# Unified Condition-Action Modeling for Accurate One-Step Action Generation

**Xinyu Zhou**[1,2], **Zikun Cai**[1,2], **Kuangji Zuo**[1,2], **Gen Li**[1,2], **Boyu Ma**[1,2], **Yanshuo Lu**[1,2], **Yutong Song**[1,2], **Mingqi Yuan**[3], **Jiayu Chen**[3], **Jianfei Yang**[1,2]

[1]MARS Lab, [2]NTU, [3]HKU

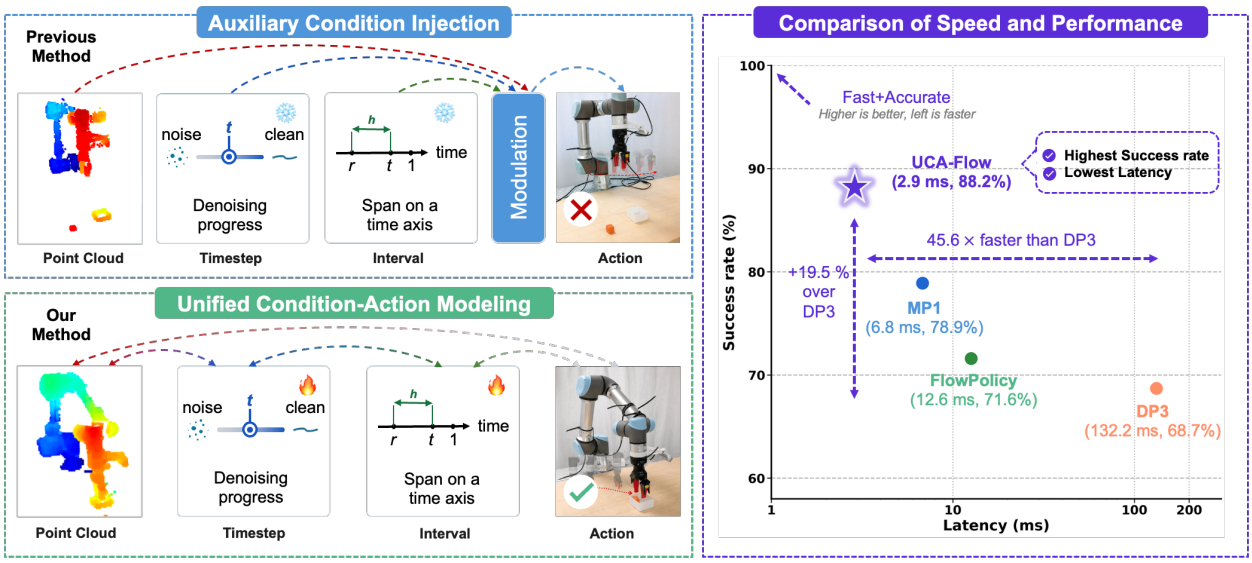
Robot manipulation requires policies that are both accurate and efficient, as robot control must respond to changing observations under tight latency constraints. Recent diffusion and flow policies are promising, but they often treat conditions as auxiliary signals rather than jointly evolving them with action trajectories. We find that this limitation can be effectively mitigated by a **simple yet effective unified condition-action modeling design** that represents conditions and actions in a shared token space, allowing a compact model to achieve high performance while improving both inference speed and accuracy. Therefore, we propose UCA-Flow, a unified condition-action modeling framework for accurate one-step action generation. Our method unifies observation conditions, timestep conditions, interval conditions, and action tokens into a single sequence, and processes them with a Unified Condition-Action Transformer for joint condition-action representation learning. As a result, condition representations are dynamically reconstructed according to the current generation stage, highlighting information most relevant for action refinement. Furthermore, we introduce an improved dual-pass supervision scheme over $u$ and $v$ for stronger optimization of unified condition-action modeling. UCA-Flow improves the average success rate by 9.3 percentage points over the strongest baseline, while achieving 45.6× and 33.4× speedups over DP3 and Simple DP3, and remaining 4.3× and 2.3× faster than one-step FlowPolicy and MP1, respectively. Project page: https://uca-policy.github.io/UCA.github.io/.

**Correspondence:** Jianfei Yang at jianfei.yang@ntu.edu.sg, Jiayu Chen at jiayuc@hku.hk

## 1 Introduction

Robot manipulation aims to generate executable actions from complex observations for physical interaction, such as grasping, pushing, assembly, tool use, and dexterous object manipulation Suomalainen et al. (2022); Bohg et al. (2014); Qin et al. (2023), with broad applications in industrial automation, service robotics, and logisticsSuomalainen et al. (2022); Xu et al. (2024). A key challenge is to extract task-relevant information while generating actions under tight control latency for closed-loop execution. Recent generative robot policies, including Diffusion PolicyChi et al. (2025), Flow MatchingZhang et al. (2025), and Mean FlowSheng et al. (2025), have shown strong potential for modeling action distributions Chi et al. (2025); Ze et al. (2024); Zhang et al. (2025); Sheng et al. (2025); Yan et al. (2025). However, they still face an efficiency-performance bottleneck.

To address this challenge, prior work has improved generative robot policies from several complementary directions. Some methods introduce compact 3D point-cloud features or multimodal state embeddings as conditions for action generation, improving spatial understanding and data efficiency Ze et al. (2024); Zhang et al. (2025); Sheng et al. (2025). Others improve sampling efficiency through flow matching, consistency training, or mean-flow objectives, enabling few-step or one-step action generation Zhang et al. (2025); Geng et al. (2025a); Sheng et al. (2025). Recent transformer-based policies further enhance multimodal conditioning through cross-attention, adaptive cross-attention, AdaLN, or AdaLN-Zero modulation, allowing action tokens to selectively attend to visual, language, timestep, and proprioceptive inputs Reuss et al. (2024); Yan et al. (2025). Despite these advances, as illustrated in the upper part of Figure 1(a), most designs still follow a condition-to-action paradigm: observations and timesteps are encoded as external auxiliary signals and injected into the action-generation backbone through modulation, rather than represented as first-class tokens that dynamically interact with the evolving action. This condition-injection bottleneck is especially restrictive for one-step action generation.




(a) Condition Modeling Paradigms (b) Superiority of UCA-Flow on Adroit and MetaWorld

**Figure 1 Motivation of UCA-Flow**. (a) Existing policies often treat condition as auxiliary condition signals. Unified condition-action modeling enables conditions and actions to evolve together, allowing condition representations to adapt to the current generation stage and action. (b) UCA-Flow achieves a superior efficiency-performance trade-off, delivering the highest success rate with the lowest latency on Adroit and MetaWorld.

In this paper, we propose UCA-Flow, a unified condition-action modeling framework for accurate one-step action generation. As shown in the lower part of Figure 1(a), instead of injecting conditions into an action generator as external guidance, UCA-Flow represents observation conditions, timestep conditions, interval conditions, and action tokens as a unified token sequence. The sequence is processed by a compact Unified Condition-Action Transformer, allowing condition and action representations to be jointly refined within a unified representation space. Thus, condition reconstruction becomes an internal part of generation rather than a pre-computed external input. To better optimize unified condition-action modeling, we further introduce an improved dual-pass supervision scheme over average velocity $u$ and instantaneous velocity $v$. Rather than serving as an isolated MeanFlow objective, this dual supervision regularizes the shared condition-action representation from two complementary velocity views, encouraging more stable and discriminative token representations Geng et al. (2025a,b). As summarized in Figure 1(b), experiments show that UCA-Flow achieves a substantially improved efficiency-performance trade-off. It improves the average success rate by 9.3 percentage points over the strongest baseline, while achieving 45.6× and 33.4× speedups over DP3 and Simple DP3, and remaining 4.3× and 2.3× faster than one-step FlowPolicy and MP1, respectively.

Our contributions are summarized as follows:

- We propose UCA-Flow, a unified condition-action modeling framework for accurate one-step action generation, which represents observation, timestep, interval, and action as first-class tokens in a Unified Condition-Action Transformer, enabling condition representations to be dynamically refined according to the generation stage.
- We introduce an improved dual-pass supervision scheme over average velocity $u$ and instantaneous velocity $v$. This training strategy provides complementary velocity-level constraints to better optimize the unified condition-action modeling framework, improving one-step action accuracy without increasing inference cost.
- Extensive experiments on 37 Adroit and MetaWorld tasks and 2 real-world manipulation tasks demonstrate that UCA-Flow achieves a superior efficiency and performance.

## 2 Related works

*Generative policies for robot manipulation.* Robot manipulation requires policies that infer task-relevant information from complex observations and generate accurate actions for physical interaction Suomalainen et al. (2022); Bohg et al. (2014); Qin et al. (2023); Xu

et al. (2024). Imitation learning provides a practical way to acquire such skills from demonstrations Atkeson and Schaal (1997); Argall et al. (2009); Mandlekar et al. (2021); Florence et al. (2022); Zhao et al. (2023); Haldar et al. (2023). Early behavior cloning and implicit-policy methods learn observation-action mappings, but they struggle with multimodal action distributions and high-dimensional control Florence et al. (2022); Shafiullah et al. (2022); Zeng et al. (2021); Kalashnikov et al. (2018); Pari et al. (2022); Hansen et al. (2023). Recent policies address these limitations by modeling action distributions with expressive models. Diffusion Policy formulates visuomotor control as conditional action denoising and shows strong performance across simulated and real tasks Chi et al. (2025). Subsequent works extend this idea to human-behavior imitation, goal-conditioned diffusion, policy optimization, trajectory planning, grasp-motion optimization and flow-based policy learning Pearce et al. (2023); Reuss et al. (2023); Ren et al. (2024); Janner et al. (2022); Urain et al. (2023); Prasad et al. (2024). However, diffusion-based policies require iterative sampling and increases inference latency.

*3D representations for policy learning.* Because manipulation is inherently spatial, 3D representations are widely used to improve geometric reasoning and data efficiency. Benchmarks such as RLBench provide diverse vision-guided manipulation tasks for evaluating such capabilities James et al. (2020). Recent policies have explored different forms of 3D representations, including voxels, multi-view images, feature fields, and point clouds, to better capture spatial structure for manipulation. Shridhar et al. (2023); Gervet et al. (2023); Goyal et al. (2023, 2024); Ze et al. (2023); Yan et al. (2024); Ke et al. (2025). Point-cloud encoders such as PointNet, PointNet++, and PointNeXt provide compact spatial representations that avoid the high cost of dense voxels Qi et al. (2017a,b); Qian et al. (2022). DP3 shows that sparse point clouds combined with diffusion policies improve data efficiency, and deployment safety Ze et al. (2024). FlowPolicy and MP1 further use compact 3D point-cloud features for efficient flow-based and action generation Zhang et al. (2025); Sheng et al. (2025). Nevertheless, these methods still encode observations as external conditions and inject them into action generators through modulation.

*Efficient one-step action generation.* To reduce sampling cost, recent research has explored more efficient sampling and fast generation, including DDPM, DDIM, score-based modeling, flow matching, consistency models, and MeanFlow Ho et al. (2020); Song et al. (2021a,b); Lipman et al. (2023); Liu et al. (2022); Song et al. (2023); Geng et al. (2025a,b). In robot learning, AdaFlow, FlowPolicy, ManiFlow, and MP1 reduce the number of function evaluations for faster policy inference Hu et al. (2024); Zhang et al. (2025); Yan et al. (2025); Sheng et al. (2025). However, most of them improve the generative objective or sampling process, conditions injected into the action generator as external context. In contrast, UCA-Flow places observation, timestep, interval, and action into a unified condition-action sequence processed by a Unified Condition-Action Transformer, enabling joint condition-action representation learning and dynamic condition reconstruction for accurate one-step action generation.

# 3 Method

We first present the overall formulation of UCA-Flow and its one-step action generation process. We then describe how observation conditions, timestep and interval variables, and noisy action are represented as a unified condition-action token sequence. Based on this sequence, we introduce Unified Condition-Action Interaction for jointly updating condition and action representations. Finally, we detail the dual-pass supervision objective for optimizing interval-aware average velocity prediction.

## 3.1 Overview of UCA-Flow

As shown in Figure 2, UCA-Flow takes an observation history $o$, a noisy action trajectory $z_t$, a timestep $t$, and an interval length $h$ as inputs, and predicts an interval-aware average velocity $u_\theta(z_t, t, h, o)$ for one-step action generation. It unifies observation, timestep, interval, and action tokens into a single sequence, which is processed by a Unified Condition-Action Transformer to jointly update condition and action representations for dynamic condition reconstruction.

Let $x \in \mathbb{R}^{T\times d_a}$ denote an expert action trajectory and $\epsilon \sim \mathcal{N}(0, I)$ denote Gaussian noise. Following mean-flow, we construct $z_t = (1-t)x + t\epsilon,\ t \in [0, 1]$, where larger $t$ indicates a noisier action state, and the instantaneous velocity is $v = \epsilon - x$. UCA-Flow learns $u_\theta(z_t, t, h, o)$ over the interval $h = t - r$. During training, dual-pass supervision optimizes both $u$ and $v$. During inference, starting from $z_1$, it predicts $\hat{x} = z_1 - u_\theta(z_1, 1, 1, o)$ using only the final action-token representations.

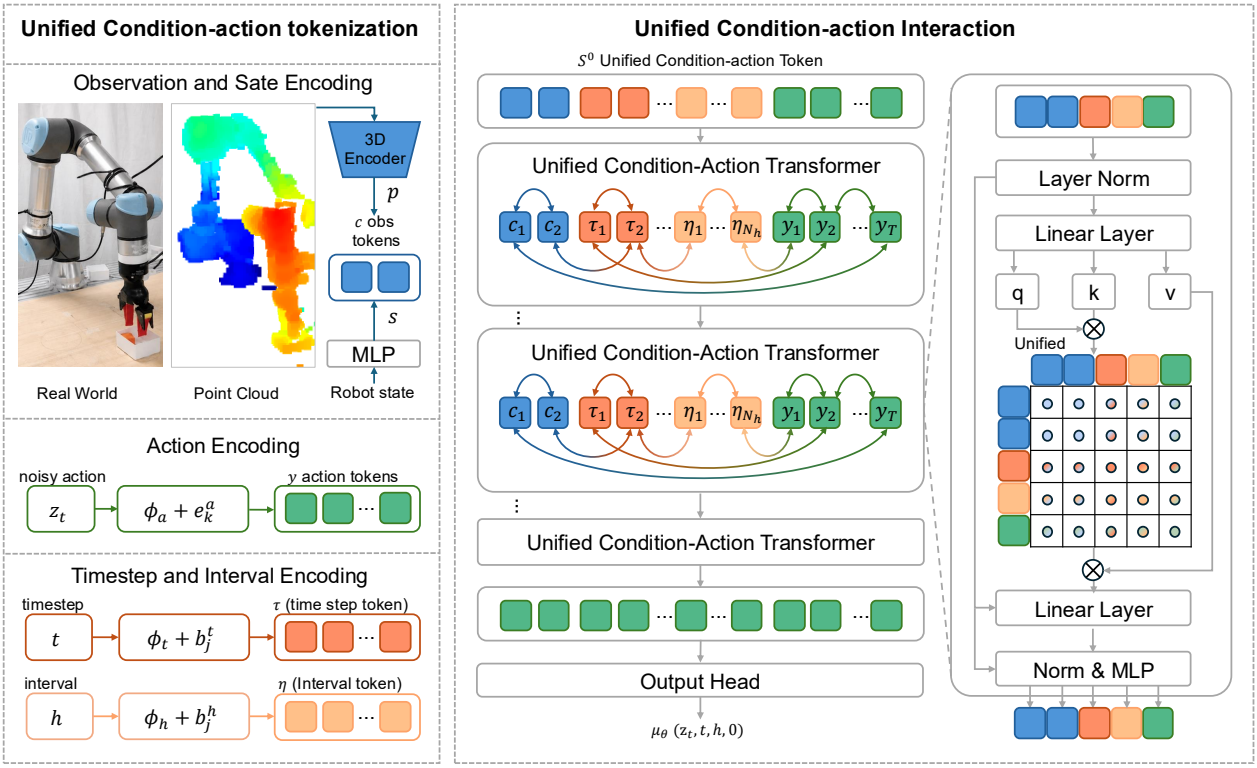



**Figure 2 Overview of UCA-Flow**. Observation, timestep, interval, and noisy action are unified as condition-action tokens and processed by a Unified Condition-Action Transformer to predict the interval-aware velocity.

### 3.2 Unified Condition-Action Tokenization

As shown on the left of the Figure 2, UCA-Flow converts observation tokens, timestep tokens, interval tokens and action tokens into a unified token sequence. The timestep and interval are encoded using sinusoidal embeddings PE followed by independent learnable MLP $\phi$, producing $e_t = \phi_t(\mathrm{PE}(t))$ and $e_h = \phi_h(\mathrm{PE}(h))$. The timestep embedding $e_t$ indicates where the current noisy action lies on the probability path, while the interval embedding $e_h$ specifies the interval over which the average velocity is estimated. To increase representation capacity, we use $N_t$ timestep tokens and $N_h$ interval tokens by adding learnable token-specific biases:

$$\tau_j = e_t + b_j^t, \quad j = 1, \dots, N_t, \tag{1}$$

$$\eta_j = e_h + b_j^h, \quad j = 1, \dots, N_h. \tag{2}$$

For observation tokenization, each observation step is represented by a point-cloud feature $p_i \in \mathbb{R}^{d_p}$ and a robot state feature $s_i \in \mathbb{R}^{d_s}$. We concatenate these two features and project them into the Transformer hidden space as $c_i = \phi_o([p_i; s_i]) + e_i^o$, with $e_i^o$ serving as a learnable observation-position embedding. With two observation steps, the observation token sequence is $O = [c_1, c_2]$. The condition tokens is then constructed as

$$P_{t,h,o} = [c_1, c_2, \tau_1, \dots, \tau_{N_t}, \eta_1, \dots, \eta_{N_h}]. \tag{3}$$

For the noisy action trajectory $z_t$, each horizon step is projected into the same hidden space as $y_k = \phi_a(z_{t,k}) + e_k^a$, $k = 1, \dots, T$, with $e_k^a$ serving as a learnable action-position embedding. Collecting horizon-wise action tokens gives $Y = [y_1, \dots, y_T]$. Finally, the Transformer input is

$$\begin{aligned} S^0 &= [P_{t,h,o}, Y] \\ &= [c_1, c_2, \tau_1, \dots, \tau_{N_t}, \eta_1, \dots, \eta_{N_h}, y_1, \dots, y_T]. \end{aligned} \tag{4}$$

### 3.3 Unified Condition-Action Interaction

As shown on the right of the Figure 2, given the unified input sequence $S^0$, UCA-Flow applies a stack of Unified Condition-Action Transformer blocks to jointly update condition tokens and action tokens. For notational clarity, we define the layer-wise condition representations and action representations as

$$P_{t,h,o}^0 = P_{t,h,o}, \qquad Y^0 = Y. \tag{5}$$

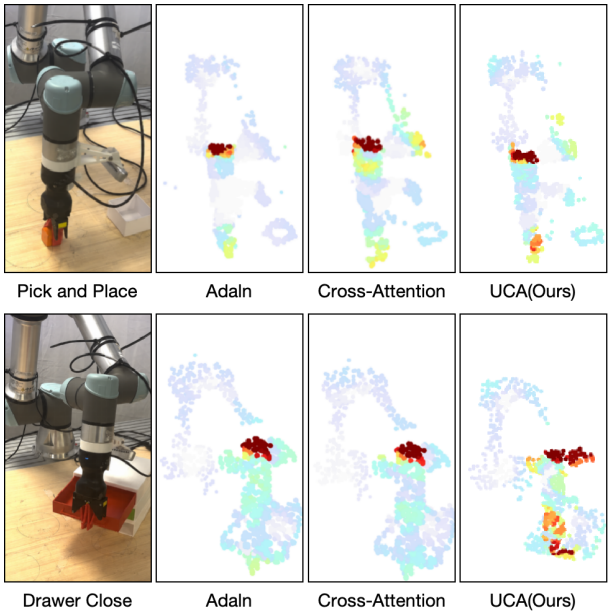



**Figure 3 Gradient-based saliency visualization on real-world tasks**. UCA-Flow highlights more task-relevant regions, including the end-effector, target object, and upper arm joints, compared with the auxiliary condition-injection AdaLN and Cross-Attention.

Then each Transformer block jointly updates both condition tokens and action tokens:

$$[P^{\ell+1}_{t,h,o}, Y^{\ell+1}] = \text{UnifiedConditionActionTransformer}^{\ell}\left([P^{\ell}_{t,h,o}, Y^{\ell}]\right), \quad \ell = 0, \ldots, L-1. \tag{6}$$

Each block consists of multi-head self-attention and a feed-forward network. Unlike designs that update action tokens with externally injected conditions, UCA-Flow processes condition and action tokens within the same Transformer sequence. This allows condition representations to be updated according to the current action state rather than remaining fixed external context, which is also supported by Figure 3, where UCA-Flow shows more task-relevant saliency around the end-effector, target object, and upper arm joints.

This creates an action-condition-action information path across layers. At layer $\ell$, condition tokens absorb information from the current action tokens $Y^{\ell}$, forming action-aware condition representations $P^{\ell+1}_{t,h,o}$. In subsequent layers, action tokens attend to these updated conditions to refine the trajectory. In this way, UCA-Flow turns conditioning from external injection into internal representation evolution, jointly updating condition and action representations during generation.

Finally, we discard the condition outputs and keep only the final action representations:

$$Y^L = [y^L_1, \ldots, y^L_T]. \tag{7}$$

The model prediction is obtained by applying a output head $\psi(\cdot)$ to the action-token sequence:

$$u_\theta(z_t, t, h, o) = \psi(Y^L) \in \mathbb{R}^{T \times d_a}. \tag{8}$$

Here $\psi(\cdot)$ denotes a single output head that maps the final action-token sequence to a velocity trajectory with the same horizon length as the action trajectory. Thus, condition tokens are used as dynamic intermediate representations, while the output is produced only from action tokens.

### 3.4 Dual-Pass Supervision

To optimize UCA-Flow for accurate one-step generation, we use an improved dual-pass supervision scheme over the average velocity $u$ and the instantaneous velocity $v$. Given an expert action trajectory $x$ and Gaussian noise $\epsilon$, we sample $t \geq r$ and define $h = t - r$. We then construct $z_t = (1-t)x + t\epsilon$ and $v = \epsilon - x$. UCA-Flow performs two forward passes during training. The first pass sets $h = 0$ and predicts the instantaneous velocity $v_\theta = u_\theta(z_t, t, 0, o)$, while the second pass uses the sampled interval length $h$ and predicts the interval-aware average velocity $u_\theta = u_\theta(z_t, t, h, o)$. Following the mean-flow identity, we form a composite velocity by estimating the pathwise derivative of $u_\theta$ with a Jacobian-vector product along $(\text{sg}(v_\theta), \mathbf{1}, \mathbf{1})$:

$$D_t u_\theta = \nabla_z u_\theta \cdot \text{sg}(v_\theta) + \partial_t u_\theta + \partial_h u_\theta, \tag{9}$$

$$V_\theta = u_\theta + h\,\text{sg}(D_t u_\theta). \tag{10}$$

Here $\text{sg}(\cdot)$ denotes stop-gradient. We supervise both velocity views with the same target velocity $v$:

$$\mathcal{L} = \rho(V_\theta - \text{sg}(v)) + \rho(v_\theta - \text{sg}(v)), \tag{11}$$

where $\rho(\cdot)$ is the adaptive weighted L2 loss. Given an error tensor $\Delta$, it is defined as

$$\rho(\Delta) = \text{sg}\left(\frac{1}{(\|\Delta\|_2^2 + c)^{1-\gamma}}\right)\|\Delta\|_2^2. \tag{12}$$

Here $\|\cdot\|_2^2$ is computed over the action horizon and action dimensions for each training sample, followed by batch averaging.

The $u$-branch learns interval-aware average velocity for one-step generation, while the $v$-branch provides an auxiliary instantaneous-velocity constraint to stabilize training. During inference, only one forward pass is used: starting from $z_1 = \epsilon$, we set $t = 1$ and $h = 1$, and generate $\hat{x} = z_1 - u_\theta(z_1, 1, 1, o)$.

**Table 1 Ablation study on UCA-Flow components.** -DualPass removes dual-pass supervision; -Unified/c and Unified/a replace unified condition-action modeling with cross-attention and AdaLN, respectively. Results are success rates on representative Adroit and Meta-World tasks.

| Method | Adroit | | MetaWorld | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Door | Pen | Dial Turn | Peg Insert Side | Lever Pull | Pick Place Wall | Stick Pull | Shelf Place |
| **UCA-Flow** | $\mathbf{77.3 \pm 2.0}$ | $\mathbf{63.0 \pm 3.4}$ | $\mathbf{92.3 \pm 4.7}$ | $\mathbf{90.6 \pm 4.0}$ | $\mathbf{89.3 \pm 3.2}$ | $\mathbf{90.6 \pm 3.2}$ | $\mathbf{75.3 \pm 2.3}$ | $\mathbf{74.0 \pm 3.5}$ |
| -DualPass | $74.3 \pm 5.0$ | $57.0 \pm 3.6$ | $91.3 \pm 7.3$ | $88.0 \pm 3.6$ | $89.0 \pm 0.0$ | $89.6 \pm 5.8$ | $70.0 \pm 3.5$ | $64.0 \pm 7.8$ |
| -Unified/c | $68.2 \pm 1.8$ | $58.2 \pm 6.9$ | $87.3 \pm 10.5$ | $86.3 \pm 3.2$ | $79.0 \pm 1.0$ | $84.0 \pm 3.4$ | $62.3 \pm 3.2$ | $56.6 \pm 3.8$ |
| -Unified/a | $56.5 \pm 2.1$ | $48.6 \pm 4.6$ | $78.6 \pm 9.3$ | $61.0 \pm 1.7$ | $63.6 \pm 6.0$ | $60.3 \pm 11.2$ | $68.0 \pm 3.5$ | $32.3 \pm 1.5$ |

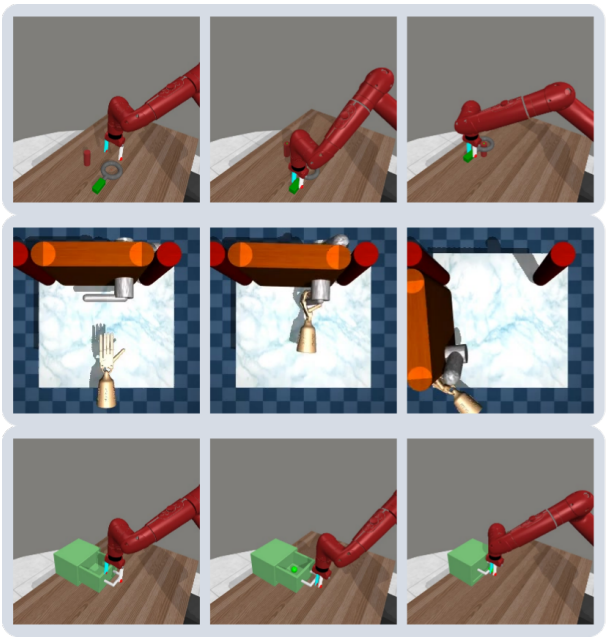


**Figure 4 Representative simulated rollouts.** Rows correspond to Assembly, Adroit Door, and Drawer Close, and columns show successive stages of each execution. The trajectories require different interaction patterns: aligning and inserting an object, coordinating a dexterous hand with a door handle, and maintaining contact while pushing a drawer closed. UCA-Flow completes all three sequences from approach to task completion using one-step action generation.

# 4 Simulation Experiment

## 4.1 Experiment Setup

We evaluate UCA-Flow on 37 simulated manipulation tasks from AdroitRajeswaran et al. (2017) and Meta-WorldYu et al. (2020), following prior 3D policy learning works Ze et al. (2024); Zhang et al. (2025); Sheng et al. (2025). The benchmark includes 3 Adroit dexterous-hand tasks and 34 Meta-World tabletop tasks across different difficulty levels, including 21 Easy, 4 Medium, 4 Hard, and 5 Very Hard tasks. These tasks evaluate action-generation accuracy, task generality, and inference efficiency metrics. We report success rate and inference time as the main performance and efficiency, respectively. All methods use 10 expert demonstrations per task and three seeds (0, 1, 2). Point clouds are downsampled by FPS to 1024 points for Adroit and 512 for Meta-World. Models are trained for 3,000/1,000 epochs on Adroit/Meta-World, evaluated every 200 epochs, and reported using the top five checkpoints per seed. We use AdamW with batch size 128, learning rate $1 \times 10^{-4}$, observation window 2, and prediction horizon 4 on a single NVIDIA RTX A5000 GPU. Inference latency is measured under stable GPU load with three seeds running simultaneously.

## 4.2 Ablation Study on the Core Components of UCA-Flow

Table 1 shows that the full UCA-Flow performs best across most representative tasks. Removing dual-pass supervision causes moderate drops, e.g., Pen decreases from 63.0% to 57.0% and Shelf Place from 74.0% to 64.0%, confirming the benefit of complementary $u$ and $v$ supervision. Replacing unified condition-action modeling with cross-attention injection leads to larger drops, such as Shelf Place from 74.0% to 56.6%. AdaLN-based injection performs worst, with clear failures on Door (77.3% $\rightarrow$ 56.5%), Peg Insert Side (90.6% $\rightarrow$ 61.0%), and Shelf Place (74.0% $\rightarrow$ 32.3%). These results support our motivation that conditions should be jointly modeled with action tokens rather than externally injected as auxiliary signals.

## 4.3 Comparison with State-of-art Methods

*Performance and speed comparison.* Table 2 and Table 3 compare UCA-Flow with representative diffusion-based and flow-based policies. UCA-Flow achieves the best average success rate of 88.2%, improving over MP1 from 78.9% to 88.2% by 9.3 percentage points and outperforming the one-step FlowPolicy by 16.6 percentage points. The gains are especially clear on challenging Meta-World tasks, where UCA-

**Table 2 Success rate comparison on 37 Adroit and Meta-World tasks**. NFE denotes the number of function evaluations required for action generation. All methods are evaluated with three random seeds, and the best results are highlighted in bold.

| Methods | Publication | NFE | Adroit | | | MetaWorld | | | | **Average** |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Hammer | Door | Pen | Easy (21) | Medium (4) | Hard (4) | Very Hard (5) | |
| DP3 | RSS'24 | 10 | $100 \pm 0$ | $56 \pm 5$ | $46 \pm 10$ | $87.3 \pm 2.2$ | $44.5 \pm 8.7$ | $32.7 \pm 7.7$ | $39.4 \pm 9.0$ | $68.7 \pm 4.7$ |
| Simple DP3 | RSS'24 | 10 | $98 \pm 2$ | $40 \pm 17$ | $36 \pm 4$ | $86.8 \pm 2.3$ | $42.0 \pm 6.5$ | $38.7 \pm 7.5$ | $35.0 \pm 11.6$ | $67.4 \pm 5.0$ |
| FlowPolicy | AAAI'25 | 1 | $98 \pm 1$ | $61 \pm 2$ | $54 \pm 4$ | $84.8 \pm 2.2$ | $58.2 \pm 7.9$ | $40.2 \pm 4.5$ | $52.2 \pm 5.0$ | $71.6 \pm 3.5$ |
| MP1 | AAAI'26 | 1 | $100 \pm 0$ | $69 \pm 2$ | $58 \pm 5$ | $88.2 \pm 1.1$ | $68.0 \pm 3.1$ | $58.1 \pm 5.0$ | $67.2 \pm 2.7$ | $78.9 \pm 2.1$ |
| **UCA-Flow** | **Ours** | 1 | $\mathbf{100 \pm 0}$ | $\mathbf{77.3 \pm 2.0}$ | $\mathbf{63.0 \pm 3.4}$ | $\mathbf{92.4 \pm 1.2}$ | $\mathbf{79.3 \pm 3.4}$ | $\mathbf{68.0 \pm 2.0}$ | $\mathbf{85.3 \pm 2.0}$ | $\mathbf{88.2 \pm 1.7}$ |

**Table 3 Inference time comparison on Adroit and MetaWorld**. We report the inference latency in milliseconds with standard deviation. NFE denotes the number of function evaluations required. All methods are evaluated with three random seeds, and the best results are highlighted in bold.

| Methods | Publication | NFE | Adroit /ms | | | MetaWorld /ms | | | | **Average /ms** |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Hammer | Door | Pen | Easy (21) | Medium (4) | Hard (4) | Very Hard (5) | |
| DP3 | RSS'24 | 10 | $129.5 \pm 13.9$ | $141.3 \pm 14.8$ | $145.1 \pm 12.3$ | $129.3 \pm 10.7$ | $134.7 \pm 11.5$ | $131.9 \pm 12.4$ | $138.4 \pm 10.8$ | $132.2 \pm 11.2$ |
| Simple DP3 | RSS'24 | 10 | $103.1 \pm 11.4$ | $111.3 \pm 10.2$ | $128.2 \pm 13.1$ | $91.9 \pm 8.6$ | $98.3 \pm 9.1$ | $101.3 \pm 9.7$ | $103.8 \pm 10.2$ | $97.0 \pm 9.2$ |
| FlowPolicy | AAAI'25 | 1 | $15.3 \pm 1.1$ | $13.2 \pm 4.0$ | $12.0 \pm 2.8$ | $12.0 \pm 1.4$ | $12.2 \pm 1.5$ | $13.5 \pm 1.4$ | $14.5 \pm 1.6$ | $12.6 \pm 1.5$ |
| MP1 | AAAI'26 | 1 | $7.1 \pm 0.2$ | $7.2 \pm 0.1$ | $7.4 \pm 0.3$ | $6.7 \pm 0.0$ | $6.7 \pm 0.1$ | $6.7 \pm 0.1$ | $6.8 \pm 0.1$ | $6.8 \pm 0.1$ |
| **UCA-Flow** | **Ours** | 1 | $\mathbf{2.9 \pm 0.2}$ | $\mathbf{2.9 \pm 0.1}$ | $\mathbf{3.0 \pm 0.3}$ | $\mathbf{3.1 \pm 0.1}$ | $\mathbf{2.9 \pm 0.1}$ | $\mathbf{3.0 \pm 0.1}$ | $\mathbf{2.8 \pm 0.1}$ | $\mathbf{2.9 \pm 0.1}$ |

Flow reaches 79.3%, 68.0%, and 85.3% on Medium, Hard, and Very Hard tasks, respectively. UCA-Flow also achieves the lowest average inference latency of 2.9 ms with only one function evaluation. It is 45.6× faster than DP3, 33.4× faster than Simple DP3, 4.3× faster than FlowPolicy, and 2.3× faster than MP1. These results demonstrate that unified condition-action modeling enables accurate and real-time action generation with a compact model.

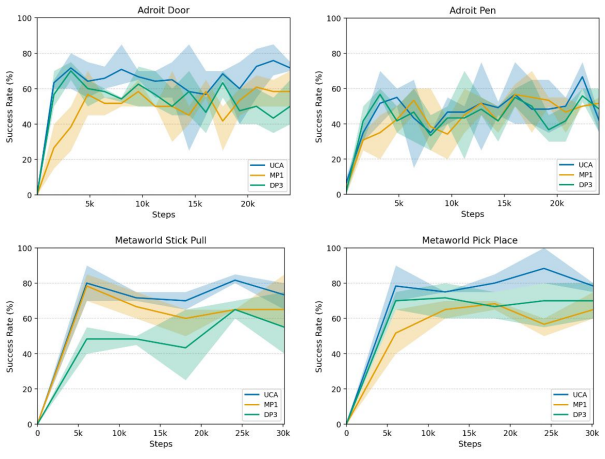



**Figure 5 Training success curves on Adroit and Meta-World.** Solid lines show success rates and shaded regions indicate variation across seeds. Using the top-five-checkpoint mean per seed, UCA-Flow outperforms the baselines on all four tasks, including an approximately 5% gain on Adroit Pen.

# 5 Realworld Experiments

## 5.1 Experiment Setup

We conduct real-world experiments on a UR5e robotic arm with a RealSense L515 RGB-D camera mounted for a top-down workspace view, as shown in Figure 6. We evaluate two tasks: Pick and Place and Drawer Close, involving object placement, drawer closing under different opening degrees. For each task, we collect 40 human demonstrations and use a unified setting with prediction horizon 16 and observation stride 2. Models are trained for 300 epochs per task . We use the AdamW optimizer with a batch size of 128 and learning rate $1 \times 10^{-4}$. Each task is evaluated over 20 independent trials.

## 5.2 Realworld Experimental Results

Figure 6 visualizes the real-world robot tasks, including pick-and-place and drawer-closing. The sequential snapshots show that UCA-Flow can generate executable actions across different manipulation stages, such as approaching the object, completing the grasp, moving to the target region, and closing the drawer. Table 4 compares UCA-Flow with representative baselines on these two real-world tasks, where UCA-Flow improves Pick and Place from 55.0% to 65.0% over MP1 and reaches 100.0% success on Drawer Close. Table 5 further evaluates key design choices in real-world settings. Compared with AdaLN and Cross Attention, UCA-Flow achieves higher success rates on both tasks, indicating that unified condition-action modeling is more effective than auxiliary condition

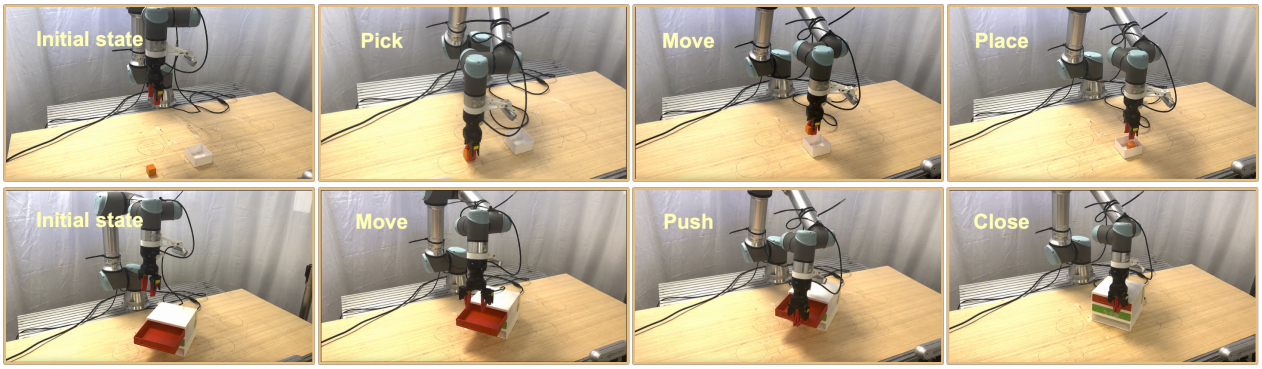



**Figure 6 Real-world execution sequences.** The top row shows Pick and Place progressing from the initial scene through grasping and transport to placement in the target box. The bottom row shows Drawer Close progressing from approach to contact, sustained pushing, and full closure. These sequences illustrate that UCA-Flow maintains task-relevant control through distinct free-space and contact-rich phases despite sensing and execution uncertainty.

injection. These results suggest that UCA-Flow remains effective under real-world sensing noise and execution uncertainty, providing evidence for its real-world manipulation capability.

**Table 4 Real-world comparison with policy baselines.** Success rates (%) are measured over 20 independent trials per task. UCA-Flow improves Pick and Place by 10 percentage points over MP1 and achieves 100% success on Drawer Close.

| **Method** | Pick & Place | Drawer Close |
|---|---|---|
| DP3 | 50.0 | 95.0 |
| MP1 | 55.0 | 95.0 |
| **UCA-Flow** | **65.0** | **100.0** |

**Table 5 Real-world comparison of condition-integration designs.** Success rates (%) are measured over 20 independent trials per task. Unified condition-action modeling performs best on both tasks, exceeding AdaLN by 5 and 10 percentage points on Pick and Place and Drawer Close, respectively.

| **Method** | Pick & Place | Drawer Close |
|---|---|---|
| AdaLN | 60.0 | 90.0 |
| Cross Attention | 50.0 | 95.0 |
| **UCA-Flow** | **65.0** | **100.0** |

## 6 Conclusion

In this paper, we proposed UCA-Flow, a unified condition-action modeling framework for accurate and efficient one-step action generation. UCA-Flow represents observation, timestep, interval, and action tokens in a unified sequence and updates them with a shared Transformer, enabling condition and action representations to be jointly refined during generation. We further introduced dual-pass supervision to improve one-step velocity prediction. Experiments on Adroit and Meta-World show that UCA-Flow outperforms representative diffusion- and flow-based policies while maintaining low inference latency. Real-world experiments further validate its effectiveness on physical robot tasks, demonstrating that unified condition-action modeling is a simple and effective design for fast generative robot policies.

## 7 Limitation

UCA-Flow mainly focuses on accurate and efficient one-step action generation, and our current real-world evaluation is conducted on representative tabletop manipulation tasks. Although these experiments validate the effectiveness of unified condition-action modeling in physical robot settings, future work can further extend UCA-Flow to broader real-world scenarios, such as longer-horizon manipulation and more contact-rich tasks.

## References


Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *International Conference on Machine Learning*, 1997.

Jeannette Bohg, Antonio Morales, Tamim Asfour, and Danica Kragic. Data-driven grasp synthesis—a survey. *IEEE Transactions on Robotics*, 30(2):289–309, 2014.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168, 2022.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025a.

Zhengyang Geng, Yiyang Lu, Zongze Wu, Eli Shechtman, J. Zico Kolter, and Kaiming He. Improved mean flows: On the challenges of fastforward generative models, 2025b. https://arxiv.org/abs/2512.02012.

Theophile Gervet, Zhou Xian, Nikolaos Gkanatsios, and Katerina Fragkiadaki. Act3d: 3d feature field transformers for multi-task robotic manipulation. In *Conference on Robot Learning*, 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, pages 694–710, 2023.

Ankit Goyal, Valts Blukis, Jie Xu, Yijie Guo, Yu-Wei Chao, and Dieter Fox. Rvt-2: Learning precise manipulation from few demonstrations. In *Robotics: Science and Systems*, 2024.

Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. In *Robotics: Science and Systems*, 2023.

Nicklas Hansen, Zhecheng Yuan, Yanjie Ze, Tongzhou Mu, Aravind Rajeswaran, Hao Su, Huazhe Xu, and Xiaolong Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline. In *International Conference on Machine Learning*, pages 12511–12526, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, 2020.

Xixi Hu, Bo Liu, Xingchao Liu, and Qiang Liu. Adaflow: Imitation learning with variance-adaptive flow-based policies. In *Advances in Neural Information Processing Systems*, 2024.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pages 9902–9915, 2022.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on Robot Learning*, pages 651–673, 2018.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. In *Conference on Robot Learning*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. In *Robotics: Science and Systems*, 2022.

Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, and Sam Devlin. Imitating human behaviour with diffusion models. In *International Conference on Learning Representations*, 2023.

Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Robotics: Science and Systems*, 2024.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, 2017b.

Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *Advances in Neural Information Processing Systems*, 2022.

Meiying Qin, Jake Brawer, and Brian Scassellati. Robot tool use: A survey. *Frontiers in Robotics and AI*, 9: 1009488, 2023.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint arXiv:2409.00588*, 2024.

Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, and Rudolf Lioutikov. Multimodal diffusion transformer: Learning versatile behavior from multimodal goals. *arXiv preprint arXiv:2407.05996*, 2024.

Nur Muhammad Mahi Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Advances in Neural Information Processing Systems*, 2022.

Juyi Sheng, Ziyi Wang, Peiming Li, and Mengyuan Liu. Mp1: Meanflow tames policy learning in 1-step for robotic manipulation, 2025. https://arxiv.org/abs/2507.10543.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252, 2023.

Markku Suomalainen, Yiannis Karayiannidis, and Ville Kyrki. A survey of robot manipulation in contact. *Robotics and Autonomous Systems*, 156:104224, 2022.

Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. SE(3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion. In *IEEE International Conference on Robotics and Automation*, pages 5923–5930, 2023.

Zhiyuan Xu, Kun Wu, Junjie Wen, Jinming Li, Ning Liu, Zhengping Che, and Jian Tang. A survey on robotics with foundation models: Toward embodied ai. *arXiv preprint arXiv:2402.02385*, 2024.

Ge Yan, Yueh-Hua Wu, and Xiaolong Wang. Dnact: Diffusion guided multi-task 3d policy learning. *arXiv preprint arXiv:2403.04115*, 2024.

Ge Yan, Jiyue Zhu, Yuquan Deng, Shiqi Yang, Ri-Zhao Qiu, Xuxin Cheng, Marius Memmel, Ranjay Krishna, Ankit Goyal, Xiaolong Wang, et al. Maniflow: A general robot manipulation policy via consistency flow training. *arXiv preprint arXiv:2509.01819*, 2025.

Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In *Conference on Robot Learning*, 2023.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations, 2024. https://arxiv.org/abs/2403.03954.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Ayzaan Wahid, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. In *Conference on Robot Learning*, pages 726–747, 2021.

Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 14754–14762, 2025.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023.