Closed-Loop Action Chunks with Dynamic Corrections for Training-Free Diffusion Policy

Pengyuan Wu[1,2,*], Pingrui Zhang[2,3,*],
Zhigang Wang[2], Dong Wang[2], Bin Zhao[2,4], Xuelong Li[5,✉]

***Abstract*— Diffusion-based policies have achieved remarkable results in robotic manipulation but often struggle to adapt rapidly in dynamic scenarios, leading to delayed responses or task failures. We present DCDP, a Dynamic Closed-Loop Diffusion Policy framework that integrates chunk-based action generation with real-time correction. DCDP integrates a self-supervised dynamic feature encoder, cross-attention fusion, and an asymmetric action encoder–decoder to inject environmental dynamics before action execution, achieving real-time closed-loop action correction and enhancing the system's adaptability in dynamic scenarios. In dynamic PushT simulations, DCDP improves adaptability by 19% without retraining while requiring only 5% additional computation. Its modular design enables plug-and-play integration, achieving both temporal coherence and real-time responsiveness in dynamic robotic scenarios, including real-world manipulation tasks.**

I. Introduction

In recent years, diffusion policies have achieved remarkable progress in robotic manipulation tasks [1], [2], [3]. These methods typically reason over action chunks to capture non-Markovian dependencies, reducing compounding errors in sequential prediction and enabling coherent long-horizon action generation [4], [5], [6].

However, efficient manipulation in dynamic environments requires not only long-horizon planning but also rapid responsiveness to environmental changes.

This dual requirement poses a significant challenge to existing approaches: the policy must generate coherent action sequences over extended time horizons while simultaneously perceiving and adapting to external disturbances or target motions during execution.

To address these challenges, prior work has mainly pursued two directions. The first improves system responsiveness by shortening the prediction horizon or reducing denoising steps [7], [1], [5], but often at the expense of action quality and sequence smoothness, compromising task stability. The second leverages temporal ensembling or bidirectional decoding to integrate multi-step action sequences for closed-loop control [8], [9]. However, these methods are either constrained by reliance on past inference—hindering real-time feedback—or require sampling multiple candidate sequences per step, leading to high computational cost.

To address these limitations, we propose a Dynamic Closed-Loop Diffusion Policy (DCDP) framework (Fig. 1) that integrates long-horizon action chunks planning with real-time correction. It leverages diffusion policies for consistent long-horizon planning while incorporating a lightweight dynamic module that injects high-frequency environmental feedback at each step for closed-loop adaptation.

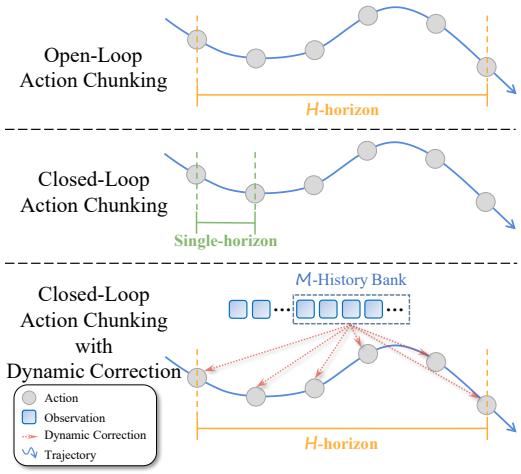

Fig. 1. Comparison among open-loop, closed-loop, and our dynamic correction diffusion policies. Our method leverages a history bank for lightweight real-time correction and efficient closed-loop control.

Specifically, we design a self-supervised dynamic feature encoder that captures environmental variations by learning differential representations from recent observations.Temporal and cross-attention modules enhance temporal modeling. An asymmetric action encoder compresses action sequences into latent representations, which are reconstructed using dynamic features. The combined reconstruction loss and KL regularization encourage the decoder to leverage recent dynamics, improving action adaptability.

Generally, the training and inference stages of the proposed method can be illustrated as follows:

Stage 1 (Training): Based on human demonstration data, the asymmetric action encoder and the self-supervised dynamic feature encoder are trained end-to-end. The differential loss enforces the dynamic feature module to capture temporal dynamics, whereas the reconstruction loss and KL divergence regularize the action decoder to attend to dynamic features.

Stage 2 (Inference): The pretrained policy outputs action chunks for long-term consistency, while the dynamic feature module extracts environmental changes in real time and corrects actions at every step. Updating at the same rate as action execution, this process improves adaptability to

[1]Zhejiang University. [2]Shanghai Artificial Intelligence Laboratory. [3]Fudan University. [4]Northwestern Polytechnical University. [5]Institute of Artificial Intelligence, China Telecom Corp Ltd.
*Equal contribution. ✉Corresponding author.

dynamic environments without sacrificing action coherence.

Notably, our method requires no retraining of the diffusion policy. By inserting the dynamic correction module at inference, it greatly improves responsiveness and robustness in dynamic settings. Its modular design allows seamless integration with other chunk-based policies, enabling a flexible way to balance long-term planning and real-time control.

We evaluated DCDP on a dynamic PushT simulation and two real-world manipulation tasks. Results show that DCDP alleviates the adaptability limitations of diffusion policy in dynamic scenarios without retraining, achieving a 19% higher success rate with only 5% extra computation. Even in static settings, its efficient closed-loop design further improved task performance.

In summary, the contributions of this work are as follows:

- **Closed-Loop Action Correction Framework**: Integrates long-horizon action chunks planning with real-time correction, preserving temporal coherence of actions while enabling fast responses to environmental changes.
- **Dynamic Feature Fusion Module**: Lightweight temporal attention module learns environmental dynamics and fuses them with latent actions, allowing flexible adaptation to perturbations and moving targets.
- **Training-Free Modular Integration**: Improves dynamic adaptability without retraining and supports plug-and-play integration with various chunk-based policies

II. METHOD

A. *Stage 1: Fast Dynamic Aware Policy Training*

This stage trains the Fast Dynamic-Aware Policy (Fig 2, left).

*1) **History Bank Memory Learning**:* We propose a Dynamic Feature Extractor that uses a sliding window of recent observations from the History Bank and applies convolution and attention to capture temporal dependencies and scene dynamics. Let $\mathbf{O}_{t-M+1:t}$ be the M most recent observations. These frames are passed through a pre-trained ResNet18 to extract spatial features, which serve as the input to the temporal modeling module.

*2) **Differential Feature Computation**:* To capture the dynamic changes between consecutive frames, we compute the differential feature $\mathbf{D}_t$ as follows:

$$\mathbf{D}_t = \alpha \cdot (\mathbf{X}_{t+1} - \mathbf{X}_t), \tag{1}$$

where $\mathbf{X}_t$ denotes the extracted feature map at frame t, and α is a learnable scaling parameter. The differential feature $\mathbf{D}_t \in \mathbb{R}^{(M-1)\times C'\times H_f\times W_f}$ represents the temporal differences within the sliding window. This operation enables the model to effectively capture the temporal dynamics between adjacent frames.

*3) **Temporal Attention**:* To model dependencies across time, we apply a temporal attention mechanism over the frame features $\mathbf{X}_{\text{spatial}}$. For each time step t, queries, keys, and values are obtained via linear projections, and attention scores are computed as

$$\text{Attn}_{t,t'} = \text{SoftMax}\left(\frac{\mathbf{Q}_t \mathbf{K}_{t'}^\top}{\sqrt{D}}\right), \tag{2}$$

where D is the feature dimension. The attended feature is the weighted sum of values:

$$\mathbf{X}_{\text{attended},t} = \sum_{t'} \text{Attn}_{t,t'} \mathbf{V}_{t'}. \tag{3}$$

This allows the model to focus on the most relevant frames and capture long-range temporal dependencies, crucial for dynamic scene understanding.

*4) **Fusion Cross-Attention**:* To relate dynamic features to the observation history, we apply cross-attention to fuse the differential feature $\mathbf{D}_t$ with temporal context from the history bank $\mathbf{X}_{\text{temporal}}$. Given queries $\mathbf{Q}_t$ from $\mathbf{X}_{\text{temporal}}$ and keys/values $\mathbf{K}_t, \mathbf{V}_t$ from $\mathbf{D}_t$, the attention output is

$$\text{Attn}(\mathbf{Q}_t, \mathbf{K}_t, \mathbf{V}_t) = \text{softmax}\left(\frac{\mathbf{Q}_t \mathbf{K}_t^\top}{\sqrt{d_k}}\right)\mathbf{V}_t, \tag{4}$$

where d_k is the key dimensionality. This cross-attention aligns temporal context with differential cues, enabling the model to attend to dynamic changes over time. Let $\mathbf{F}_M$ denote the fused representation that summarizes both historical memory and dynamic variation.

*5) **Self-Supervised with Differential**:* In this section, we present a self-supervised learning scheme for the dynamic feature extractor that leverages frame-to-frame differentials to model temporal change, removing the need for manual labels.

During training, at step M the extractor produces predicted dynamic features $\mathbf{F}_M$, while the history bank provides differential targets $\mathbf{D}_{M-1}$ (Sec. II-A.1). The model is conditioned on preceding frames, and $\mathbf{D}_{M-1}$ serves as supervision.

To align predictions with observed changes, we minimize the KL divergence between normalized features:

$$\mathscr{L}_{\text{diff}} = \sum_{t=1}^{T} \text{KL}\Big(\text{softmax}(\mathbf{F}_M^{(t)}) \,\|\, \text{softmax}(\mathbf{D}_{M-1}^{(t)})\Big). \tag{5}$$

This objective encourages the representation to capture temporal dynamics without manual annotations.

*6) **Variational Autoencoder**:* We adopt a lightweight Variational Autoencoder (VAE) to model future actions. The encoder maps the input action sequence $\mathbf{A}_t$ to a Gaussian latent distribution $q(\mathbf{z} \mid \mathbf{A}_t) = \mathscr{N}(\mu, \sigma^2)$, and samples $\mathbf{z}$ using the reparameterization trick $\mathbf{z} = \mu + \sigma\varepsilon,\ \varepsilon \sim \mathscr{N}(0, I)$ for backpropagation. The decoder, conditioned on $\mathbf{z}$ and dynamic features $\mathbf{F}_M$, reconstructs the action sequence using an RNN:

$$p(\mathbf{A}_t \mid \mathbf{z}, \mathbf{F}_M) = \mathscr{N}(\hat{\mathbf{A}}_t, \sigma_{\text{dec}}^2). \tag{6}$$

This enables the policy to learn a compact latent representation and predict actions informed by temporal context.

*7) **Loss**:* The VAE is trained with a weighted sum of reconstruction, KL, and differential losses:

$$\mathscr{L}_{\text{total}} = \mathscr{L}_{\text{recon}} + \lambda_{\text{KL}}\mathscr{L}_{\text{KL}} + \lambda_{\text{diff}}\mathscr{L}_{\text{diff}}, \tag{7}$$

Here λ_{KL} and λ_{diff} balance regularization and differential feature learning.

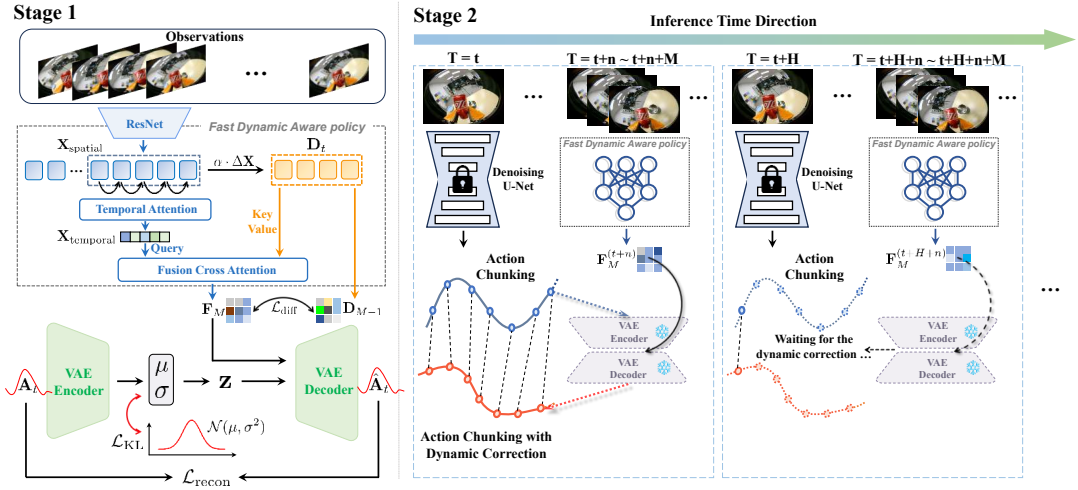

Fig. 2. **Overview of the DCDP.** Our method uses a two-stage framework: (**Stage 1**) train a Fast Dynamic-Aware Policy and a asymmetric Variational Autoencoder (VAE); (**Stage 2**) apply training-free, per-step action corrections using the fast policy and decode corrected actions with the VAE.

B. Stage 2: Dynamic Injection for Training-Free Diffusion Policy

During dynamic evaluation (Fig 2, right), we maintain a sliding window of length M over the observations, $\mathbf{O}_{t-M+1:t} = [\mathbf{o}_{t-M+1}, \ldots, \mathbf{o}_t]$. We then compute dynamics-aware features online using the Stage-1 Fast Dynamic-Aware extractor π_f:

$$\mathbf{F}_t = \pi_f(\mathbf{O}_{t-M+1:t}) \in \mathbb{R}^{d_f}. \tag{8}$$

A frozen slow diffusion policy produces a horizon-H open-loop action chunk conditioned on the current observation,

$$\mathbf{A}_{t:t+H-1} \sim \pi_s(\cdot \mid \mathbf{o}_t), \quad \mathbf{A}_{t:t+H-1} = [\mathbf{a}_t, \ldots, \mathbf{a}_{t+H-1}]. \tag{9}$$

For feature injection, we first apply elementwise normalization using known bounds $\mathbf{a}_{\min}, \mathbf{a}_{\max}$:

$$\tilde{\mathbf{A}}_{t:t+H-1} = 2\,\frac{\mathbf{A}_{t:t+H-1} - \mathbf{a}_{\min}}{\mathbf{a}_{\max} - \mathbf{a}_{\min}} - \mathbf{1}, \tag{10}$$

where all operations are elementwise and $\mathbf{1}$ denotes an all-ones tensor matching the shape of $\mathbf{A}_{t:t+H-1}$. We then encode this chunk into a latent vector with the frozen VAE encoder E trained in Stage 1:

$$\mathbf{z}_t = E\big(\tilde{\mathbf{A}}_{t:t+H-1}\big) \in \mathbb{R}^{d_z}. \tag{11}$$

Within each chunk, execution proceeds in a closed loop. For each step $s \in \{0, \ldots, H-1\}$, we refresh the history window and recompute features:

$$\mathbf{F}_{t+s} = \pi_f(\mathbf{O}_{t+s-M+1:t+s}). \tag{12}$$

We then decode a dynamically corrected action with the frozen VAE decoder D trained in Stage 1, conditioned on a step embedding $\mathbf{e}_s$:

$$\hat{\mathbf{a}}_{t+s} = D(\mathbf{z}_t, \mathbf{F}_{t+s}, \mathbf{e}_s). \tag{13}$$

Aggregating over steps yields the closed-loop chunk:

$$\hat{\mathbf{A}}'_{t:t+H-1} = [D(\mathbf{z}_t, \mathbf{F}_t, \mathbf{e}_0),\ \ldots,\ D(\mathbf{z}_t, \mathbf{F}_{t+H-1}, \mathbf{e}_{H-1})]. \tag{14}$$

Equivalently, this induces a deterministic joint closed-loop policy

$$\pi_c(\hat{\mathbf{a}}_{t+s} \mid \mathbf{o}_t, \mathbf{O}_{t+s-M+1:t+s}) = \delta(\hat{\mathbf{a}}_{t+s} - D(E(\mathrm{norm}(\pi_s(\mathbf{o}_t))), \pi_f(\mathbf{O}_{t+s-M+1:t+s}), \mathbf{e}_s)) \tag{15}$$

where $\mathrm{norm}(\cdot)$ denotes the linear normalization above and $\delta(\cdot)$ is a Dirac delta indicating a deterministic mapping. After $s = H-1$, we replan by resampling a new chunk from π_s with the latest observation (receding-horizon), while keeping all modules frozen; thus, the entire Stage-2 procedure is *training-free*. The continual injection of $\mathbf{F}_{t+s}$ provides fine-grained online corrections to the open-loop diffusion chunk, improving responsiveness and robustness in rapidly changing scenes.

III. Experiment

A. Experimental Settings

For the PushT task, we employ a diffusion policy trained from human demonstrations as the basic policy.

During inference, we set the batch size to $N = 50$, meaning that 50 initial poses are randomly generated in the simulation environment and kept fixed throughout the experiments.

For each initial condition, we perform rollouts and evaluate task success rate and inference latency. Each episode ends when $T_{\max} = 300$ steps are reached or the object–target overlap σ exceeds 95%. The environment runs at 10 Hz. The model predicts 16 actions per inference and executes the first 8 in open-loop mode.

Baselines: We compare our method with three representative inference baselines: Open-loop(H=8): Execute an entire action chunk at each inference step; Closed-loop(H=1): Execute only the most recent action at each inference step; Temporal Ensemble [10](H=1): At each overlapping step, we average the new prediction a with the previous prediction $\hat{a}$ to produce smoother action chunks: $a_t = \lambda a_t + (1-\lambda)\hat{a}_t$, with λ set to 0.5.

H denotes that the diffusion policy model is inferred every H time steps.

Perturbations: To evaluate robustness and generalization under dynamic scenarios, we introduce two perturbations in simulation: (1) Constant-direction: a fixed-magnitude offset with constant direction is applied at each step to simulate sustained external forces; (2) Random-direction: a fixed-magnitude offset with randomly sampled direction is applied at each step, resampled every $N = 50$ steps.

Inference Latency Measurement: To ensure a fair evaluation of inference efficiency, we measure all inference latencies on a single NVIDIA RTX 4090 GPU 24GB under identical hardware and software configurations.

B. Quantitative Analysis

Table I presents a comparison of task success rates between the proposed method and several baselines in both static scenarios and dynamic scenarios under varying perturbations. Table II shows the average single-step latency for each method, highlighting the real-time and computational improvements of our method.

TABLE I
SUCCESS RATES

Methods	Static	Constant perturbations	Random perturbations
Open-Loop	88.4	58.2	52.8
Closed-Loop	84.6	76.1	61.6
Temporal Ensemble	81.0	65.8	57.3
DCDP	**92.5**	**77.6**	**71.9**

TABLE II
COMPARISON OF PER-STEP INFERENCE LATENCY.

Methods	OL(H=8)	CL(H=1)	TE(H=1)	**DCDP(H=8)**
Delay (ms)	7.05	53.60	53.74	**7.39**

The original closed-loop method improves task success in dynamic scenarios but degrades performance in static settings. This is likely due to full-sequence replanning at each inference step, which disrupts continuity between consecutive actions and impairs long-horizon coordination learned from human demonstrations. In contrast, DCDP reduces replanning frequency, combining long-horizon planning with recent observations for rapid closed-loop control, resulting in significant gains in both static and dynamic tasks.

Although DCDP introduces a lightweight feature extractor and an asymmetric action VAE for dynamic correction, it adds only 5% computational overhead (Table II) while keeping inference latency below 10 ms per step on an RTX 4090.

C. Ablation Study

To evaluate the necessity and effectiveness of each module in the proposed architecture, we conduct ablation studies to analyze their relative contributions to model performance.

Specifically, we conducted ablation studies on the self-supervised dynamic feature extraction module. Key components were sequentially removed, and Stage 1 was retrained after each removal.

TABLE III
TASK SUCCESS RATES AFTER ABLATING INDIVIDUAL MODULES.

TA	SSD	DCA	SR_{static}	$SR_{dynamic}$
			88.40	58.20
	✓	✓	92.05	73.59
✓		✓	92.31	73.50
✓	✓		93.36	68.23
✓	✓	✓	**92.50**	**77.60**

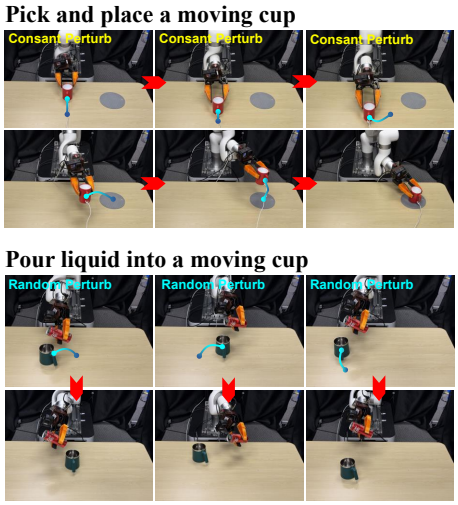

Fig. 3. The two tasks involve two types of perturbations: constant-direction and random-direction. These perturbations were applied exclusively to the components highlighted in the figure.

Table III presents the results of the ablation experiments.**TA** refers to temporal attention, **SSD** refers to self-supervised diff loss, **DCA** refers to diff cross attention, and $\mathbf{SR_{static}}$, $\mathbf{SR_{dynamic}}$, denotes success rate (%). We selected static scenarios and constant-direction perturbations as evaluation metrics.

The results demonstrate that all proposed modules make positive contributions to system performance.

D. Real-World Application

To validate the proposed algorithm in real-world scenarios, we evaluate it on two representative manipulation tasks: *pick-and-place of a moving cup* and *pouring liquid into a moving cup*, as shown in Fig. 3.

We employed the UMI [11] gripper and utilized the publicly available FastUMI [12] dataset, reproducing its scenarios to construct a real-world evaluation platform.

In real-world evaluations, we adopted a similar task setup with constant- and random-direction perturbations. Constant perturbations were applied in the *pick-and-place of a moving cup* task, where the cup moved along a fixed direction. Random perturbations were applied in the *pouring liquid into a moving cup* task, requiring the robot to accurately pour into a cup moving in unpredictable directions.

The results indicate that, compared with the original method, DCDP exhibits significantly enhanced adaptability in dynamic scenarios.

ACKNOWLEDGMENT

This work is supported by the Shanghai AI Laboratory.

REFERENCES

[1] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion," in *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023.

[2] M. Reuss, M. Li, X. Jia, and R. Lioutikov, "Goal-conditioned imitation learning using score-based diffusion policies," *arXiv preprint arXiv:2304.02532*, 2023.

[3] Y. Liu, W. C. Shin, Y. Han, Z. Chen, H. Ravichandar, and D. Xu, "Immimic: Cross-domain imitation from human videos via mapping and interpolation," *arXiv preprint arXiv:2509.10952*, 2025.

[4] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," *arXiv preprint arXiv:2403.03954*, 2024.

[5] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine, "Planning with diffusion for flexible behavior synthesis," *arXiv preprint arXiv:2205.09991*, 2022.

[6] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, *et al.*, "Imitating human behaviour with diffusion models," *arXiv preprint arXiv:2301.10677*, 2023.

[7] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, "Consistency policy: Accelerated visuomotor policies via consistency distillation," *arXiv preprint arXiv:2405.07503*, 2024.

[8] Y. Liu, J. I. Hamid, A. Xie, Y. Lee, M. Du, and C. Finn, "Bidirectional decoding: Improving action chunking via closed-loop resampling," *International Conference on Learning Representations (ICLR)*, 2025.

[9] A. George and A. B. Farimani, "One act play: Single demonstration behavior cloning with action chunking transformers," *arXiv preprint arXiv:2309.10175*, 2023.

[10] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[11] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song, "Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots," *arXiv preprint arXiv:2402.10329*, 2024.

[12] K. Liu, C. Guan, Z. Jia, Z. Wu, X. Liu, T. Wang, S. Liang, P. Chen, P. Zhang, H. Song, *et al.*, "Fastumi: A scalable and hardware-independent universal manipulation interface with dataset," *arXiv preprint arXiv:2409.19499*, 2024.