and re-planning with new observations. This setup naturally supports temporal ensembling [53], where the unexecuted portion of the previous sequence is blended with the latest prediction for more accurate and smoother actions. A representative approach is Diffusion Policy (DP) [9], which models the conditional distribution of continuous action sequences using a diffusion process [15]. Its training objective is formulated as:

$$\mathcal{L}_{\text{DP}} = \mathbb{E}_{a^0_{t:t+l},\,\epsilon,\,k} \left[\|\epsilon - \epsilon_\theta(a^k_{t:t+l}, k, o_{t-h:t}, s_t)\|^2 \right], \tag{2}$$

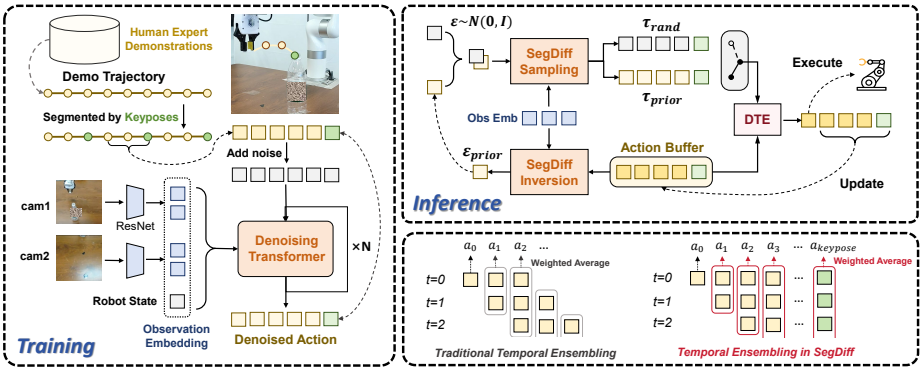

Fig. 2: Overview of SegDiff. During training, demonstration episodes are segmented by keyposes, and the model learns to predict the complete trajectory from the current state to the next keypose (**Segmented Trajectory Modeling**). During inference, **DDIM inversion** [39] generates a prior-informed candidate, allowing **Dynamic Temporal Ensembling (DTE)** to produce a trajectory that adapts to the current observation while maintaining temporal consistency for execution and buffer update.

where k denotes the diffusion step, l is the length of the predicted action sequence, and h indicates the number of historical observations used.

Keypose prediction methods [12,13,18,23,38,49], by contrast, replace a_t with the next keypose (*e.g.*, grasp pose), focusing on the most salient points within a trajectory while omitting intermediate motions.

DDIM Inversion. To enable prior-informed trajectory refinement within the diffusion policy framework, we employ DDIM [39], which provides a deterministic formulation of diffusion models that permits inversion of the sampling process. The deterministic form of DDIM [39] modifies the sampling process in DDPM [15] to:

$$x_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{1-\bar{\alpha}_{t-1}}\, \epsilon_\theta(x_t, t) \tag{3}$$

This deterministic process enables **DDIM inversion**, which maps a data point $\mathbf{x}_0$ to its corresponding noise vector $\mathbf{x}_K$ for prior-informed sampling and conditional generation.

3.2 Segmented Trajectory Modeling

As discussed in Sec. 3.1, continuous prediction models generate smooth actions but suffer from compounding errors inherent to their sliding-window paradigm. In contrast, keypose-based methods focus on the most important states in a demonstration (keyposes), making them particularly suitable for precise manipulation tasks; however, their open-loop nature requires external planners and prevents real-time adaptation. While approaches such as ChainedDiffuser [44] attempt to model intermediate paths, they remain limited in dynamic scenarios.

To bridge these paradigms, we introduce **Segmented Trajectory Modeling** (STM). By using keyposes as prediction anchors, SegDiff mitigates compounding errors (see the supplementary material for a formal derivation) while maintaining a closed-loop control paradigm. Specifically, instead of predicting isolated keyposes or fixed-horizon action sequences, the model is trained to predict a continuous trajectory that originates from an arbitrary timestep t and terminates at the subsequent keypose. We follow PerAct [38] to extract keyposes from expert demonstrations in which a keypose is defined as where the gripper state changes (*i.e.*, opening or closing) or the velocity of the end-effector approaches zero. We denote a specific human expert demonstration trajectory as $\{a_1, a_2, \ldots, a_T\}$, and extracted keyposes as $\{a_{k_1}, a_{k_2}, \ldots, a_{k_n}\}$. To ensure complete coverage of the entire trajectory, the last action a_T is included in the keypose sequence ($a_{k_n} = a_T$), and we define $a_{k_0} = a_1$. The trajectory can then be segmented into multiple sub-trajectories based on these keyposes. Each sample in the training dataset is therefore represented as:

$$\{(o_{t-h:t},\, s_t,\, a_{t:k_m}) \mid k_{m-1} \leq t < k_m\}, \tag{4}$$

where $1 \leq m \leq n$, and h denotes the duration of the observation history. In this way, we obtain demonstration sub-segments starting from an arbitrary timestep t and ending at the subsequent keypose a_{k_m}. Utilizing these sub-segments as training samples ensures that our model focuses on the keyposes of the demonstrations while simultaneously retaining an understanding of the intermediate motion process to support continuous action prediction.

We use a diffusion model to learn the conditional distribution of these sub-segments. Because each sub-segment has a variable length $k_m - t$, we interpolate all trajectories to a fixed length L, using cubic spline interpolation for translation and spherical linear interpolation for rotation. The training objective for the diffusion policy then becomes:

$$\mathcal{L}_{\text{SegDiff}} = \mathbb{E}_{a_{t:k_m},\, \epsilon,\, k}\left[\left\|\epsilon - \epsilon_\theta(\tilde{a}^k_{(t,k_m)}, k, o_{t-h:t}, s_t)\right\|^2\right], \tag{5}$$

where $\tilde{a}_{(t,k_m)}$ is the sub-segment of fixed length L obtained by interpolating the ground-truth sub-segment $a_{t:k_m}$, and k denotes the diffusion timestep sampled uniformly from $\{1, 2, \ldots, K\}$.

3.3 Dynamic Temporal Ensembling for Inference

During inference, SegDiff follows the RHC scheme (Sec. 3.1). At step t, the policy predicts a trajectory $\tau_t = \{a_t, \ldots, a_{t+L-1}\}$, executing only the first n_{exec} actions while storing the remainder in an **action buffer** b_t for temporal ensembling with later predictions. However, naive reuse of the action buffer introduces two challenges. First, due to the multi-modality in action distributions (Fig. 3(a)), newly predicted trajectories may belong to different modes, and directly switching between modes can lead to unsafe behavior (Fig. 4, left). Second, if environmental changes invalidate the action buffer, averaging the invalid buffer with the latest

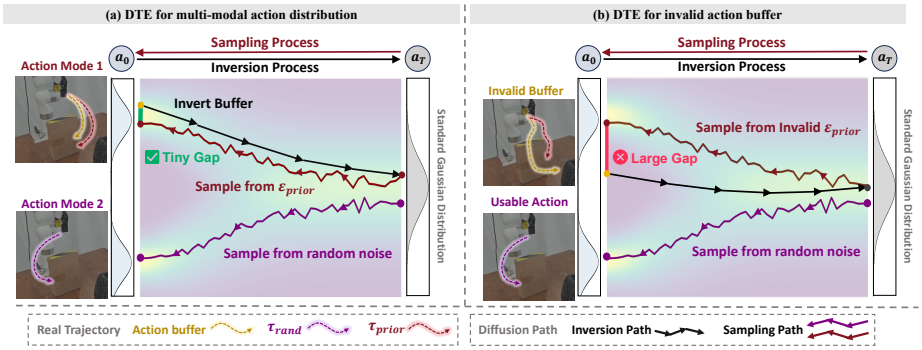

Fig. 3: Comparison between sampling from noise inverted from action buffer and from random noise. If there exists a large gap between action buffer and the action sampled from the inverted noise, we discard the action buffer as unusable.

prediction may cause the robot to respond too slowly to environmental changes and even enter out-of-distribution states, potentially resulting in task failure.

To address these challenges, we propose **Dynamic Temporal Ensembling** (DTE), a mechanism that leverages DDIM inversion to validate and refine the action buffer in real time. Unlike standard temporal ensembling which operates on a *sliding window* with shifting horizons, SegDiff instead anchors trajectory prediction to the next keypose (Sec. 3.2). Consequently, the action buffer b_t and the newly predicted trajectory τ_{t+1} share the same starting point and both terminate at the next keypose. This alignment facilitates more accurate keypose refinement as the keypose is repeatedly predicted before execution and provides the basis for using DDIM inversion to assess buffer consistency.

At each inference step t, we generate two candidates: (1) a **standard prediction** τ_{rand} sampled from Gaussian noise, representing the most responsive prediction; and (2) a **prior-informed prediction** τ_{prior}, obtained by first performing DDIM inversion on the action buffer b_t to acquire a noise vector ϵ_{prior}, and then denoising from ϵ_{prior} conditioned on the new observation. As illustrated in Fig. 3(a), if the action buffer is valid, denoising from the DDIM-inverted noise ϵ_{prior} (red sampling path) under the current observation yields a trajectory that remains consistent with the action mode of the buffer. Conversely, if the action buffer is no longer feasible under the updated observation, sampling from the inverted noise produces a trajectory that deviates significantly from b_t (Fig. 3(b)).

We quantify this mode-level consistency and feasibility using two RMSE-based metrics. The **buffer validity** is defined as $d_{\text{valid}} = \text{RMSE}(b_t, \tau_{\text{prior}})$, where a large d_{valid} indicates that the buffered actions are no longer feasible under the current observation. The **mode discrepancy** is defined as $d_{\text{mode}} = \text{RMSE}(\tau_{\text{rand}}, \tau_{\text{prior}})$, where a large d_{mode} indicates that the standard prediction deviates from the buffered action mode. To determine whether these deviations are significant, we introduce a threshold δ. The threshold δ is determined in a

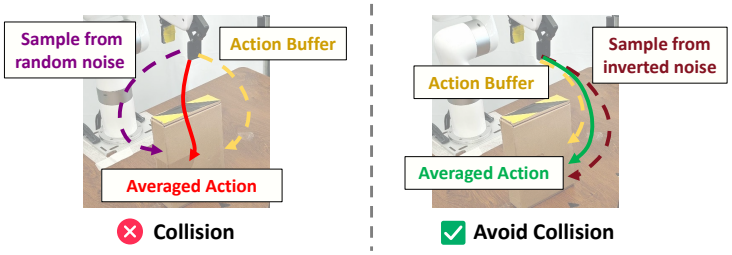

Fig. 4: Mode switching in model predictions can lead to undesirable actions (left), whereas sampling from the inverted noise provides a consistent alternative (right).

data-driven manner by computing the DDIM inversion reconstruction error on the training set and selecting the maximum observed RMSE.

Formally, the Dynamic Temporal Ensembling (DTE) mechanism computes the final action sequence τ_{DTE} as:

$$\tau_{DTE} = \begin{cases} \eta \cdot \tau_{\text{rand}} + (1-\eta) \cdot b_t, & d_{valid} \leq \delta \text{ and } d_{mode} \leq \delta \\ \eta \cdot \tau_{\text{prior}} + (1-\eta) \cdot b_t, & d_{valid} \leq \delta \text{ and } d_{mode} > \delta \\ \tau_{\text{rand}}. & d_{valid} > \delta \end{cases} \tag{6}$$

Here, $\eta \in [0, 1]$ is an update factor that controls the blending ratio between the new prediction and the action buffer.

The rule in Eq. 6 operates as follows. When the buffer is valid ($d_{\text{valid}} \leq \delta$) and the new prediction remains mode-consistent ($d_{\text{mode}} \leq \delta$), we prioritize τ_{rand} due to its stronger conditioning on the latest observation while blending it with the buffer. If the buffer is valid but a mode switch is detected ($d_{\text{mode}} > \delta$), we suppress the switch by utilizing τ_{prior} to preserve action-mode consistency. In contrast, when the buffer becomes invalid ($d_{\text{valid}} > \delta$), it is discarded entirely and τ_{rand} is fully adopted to rapidly adapt to environmental changes.

The resulting τ_{DTE} is then used for execution and buffer update: the first n_{exec} actions are executed, and the remaining trajectory is interpolated to length L to form b_{t+1}. This cycle repeats until the end-effector reaches the target keypose within a predefined tolerance δ_r. Regarding computational efficiency, the DDIM inversion step count is decoupled from the sampling process, which enables flexible configurations that balance efficiency and precision. For example, lightweight inversion can be combined with multi-step sampling to maintain trajectory accuracy without significant computational overhead (analyzed in Sec. 5.4). In addition, the sampling of τ_{rand} and τ_{prior} can be performed in parallel, resulting in negligible additional latency.

4 Experiment

4.1 Simulation Experiments

Benchmarks. We evaluate SegDiff on two simulation benchmarks: RLBench [19] and RoboMimic [30]. **RLBench** is built atop CoppeliaSim [34], where a Franka Panda robot manipulates the scene. We follow the task setup of CoA [51] and evaluate on both 10 representative tasks and the full RLBench-60 benchmark. RLBench demonstrations are generated via scripted policies, providing precise and consistent supervision. Each task contains 100 training episodes and 25 evaluation episodes. **RoboMimic** is another widely used benchmark for evaluating visuomotor policies. We conduct experiments on three tasks using the PH dataset, which consists of human expert demonstrations collected via teleoperation. This allows us to evaluate SegDiff on datasets with different characteristics,

Table 1: Success rates (%) on 10 RLBench tasks [19]. Results for ACT, DP, CoA and Octo are taken from [51] and [52]. We also reproduce the results of DP ourselves and report them as DP*. The results of SegDiff are averaged over 3 random seeds.

Task	ACT [53]	DP [9]	DP* [9]	CoA [51]	Octo [52]	SegDiff
Sweep Dust	**100.0**	**100.0**	**100.0**	92.0	80.0	**100.0**±0.0
Push Button	8.0	12.0	24.0	76.0	76.0	**90.7**±2.3
Stack Wine	56.0	56.0	56.0	80.0	52.0	**94.7**±2.3
Turn Tap	36.0	32.0	32.0	56.0	28.0	**74.7**±8.3
Open Drawer	52.0	44.0	52.0	88.0	84.0	**97.3**±2.3
Pick Up Cup	20.0	0.0	76.0	80.0	44.0	**90.7**±2.3
Take Lid	40.0	60.0	84.0	80.0	76.0	**90.7**±2.3
Press Switch	52.0	56.0	28.0	44.0	44.0	**68.0**±4.0
Reach Target	88.0	8.0	72.0	84.0	60.0	**97.3**±2.3
Open Box	36.0	48.0	20.0	76.0	**96.0**	62.7±6.1
Average	48.8	41.6	54.4	75.6	64.4	**86.7**±1.4

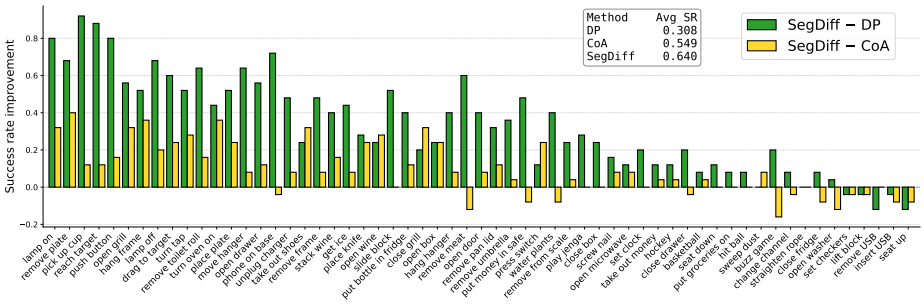

Fig. 5: Success rate improvement of SegDiff over DP/CoA. Detailed per-task results are provided in the supplementary material.

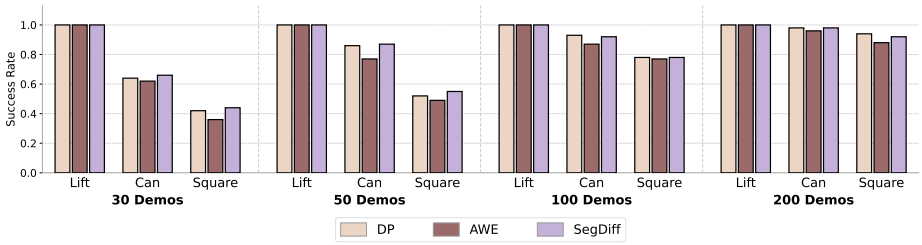

Fig. 6: Success rate on RoboMimic with varying numbers of demonstrations.

ranging from scripted demonstrations to human-collected trajectories, highlighting its robustness across diverse data sources.

Baselines. On RLBench, we evaluate our method against four representative visuomotor policies: ACT [53], Diffusion Policy (DP) [9], Chain-of-Action (CoA) [51], and a fine-tuned Octo policy [52]. Among the baselines, ACT and DP are the most commonly used baselines in prior work. CoA and the fine-tuned Octo are among the strongest previously reported methods on this benchmark. On RoboMimic, we compare our method against DP and AWE [37], where AWE enhances DP by filtering demonstration trajectories to emphasize critical waypoints. This comparison highlights the effectiveness of Segmented Trajectory Modeling over existing continuous diffusion-based approaches.

Implementation Details. Following prior work [9], SegDiff utilizes a separate ResNet18 encoder (not pretrained) for each camera view to process historical visual observations. The BatchNorm layers in the original ResNet blocks are replaced by GroupNorm [43], and the final global average pooling is substituted with spatial softmax pooling [30] to better capture spatial information. On RLBench, visual observations are captured by four RGB cameras (front, left shoulder, right shoulder, and wrist) and the images are rendered at a resolution of 128×128. On RoboMimic, observations are captured by a third-person camera and a wrist camera with a resolution of 84×84. Additional details are provided in the supplementary material.

Quantitative Results. Tab. 1 demonstrates that SegDiff consistently outperforms all baselines across 10 RLBench tasks, achieving an 11.1% higher average success rate and the best performance on 9 tasks. Notably, SegDiff yields over 30% improvement in precision-sensitive tasks (e.g., *Turn Tap*) and outperforms Diffusion Policy (DP) by 30% on average, validating the efficacy of Segmented Trajectory Modeling for high-precision manipulation. Furthermore, as shown in Fig. 5, SegDiff excels on the full RLBench-60 benchmark, outperforming DP and CoA by 34% and 9% respectively, demonstrating improved robustness and generalization. On RoboMimic (Fig. 6), SegDiff exhibits better data efficiency in low-demo regimes by mitigating compounding errors through keypose-anchored trajectory modeling, while remaining highly competitive as demonstrations increase. Detailed training configurations, task descriptions, and exhaustive results are provided in the supplementary material.

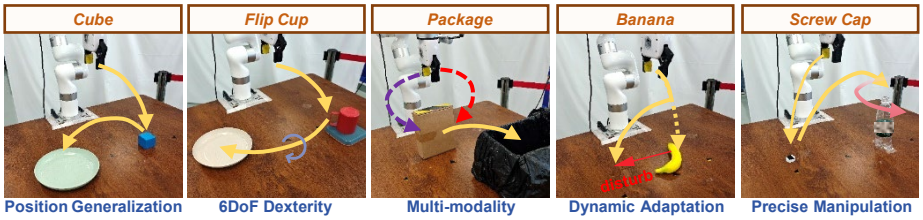

Fig. 7: Demonstrations and design of real-world tasks.

4.2 Real-World Experiments

Real-World Benchmarks. To comprehensively evaluate SegDiff, we design five real-world manipulation tasks (Fig. 7):

1) ***Cube:*** Grasp and place a cube onto a plate. Demonstrations are collected in a fixed area to evaluate robustness to **position generalization**.

2) ***Flip Cup:*** Grasp, flip, and place an inverted cup. The randomized handle orientation tests complex **6-DoF** manipulation.

3) ***Package:*** Place a cardboard into a box. Two distinct sets of demonstrations (grasping from left/right) were provided to evaluate the policy's ability to handle **multi-modal** action distributions.

4) ***Banana:*** Pick up a banana. The pose of the banana is actively perturbed during execution to test **real-time adaptation to dynamic scenarios**.

5) ***Screw Cap:*** Grasp and screw a cap onto a bottle. The initial position of the cap is randomized to test proficiency in **high-precision** operations.

Details. Our robotic platform is an xArm7 manipulator equipped with a gripper. Visual inputs are captured by two Intel RealSense D435i cameras (third-person and wrist-mounted), with images collected via human teleoperation and resized to 240×240. We use the same model architecture as described in Sec. 4.1.

Quantitative Results and Analysis. Real-world comparative results are presented in Tab. 2, where each task is evaluated with 10 trials. We specifically include DP (w/ TE) and DP (w/o TE) to illustrate the effect of **temporal ensembling (TE)**. Overall, SegDiff achieves performance superior or comparable to all baselines. Task-level analysis follows:

For ***Cube*** (**position generalization**), SegDiff achieves a performance comparable to existing approaches.

For ***Flip Cup*** (**6-DoF**), SegDiff achieves a higher success rate by accurately planning the motion to the precise grasp pose for the cup handle.

For ***Package*** (**multi-modality**), since DP (w/o TE) does not incorporate temporal ensembling, the robot may exhibit an initial period of dithering around the starting position; however, once the policy commits to a specific behavioral mode (*i.e.*, move to the left side), it can typically continue execution until successful completion. DP (w/TE) and ACT often fail due to the oscillation between different action modalities, leading to OOD states. SegDiff with Dynamic Temporal Ensembling prevents modality switching, achieving a success rate of 100%.

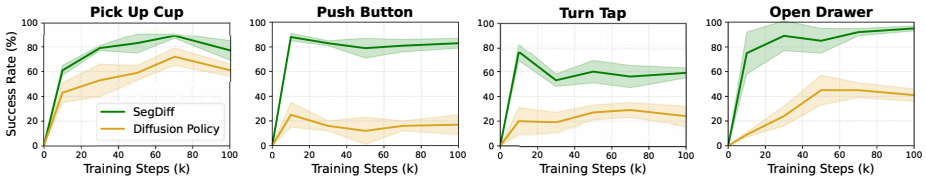

Fig. 8: Learning curves of SegDiff and Diffusion Policy across 4 tasks on RLBench.

For ***Banana*** (**dynamic adaptation**), DP (w/TE) and ACT are negatively influenced by the outdated action buffer. The short prediction horizon of DP (w/o TE) limits its ability to react promptly. SegDiff recovers faster and executes correct actions due to its long-horizon planning and its ability to promptly discard the unusable action buffer.

For ***Screw Cap*** (**precise manipulation**), SegDiff is the only method that achieves partial success, indicating its advantage in precise manipulation.

The real-world experiments confirm the effectiveness and manipulation capabilities of SegDiff in the physical world. Additional qualitative visualizations of full task executions are provided in the supplementary material.

Table 2: Success rates on real-world tasks.

Task	DP (w/ TE)	DP (w/o TE)	ACT	SegDiff
Cube	0.7	0.6	0.7	**0.8**
Flip Cup	0.2	0.6	0.3	**0.9**
Package	0.4	0.8	0.8	**1.0**
Banana	0.2	0.4	0.4	**0.9**
Screw Cap	0.0	0.0	0.0	**0.6**

Table 3: Ablation on DTE mechanism.

Task	RAND	INV	DTE
RLBench	**0.880**	0.800	0.860
Cube	**0.8**	0.7	**0.8**
Flip Cup	**0.9**	0.7	**0.9**
Package	0.8	**1.0**	**1.0**
Banana	0.5	0.3	**0.9**
Screw Cap	0.3	0.5	**0.6**

5 Ablation Study and Discussion

5.1 Segmented Trajectory Modeling

A core component of SegDiff is **Segmented Trajectory Modeling**. We use Diffusion Policy (DP) as the baseline, which corresponds to a version without Segmented Trajectory Modeling. For fair comparison, both models employ only the basic temporal ensembling, with the Dynamic Temporal Ensembling mechanism disabled. Fig. 8 shows learning curves on four selected RLBench tasks, demonstrating that Segmented Trajectory Modeling accelerates convergence and achieves higher performance with fewer training steps.

5.2 Dynamic Temporal Ensembling

We conduct ablation experiments on the **Dynamic Temporal Ensembling** (DTE) mechanism by comparing three strategies:

1) **RAND**: Actions are sampled from random Gaussian noise and executed using vanilla temporal ensembling.

2) **INV**: Random sampling is used only when no action buffer is available (segment start); afterward, predictions are initialized from the inverted noise.

3) **DTE**: The complete DTE mechanism is applied.

We use the average success rate on 10 RLBench tasks as the primary metric for simulation. As shown in Tab. 3, the DTE mechanism yields no performance improvement in the simulation environment. This is expected, as simulation environments are generally perturbation-free and exhibit highly monomodal trajectories. In contrast, DTE demonstrates significant advantages in the real-world setting. Specifically, DTE substantially boosts the success rate in the dynamically perturbed **Banana** task. For the **Package** task, which features multi-modal trajectories, DDIM inversion allows both the DTE and INV modes to perform markedly better than RAND. The performance difference between DTE and RAND is marginal for the remaining tasks with relatively fixed trajectory patterns. Crucially, both simulation and real-world results indicate that, in simpler tasks with deterministic trajectories, the INV mode negatively impacts performance. This finding aligns with our theoretical analysis in Sec. 3.3, as the action sampled from inverted noise is potentially influenced by observation from earlier timesteps, consequently reducing the policy's real-time adaptiveness.

5.3 Hyperparameter Sensitivity

We also conduct several additional experiments on RLBench to analyze the sensitivity of key hyperparameters. The results are computed using the average success rate across 10 RLBench tasks.

For the trajectory interpolation length (L) used in Segmented Trajectory Modeling, the left panel of Fig. 9 shows that the success rate peaks at $L = 25$. SegDiff consistently outperforms other methods across different values of L, demonstrating the robustness of our proposed approach.

The update factor (η) is another significant parameter controlling the blending ratio between the action buffer and the new prediction. As shown in the right panel of Fig. 9, when $\eta = 0$, SegDiff executes the entire predicted trajectory from the current state to the keypose at once (*i.e.*, no receding horizon update). Learning this highly constrained, long-horizon trajectory using a standard diffusion model proves challenging, leading to a noticeable drop in success rate. However, because the prediction still focuses on the critical keypose, its performance remains superior to that of the baseline DP. Conversely, $\eta = 1$ corresponds to discarding the action buffer entirely and relying solely on the latest prediction for execution. This also results in degraded performance, as it loses the smoothing and consistency benefits of temporal ensembling. The performance degradation observed at both $\eta = 0$ and $\eta = 1$ further validates the necessity and efficacy of our **Dynamic Temporal Ensembling** mechanism.

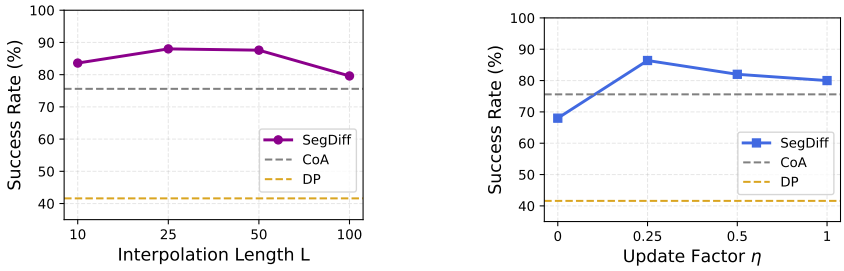

Fig. 9: Results on different interpolation length (L) and update factor (η).

We also evaluate the sensitivity of the Dynamic Temporal Ensembling (DTE) mechanism to the choice of threshold δ used for judging whether two trajectories belong to the same mode. Specifically, we compute δ using three different statistics over the training set: the maximum RMSE (max), the 99% percentile (p99), and the 95% percentile (p95) of the RMSEs of all training samples. The results, reported in Tab. 4, show that DTE is relatively robust to threshold selection.

Table 4: Results of different threshold (δ) for DTE.

δ	max	p99	p95
Success Rate	**0.860**	0.836	0.852

Table 5: Results of DTE under different sampling and inversion steps. SS: Sampling Steps, IS: Inversion Steps, SR: Success Rate, IT: Inference Time.

	SS=50/IS=50	SS=50/IS=10	SS=10/IS=10
SR	**0.860**	**0.860**	0.808
IT(ms)	595.74	380.21	**121.95**

5.4 Efficiency Analysis

As discussed in Sec. 3.3, the number of DDIM inversion steps in DTE is decoupled from the number of sampling steps. Tab. 5 reports the success rate and inference time under different inversion and sampling settings on a single RTX A6000 GPU, indicating that the inversion process is flexible and incurs only minor computational overhead.

Table 6: Efficiency analysis on a real robot system.

Method	Interp. (ms)	Inversion (ms)	Sample (ms)	Control (ms)	Others (ms)	Total (ms)	Mem (GB)
DP	–	–	121.74	407.98	92.45	622.17	1.07
SegDiff	0.52	21.98	125.92	408.83	98.57	655.83	1.07

Furthermore, we evaluate the runtime efficiency of SegDiff on a real robot system, utilizing 10 steps for DDIM inversion and 50 steps for sampling (Tab. 6). Compared to the standard Diffusion Policy (DP), SegDiff increases the total latency by approximately 33.7 ms (∼5.4%). Specifically, the trajectory interpolation and DDIM inversion together introduce an overhead of 22.5 ms per control cycle. The remaining latency arises from the data organization and processing required to sample τ_{rand} and τ_{prior} in parallel. These results demonstrate that SegDiff achieves significantly improved robustness with negligible computational cost, maintaining the real-time responsiveness necessary for high-frequency robot deployment.

6 Conclusion

In this paper, we presented SegDiff, a novel framework for robotic manipulation that bridges continuous action prediction and keypose-level constraints via Segmented Trajectory Modeling. By predicting trajectories anchored to the next keypose, SegDiff effectively guides learning toward task-critical regions and alleviates compounding errors. Furthermore, we introduced the Dynamic Temporal Ensembling mechanism, which leverages DDIM inversion to validate and refine the action buffer in real time, ensuring action mode consistency and enhancing responsiveness to dynamic environments. Extensive experiments across diverse manipulation tasks demonstrate the superior performance of SegDiff in both simulation and real-world scenarios. These results highlight the effectiveness of integrating keypose-aware trajectory modeling with dynamic temporal ensembling for generalizable and reliable robotic policy learning. The primary limitation stems from the dependency on pre-defined keyposes for segmenting demonstrations; incorporating automated or self-supervised keypose discovery would therefore be a promising direction for future research.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (Grant No. 62472098), the Science and Technology Commission of Shanghai Municipality (No. 24511103100) and the New Cornerstone Science Foundation through the XPLORER PRIZE.

References

1. Arachchige, N.R., Chen, Z., Jung, W., Shin, W.C., Bansal, R., Barroso, P., He, Y.H., Lin, Y.C., Joffe, B., Kousik, S., et al.: Sail: Faster-than-demonstration execution of imitation learning policies. In: Conference on Robot Learning. pp. 721–749. PMLR (2025)
2. Bain, M., Sammut, C.: A framework for behavioural cloning. In: Machine intelligence 15. pp. 103–129 (1995)

3. Belkhale, S., Cui, Y., Sadigh, D.: Hydra: Hybrid robot actions for imitation learning. In: Conference on Robot Learning. pp. 2113–2133. PMLR (2023)
4. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al.: \π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164 (2024)
5. Black, K., Galliker, M.Y., Levine, S.: Real-time execution of action chunking flow policies. arXiv preprint arXiv:2506.07339 (2025)
6. Chen, S., Garcia, R., Schmid, C., Laptev, I.: Polarnet: 3d point clouds for language-guided robotic manipulation. In: 7th Conference on Robot Learning (CoRL 2023) (2023)
7. Chen, X., Zhou, Z., Wang, Z., Wang, C., Wu, Y., Ross, K.: Bail: Best-action imitation learning for batch deep reinforcement learning. Advances in Neural Information Processing Systems **33**, 18353–18363 (2020)
8. Chen, Z., Yuan, X., Mu, T., Su, H.: Responsive noise-relaying diffusion policy: Responsive and efficient visuomotor control. arXiv preprint arXiv:2502.12724 (2025)
9. Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. The International Journal of Robotics Research **44**(10-11), 1684–1704 (2025)
10. Duan, Y., Yin, H., Kragic, D.: Real-time iteration scheme for diffusion policy. arXiv preprint arXiv:2508.05396 (2025)
11. Fang, H., Grotz, M., Pumacay, W., Wang, Y.R., Fox, D., Krishna, R., Duan, J.: Sam2act: Integrating visual foundation model with a memory architecture for robotic manipulation. In: International Conference on Machine Learning. pp. 15925–15942. PMLR (2025)
12. Gervet, T., Xian, Z., Gkanatsios, N., Fragkiadaki, K.: Act3d: 3d feature field transformers for multi-task robotic manipulation. In: Conference on Robot Learning. pp. 3949–3965. PMLR (2023)
13. Goyal, A., Blukis, V., Xu, J., Guo, Y., Chao, Y.W., Fox, D.: Rvt2: Learning precise manipulation from few demonstrations. RSS (2024)
14. Goyal, A., Xu, J., Guo, Y., Blukis, V., Chao, Y.W., Fox, D.: Rvt: Robotic view transformer for 3d object manipulation. In: Conference on Robot Learning. pp. 694–710. PMLR (2023)
15. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
16. Høeg, S.H., Du, Y., Egeland, O.: Streaming diffusion policy: Fast policy synthesis with variable noise diffusion models. arXiv preprint arXiv:2406.04806 (2024)
17. Hu, Y., Guo, Y., Wang, P., Chen, X., Wang, Y.J., Zhang, J., Sreenath, K., Lu, C., Chen, J.: Video prediction policy: A generalist robot policy with predictive visual representations. In: International Conference on Machine Learning. pp. 24328–24346. PMLR (2025)
18. James, S., Davison, A.J.: Q-attention: Enabling efficient learning for vision-based robotic manipulation. IEEE Robotics and Automation Letters **7**(2), 1612–1619 (2022)
19. James, S., Ma, Z., Arrojo, D.R., Davison, A.J.: Rlbench: The robot learning benchmark & learning environment. IEEE Robotics and Automation Letters **5**(2), 3019–3026 (2020)
20. James, S., Wada, K., Laidlow, T., Davison, A.J.: Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13739–13748 (2022)

21. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. The international journal of robotics research **30**(7), 846–894 (2011)
22. Karaman, S., Walter, M.R., Perez, A., Frazzoli, E., Teller, S.: Anytime motion planning using the rrt. In: 2011 IEEE international conference on robotics and automation. pp. 1478–1483. ieee (2011)
23. Ke, T.W., Gkanatsios, N., Fragkiadaki, K.: 3d diffuser actor: Policy diffusion with 3d scene representations. In: Conference on Robot Learning. pp. 1949–1974. PMLR (2025)
24. Kim, M.J., Finn, C., Liang, P.: Fine-tuning vision-language-action models: Optimizing speed and success. arXiv preprint arXiv:2502.19645 (2025)
25. Kuffner, J.J., LaValle, S.M.: Rrt-connect: An efficient approach to single-query path planning. In: Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065). vol. 2, pp. 995–1001. IEEE (2000)
26. LaValle, S.: Rapidly-exploring random trees: A new tool for path planning. Research Report 9811 (1998)
27. Liu, M., Zhao, H., Yang, Z., Shen, J., Zhang, W., Zhao, L., Liu, T.Y.: Curriculum offline imitating learning. Advances in Neural Information Processing Systems **34**, 6266–6277 (2021)
28. Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., Zhu, J.: Rdt-1b: a diffusion foundation model for bimanual manipulation. In: International Conference on Learning Representations. vol. 2025, pp. 29982–30009 (2025)
29. Lu, G., Gao, Z., Chen, T., Dai, W., Wang, Z., Ding, W., Tang, Y.: Manicm: Real-time 3d diffusion policy via consistency model for robotic manipulation. arXiv preprint arXiv:2406.01586 (2024)
30. Mandlekar, A., Xu, D., Wong, J., Nasiriany, S., Wang, C., Kulkarni, R., Fei-Fei, L., Savarese, S., Zhu, Y., Martín-Martín, R.: What matters in learning from offline human demonstrations for robot manipulation. In: Conference on Robot Learning. pp. 1678–1690. PMLR (2022)
31. Mayne, D.Q., Michalska, H.: Receding horizon control of nonlinear systems. In: Proceedings of the 27th IEEE Conference on Decision and Control. pp. 464–465. IEEE (1988)
32. Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P., Peters, J., et al.: An algorithmic perspective on imitation learning. Foundations and Trends® in Robotics **7**(1-2), 1–179 (2018)
33. Prasad, A., Lin, K., Wu, J., Zhou, L., Bohg, J.: Consistency policy: Accelerated visuomotor policies via consistency distillation. arXiv preprint arXiv:2405.07503 (2024)
34. Rohmer, E., Singh, S.P., Freese, M.: V-rep: A versatile and scalable robot simulation framework. In: 2013 IEEE/RSJ international conference on intelligent robots and systems. pp. 1321–1326. IEEE (2013)
35. Schaal, S.: Learning from demonstration. Advances in neural information processing systems **9** (1996)
36. Schaal, S.: Is imitation learning the route to humanoid robots? Trends in cognitive sciences **3**(6), 233–242 (1999)
37. Shi, L.X., Sharma, A., Zhao, T.Z., Finn, C.: Waypoint-based imitation learning for robotic manipulation. In: Conference on Robot Learning. pp. 2195–2209. PMLR (2023)
38. Shridhar, M., Manuelli, L., Fox, D.: Perceiver-actor: A multi-task transformer for robotic manipulation. In: Conference on Robot Learning. pp. 785–799. PMLR (2023)

39. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2021)
40. Sundaresan, P., Hu, H., Vuong, Q., Bohg, J., Sadigh, D.: What's the move? hybrid imitation learning via salient points. In: International Conference on Learning Representations. vol. 2025, pp. 51806–51821 (2025)
41. Team, O.M., Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., et al.: Octo: An open-source generalist robot policy. arXiv preprint arXiv:2405.12213 (2024)
42. Wang, D., Liu, C., Chang, F., Xu, Y.: Hierarchical diffusion policy: manipulation trajectory generation via contact guidance. IEEE Transactions on Robotics (2025)
43. Wu, Y., He, K.: Group normalization. In: Proceedings of the European conference on computer vision (ECCV). pp. 3–19 (2018)
44. Xian, Z., Gkanatsios, N., Gervet, T., Ke, T.W., Fragkiadaki, K.: Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In: 7th Annual Conference on Robot Learning (2023)
45. Xie, S., Cao, H., Weng, Z., Xing, Z., Chen, H., Shen, S., Leng, J., Wu, Z., Jiang, Y.G.: Human2robot: Learning robot actions from paired human-robot videos. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 40, pp. 11078–11086 (2026)
46. Yu, D., Xu, H., Chen, Y., Ren, Y., Pan, J.: Bikc: Keypose-conditioned consistency policy for bimanual robotic manipulation. arXiv preprint arXiv:2406.10093 (2024)
47. Zare, M., Kebria, P.M., Khosravi, A., Nahavandi, S.: A survey of imitation learning: Algorithms, recent developments, and challenges. IEEE Transactions on Cybernetics (2024)
48. Ze, Y., Chen, Z., Wang, W., Chen, T., He, X., Yuan, Y., Peng, X.B., Wu, J.: Generalizable humanoid manipulation with 3d diffusion policies. In: 2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 2873–2880. IEEE (2025)
49. Ze, Y., Yan, G., Wu, Y.H., Macaluso, A., Ge, Y., Ye, J., Hansen, N., Li, L.E., Wang, X.: Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In: Conference on robot learning. pp. 284–301. PMLR (2023)
50. Ze, Y., Zhang, G., Zhang, K., Hu, C., Wang, M., Xu, H.: 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. Robotics: Science and Systems XX (2024)
51. Zhang, W., Hu, T., Qiao, Y., Zhang, H., Qin, Y., Li, Y., Liu, J., Kong, T., Liu, L., Ma, X.: Chain-of-action: Trajectory autoregressive modeling for robotic manipulation. arXiv preprint arXiv:2506.09990 (2025)
52. Zhang, W., Li, Y., Qiao, Y., Huang, S., Liu, J., Dayoub, F., Ma, X., Liu, L.: Effective tuning strategies for generalist robot manipulation policies. In: 2025 IEEE International Conference on Robotics and Automation (ICRA). pp. 7255–7262. IEEE (2025)
53. Zhao, T.Z., Kumar, V., Levine, S., Finn, C.: Learning fine-grained bimanual manipulation with low-cost hardware. arXiv preprint arXiv:2304.13705 (2023)