DSDyn-VLA: A Dual-Stream Dynamic Manipulation Framework with Motion Perception, Future Awareness, and Realtime Correction

Anonymous authors
Paper under double-blind review

Abstract

While Vision-Language-Action (VLA) models excel in static tasks, they struggle in dynamic environments where objects are in motion (e.g., conveyor belt manipulation). We identify three fundamental limitations hindering current VLAs in these scenarios: the **perception gap**, where static visual inputs lack temporal motion cues; the **latency gap**, where inference delays render actions obsolete; and the **control gap**, caused by the open-loop action chunk execution without real-time adjustment. In this work, we propose **DSDyn-VLA**, a Slow-Fast **D**ual-**S**tream **Dyn**amic manipulation framework that integrates motion-aware foresighted planning with real-time residual correction. The slow **Flow-Planner** serves as a macro-planner. By enhancing the VLA with optical flow for temporal perception and a future state awareness mechanism to preemptively offset inference latency, it produces globally consistent, motion-aware action chunks. Complementing this, the fast **Res-Refiner** employs a lightweight RL policy to inject high-frequency, closed-loop corrections into the planned action chunks based on real-time observations. In addition, we introduce **DynBench**, a MuJoCo-based benchmark for dynamic object manipulation that comprises nine tasks. Extensive experiments demonstrate that DSDyn-VLA reduces the failure rate by over 76% compared to current SOTA method in high-latency setting on the Kinetix dynamic benchmark, while achieving about 6$\times$ the success rate of PI0.5 in real-world dynamic settings and about 5$\times$ on DynBench. We will open-source all the code and weights.

1 Introduction

In recent years, the field of robotic manipulation Han et al. (2023); Wolf et al. (2025); Xia et al. (2022); Khazatsky et al. (2024); Shridhar et al. (2022) has witnessed a paradigm shift with the emergence of Vision-Language-Action (VLA) models Kim et al. (2024); Shukor et al. (2025); Wen et al. (2025); Liu et al. (2024a); Team et al. (2024); Li et al. (2024). By training on internet-scale multimodal data Bu et al. (2025); O'Neill et al. (2024), these models have demonstrated remarkable capabilities in semantic reasoning and zero-shot generalization, enabling robots to interpret complex natural language instructions and perform diverse manipulation tasks Intelligence et al. (2025a); Black et al. (2024); Intelligence et al. (2025b). Current VLAs have achieved high success rates across a broad spectrum of tasks, ranging from simple pick-and-place operations to complex activities such as folding clothes, assembling boxes, and making coffee Huang et al. (2023); Mees et al. (2022); Liu et al. (2023).

However, regardless of their complexity, all of these tasks are static, where target objects do not move spontaneously. Yet, the physical world is inherently dynamic. Many practical applications—ranging from industrial sorting on moving conveyor belts Cao et al. (2025) to household human-robot handover Huang et al. (2015) or intercepting moving objects Sun et al. (2023)—require the robot not only to understand "what to do" but also to react with precise timing and agility. In these dynamic scenarios, the capabilities of current VLA models break down. We argue that the inability of existing VLAs to handle dynamic manipulation stems from three fundamental mismatches, which we categorize as the *1) Perception Gap*, the *2) Latency Gap*, and the *3) Control Gap*.

The *Perception Gap* arises because standard VLAs typically rely on static RGB snapshots as visual input Zhong et al. (2025), which preserve semantics but discard temporal motion cues. Without explicit velocity-related information, a VLA cannot estimate the target object's trajectory or time-to-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

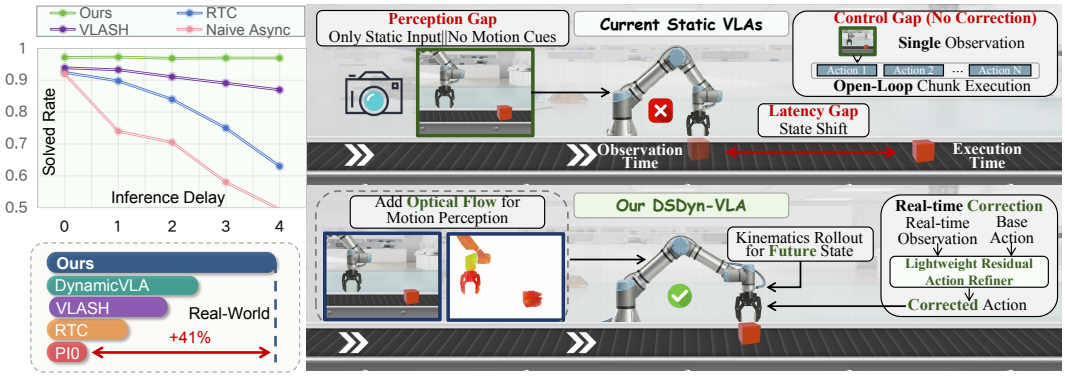

Figure 1: **Left:** Comparison on Kinetix benchmark and real-world tasks. **Top Right:** Current VLAs fail to perform dynamic tasks due to three gaps: the perception gap, latency gap, and control gap. **Bottom Right:** Our DSDyn-VLA addresses these gaps by introducing optical flow for motion perception, future state awareness for latency compensation, and a refiner for real-time correction.

contact, which is a prerequisite for dynamic interaction. The *Latency Gap* comes from the inference delay between observation and execution Black et al. (2025a); Tang et al. (2025). While this delay is negligible in static settings, the world state evolves significantly during inference in dynamic environments. Consequently, actions based on obsolete observations inevitably lead to execution failures. The *Control Gap* is caused by "action chunking" Sendai et al. (2025). To ensure temporal consistency, VLAs predict a multi-step action chunk at once. This chunk is executed in an open-loop manner. Once chunk execution begins, the robot is "blind" to new environmental changes until the next inference cycle. This lack of closed-loop correction makes it impossible to adjust to perturbations or unexpected object movements within a chunk's duration.

To bridge these gaps, we propose **DSDyn-VLA**, a novel Slow-Fast Dual-Stream Framework inspired by the complementary roles of the cerebrum and cerebellum in motor control. The "Slow" stream, designated as the **Flow-Planner**, serves as the semantic macro-planner. It is an enhanced VLA model designed to overcome perception and latency gaps. To address the Perception Gap, the Flow-Planner augments the visual input with optical flow, explicitly providing the model with temporal dynamics and motion cues. For the Latency Gap, we introduce a Future State Awareness mechanism. Instead of planning for the *current* state, the Flow-Planner predicts the future state at execution time and generate actions for that future moment, thereby preemptively offsetting system latency.

Complementing this, the "Fast" stream, termed the **Res-Refiner**, acts as a reactive motion refiner. Res-Refiner consists of a lightweight, RL-trained residual network. It operates in real-time, running in parallel with the execution of the Flow-Planner's action chunks. By taking real-time observations, the Res-Refiner injects residual corrections into the base action each timestep. This effectively closes the control loop, allowing the system to compensate for prediction errors and rapid environmental changes that the Flow-Planner cannot react to in time, thus resolving the Control Gap. Extensive experiments demonstrate that DSDyn-VLA reduces the failure rate of SOTA methods from 13% to 3% on the Kinetix dynamic benchmark, while improving the real-world success rate from 8.4% with PI0.5 to 49.0%. In addition, we construct **DynBench**, a MuJoCo-based dynamic manipulation simulation benchmark, comprising 9 tasks, on which DSDyn-VLA achieves an average success rate of 48.4%, substantially outperforming PI0.5 at 9.7%. Our main contributions are as follows:

- We propose **DSDyn-VLA**, a novel Slow-Fast Dual-Stream framework tailored for dynamic manipulation. It synergizes macro-level chunk prediction with stepwise residual correction to resolve the real-time responsiveness limitations of traditional VLAs.
- We design the **Flow-Planner**, an enhanced VLA slow stream that integrates optical flow input with a future state awareness mechanism, endowing the VLA with the capability to perceive object motions and proactively offset inference latency.
- We introduce the **Res-Refiner**, a real-time RL-based residual policy that performs high-frequency closed-loop adjustments during action chunk execution, effectively compensating for environmental perturbations to ensure robust performance in highly dynamic settings.
- We construct **DynBench**, which establish a new testbed for systematically evaluating dynamic manipulation policies.

2 Related Work

Real-time VLA Models for Latency Gap. To mitigate the "stop-and-go" behavior caused by inference latency, recent works have adopted asynchronous inference Zhao et al. (2025); Jiang et al. (2025). However, this introduces temporal misalignment, as observations become stale during generation. Real-Time Chunking (RTC) Black et al. (2025a) addresses this via inference-time inpainting, while Training-Time RTC Black et al. (2025b) optimizes this by conditioning on action prefixes during training to eliminate runtime overhead. Similarly, VLASH Tang et al. (2025) bridges the gap by rolling forward the robot's proprioceptive state to condition generation on estimated execution-time states, which inspires the state prediction mechanism in our Flow-Planner. DynamicVLA Xie et al. (2026) mitigates the latency gap through latency-aware action streaming, which discards stale actions. However, these methods still rely on static visual inputs and open-loop chunk execution. Consequently, they fail to address the *Perception Gap* and *Control Gap*, causing them to remain ineffective for highly dynamic tasks.

3 Method

3.1 Preliminaries

Optical Flow for Dynamics Perception. Standard VLA models operate on static RGB snapshots $\mathbf{I}_{rgb} \in \mathbb{R}^{H\times W\times 3}$. However, this representation lacks the high-order derivative information—specifically velocity—which is crucial for moving targets. Optical flow field $\mathbf{F}_t \in \mathbb{R}^{H\times W\times 2}$ is a dense vector map where each pixel (x, y) is assigned a displacement vector $\mathbf{v}_{x,y} = (u, v)$, representing the motion state between consecutive frames $\mathbf{I}_{t-1}$ and $\mathbf{I}_t$. Unlike raw pixel differences, $\mathbf{F}_t$ explicitly encodes the scene's motion field. By conditioning the policy on $\mathbf{F}_t$, we effectively provide the model with a visual approximation of the system's state derivative.

Action Chunking Policy. To ensure temporal coherence, most existing VLA models adopt an Action Chunking mechanism. At time step t, the policy π maps the current observation $\mathbf{o}_t$ to a sequence of H future actions:

$$\mathbf{A}_t = \pi(\mathbf{o}_t) = \{\mathbf{a}_k\}_{k=0}^{H-1} \tag{1}$$

The robot executes a subset of these actions (execution horizon $K \leq H$) before the next inference cycle completes. This chunk-based generation is essential for high-frequency control but inherently transitions the system from fully closed-loop feedback to a partially open-loop execution. During the horizon K, the robot executes the pre-planned sequence blindly, rendering it unresponsive to real-time environmental perturbations.

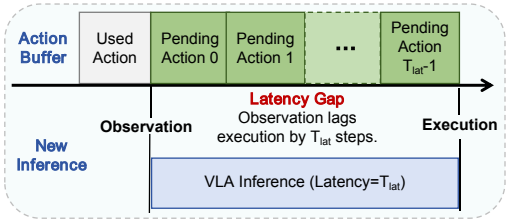

Figure 2: Illustration of VLA asynchronous inference. VLA initiates a new inference when T_{lat} (representing the VLA inference latency steps) actions remain in the action buffer, ensuring the new action chunk is available when the buffer is exhausted. The observation for inference lags the execution by T_{lat} steps.

Asynchronous Inference and Latency Gap. Deploying VLA models for real-time control typically adopts an Asynchronous Inference paradigm. As illustrated in Figure 2, the inference process is decoupled from robot execution. To ensure continuous motion, the VLA initiates a new prediction while the previous action chunk is still being executed, ensuring that the fresh action chunk arrives before the current buffer is exhausted. Let T_{lat} denote the inference latency in steps; the VLA triggers inference T_{lat} steps in advance. This delay creates a fundamental *State Shift*, which we term the *Latency Gap*. The policy generates actions conditioned on the past state $\mathbf{s}_t$, but these actions are applied to the future state $\mathbf{s}_{t+T_{lat}}$. In dynamic settings where the environment evolves significantly during T_{lat}, this temporal misalignment leads to catastrophic errors.

3.2 DSDyn-VLA Overview

We propose **DSDyn-VLA**, a bi-level framework inspired by the biological interplay between the cerebrum and cerebellum. The system decomposes the control problem into two parallel streams operating at distinct frequencies:

- **The Slow Stream (Flow-Planner):** Operates at a lower frequency ($f_{slow} \approx 2-3$Hz). It utilizes a high-capacity VLA backbone to perform semantic reasoning and long-horizon

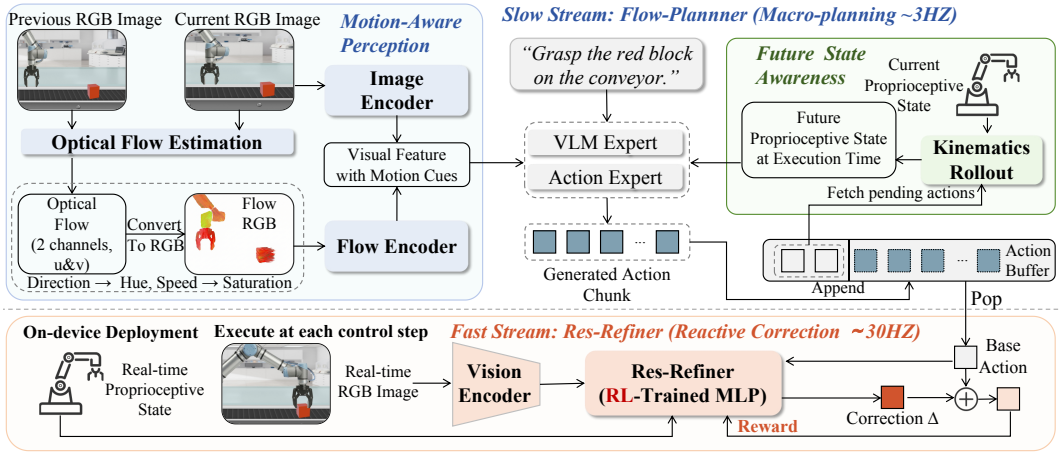

Figure 3: Architecture of our DSDyn-VLA, which consists of Flow-Planner and Res-Refiner.

planning. It addresses the *Perception Gap* via optical flow integration and the *Latency Gap* via a future-state awareness mechanism.

- **The Fast Stream (Res-Refiner):** Operates at a high frequency ($f_{fast} \approx 30$Hz). It utilizes a lightweight, edge-deployed residual policy to perform real-time closed-loop corrections. It addresses the *Control Gap* by reacting to high-frequency disturbances that occur within the duration of a macro-action chunk.

Formally, the final control action $\mathbf{a}_t^{final}$ executed by the robot is the superposition of the open-loop macro-plan and the closed-loop residual:

$$\mathbf{a}_t^{final} = \underbrace{\mathbf{a}_t^{slow}}_{\text{Macro-Plan}} + \underbrace{\pi_{fast}(\mathbf{s}_t, \mathbf{a}_t^{slow})}_{\text{Reflex-Correction}} \tag{2}$$

where $\mathbf{a}_t^{slow}$ is the specific step corresponding to time t extracted from the active chunk $\mathbf{A}_{slow}$.

3.3 The Slow Stream: Flow-Planner

The Flow-Planner serves as the semantic backbone of our system. It is built upon the PI0.5 architecture Intelligence et al. (2025b), a Conditional Flow Matching (CFM) policy. We introduce substantial architectural enhancements to adapt it for dynamic interaction.

3.3.1 Motion-Aware Perception

To inject explicit temporal cues into the VLA latent space, we design a dual-branch visual system.

Flow Extraction & Visualization: First, we extract the dense optical flow $\mathbf{F}_t$ using RAFT-Large Teed & Deng (2020), a recurrent all-pairs field transform network. To align this geometric data with the pre-trained semantic space of CLIP, we transform the 2-channel flow field into a 3-channel RGB-like representation $\mathbf{I}_{flow}$ via a differentiable polar mapping. For a flow vector (u, v):

$$\text{Hue} = \text{atan2}(v, u), \quad \text{Saturation} = \sqrt{u^2 + v^2} \tag{3}$$

This mapping visualizes motion direction as color and speed as intensity.

Dual-Encoder Architecture: We employ two parallel Vision Transformers as encoders. The first encoder E_{rgb} processes the static image $\mathbf{I}_{rgb}$ to extract semantic features (e.g., object identity). The second encoder E_{flow} processes the flow image $\mathbf{I}_{flow}$ to extract dynamic features (e.g., motion trajectory). We fine-tune both encoders end-to-end. The output tokens are concatenated along the sequence dimension to form the unified visual context:

$$\mathbf{z}_{vis} = \text{Concat}\left(E_{rgb}(\mathbf{I}_{rgb}), E_{flow}(\mathbf{I}_{flow})\right) \in \mathbb{R}^{(2N)\times D} \tag{4}$$

where N is the number of patches and D is the embedding dimension. This ensures the planner attends to both "what" the object is and "how" it is moving.

3.3.2 Future State Awareness via Kinematics Rollout

To neutralize the *Latency Gap* defined in Sec. 3.1, the Flow-Planner must condition its generation on the robot's state at the *execution time*, rather than the *observation time*. Since the visual future is unknown, we exploit the deterministic nature of the robot's internal state.

We propose a *Kinematics Rollout* mechanism. During the inference interval T_{lat}, the robot continues to execute actions from the previous chunk's buffer $\mathbf{A}_{buffer}$. We can explicitly calculate the future proprioceptive state $\hat{\mathbf{q}}_{future}$ by simulating the forward kinematics f_{kin}:

$$\hat{\mathbf{q}}_{future} = f_{kin}\left(\mathbf{q}_{query}, \sum_{i=0}^{T_{lat}} \mathbf{A}_{buffer}[i]\right) \tag{5}$$

This predicted state $\hat{\mathbf{q}}_{future}$ is injected into the VLA as the proprioceptive conditioning. To align the model with this inference logic, we employ a randomized time-shift strategy during training. By pairing the visual observation at t with the proprioceptive state and action targets from a stochastically sampled future timestep $t + \delta$, the model is forced to correlate past visual cues with future proprioceptive states across varying temporal scales. This ensures the planner remains robust to fluctuating system latencies. Consequently, the generative model learns the mapping:

$$(\text{Past Vision}, \text{Future Robot State}) \xrightarrow{\pi} \text{Future Actions} \tag{6}$$

This effectively shifts the planner's temporal frame of reference to the moment of execution, eliminating the state drift caused by latency.

3.4 The Fast Stream: Res-Refiner

While the Flow-Planner ensures global optimality and handles latency, it operates open-loop within the execution horizon (typically 0.5s - 1s). The Res-Refiner bridges the *Control Gap* by providing high-frequency (30Hz), closed-loop feedback.

The Res-Refiner adopts a pre-trained visual encoder and a multi-head actor-critic architecture optimized for pixel-based reinforcement learning.

Spatial-Learned Visual Encoding: We employ a ResNet-10 backbone He et al. (2016) followed by a Learnable Spatial Embedding layer. This layer uses a set of N_s learned spatial blocks ($N_s = 8$ in our implementation) to extract critical geometric features from the image feature map $\Phi \in \mathbb{R}^{C\times H'\times W'}$. The resulting representation $\mathbf{z}_{vis_res} \in \mathbb{R}^{256}$ captures the precise spatial relationships between the robot end-effector and the object.

Multi-Modal Policy Network: The policy network is a MLP with two hidden layers of 256 units each. It performs early fusion by concatenating the visual encoding $\mathbf{z}_{vis_res}$ from multiple camera streams, the robot's proprioceptive state $\mathbf{q}_t$, and the macro-action $\mathbf{a}_t^{slow}$ provided by the planner. To prevent over-fitting and enhance robustness, we incorporate DrQ-style data augmentation, applying random pixel shifts (± 4 pixels) to the visual input during training. The network outputs the residual correction $\Delta\mathbf{a}_t$ through a Tanh-squashed multivariate Normal distribution:

$$\begin{aligned} \mathbf{h} &= \text{Concat}(\mathbf{z}_{vis_res}, \mathbf{q}_t, \mathbf{a}_t^{slow}) \\ \Delta\mathbf{a}_t &= \alpha_{scale} \cdot \text{Tanh}\left(\text{MLP}(\mathbf{h})\right) \end{aligned} \tag{7}$$

3.5 Training Strategy

Stage 1: Motion-Aware SFT with Time-Shift. We first train the Flow-Planner using a dataset of dynamic demonstrations $\mathcal{D} = \{(\mathbf{I}_t, \mathbf{q}_t, \mathbf{A}_t)\}$. The objective is to minimize the Conditional Flow Matching loss. To endow the model with Future State Awareness, we apply a Time-Shift Augmentation to the dataloader. For a training sample recorded at time t, we construct the input-target pair as follows: **Input:** Visual observation at time t ($\mathbf{I}_t$) + Proprioceptive state at time $t+\Delta_{steps}$ ($\mathbf{q}_{t+\Delta}$). **Target:** Action chunk starting at time $t + \Delta_{steps}$ ($\mathbf{A}_{t+\Delta}$). The loss function regresses the vector field v_θ against the target vector field u_t:

$$\begin{aligned} \mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t,\mathbf{x}_0,\mathbf{A}_{gt}}\Big[\|v_\theta(\psi_t(\mathbf{x}_0), t, \mathbf{z}_{vis}, \mathbf{q}_{t+\Delta}) \\ - u_t(\psi_t(\mathbf{x}_0)|\mathbf{A}_{t+\Delta})\|^2\Big] \end{aligned} \tag{8}$$

Table 1: Real-world success rates on dynamic manipulation tasks (5 repeated 50-trials).

Task	Baseline (PI0.5)	RTC	VLASH	DynamicVLA	**DSDyn-VLA (Ours)**
Conveyor Picking	8.8±3.6%	12.4±3.8%	20.0±4.5%	33.2±5.4%	**44.8±4.1%**
Dynamic Dropping	14.8±4.6%	43.6±5.7%	50.0±5.5%	56.0±5.1%	**62.8±5.4%**
Dynamic Stacking	1.2±1.8%	2.4±2.6%	6.8±4.1%	18.0±5.1%	**32.0±4.5%**
Mobile Pouring	8.8±4.6%	16.0±4.5%	30.4±5.7%	45.2±6.1%	**56.4±5.7%**
Average	8.4±3.6%	18.6±4.1%	26.8±4.9%	38.1±5.4%	**49.0±4.9%**

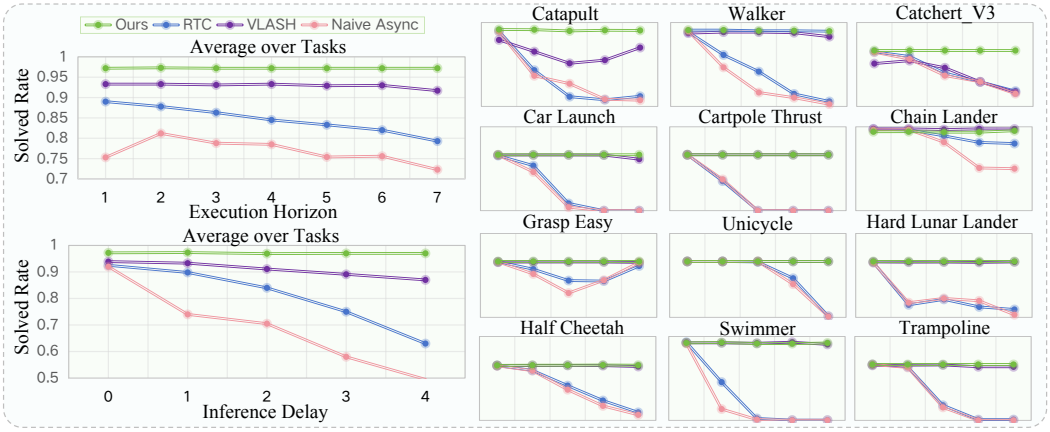

Figure 4: Performance comparison on the Kinetix benchmark. **Top left:** Execution horizon vs. average solve rate with a fixed inference delay of 1. **Bottom left:** Inference delay vs. average solve rate with a fixed execution horizon = max (inference delay, 1). **Right:** Inference delay vs. solve rate for individual tasks. Each setting consists of 2048 trials.

This forces the model to exploit the motion dynamics encoded in $\mathbf{z}_{vis}$ to learn the causal relationship between current visual cues and future actions, implicitly modeling the latency physics.

Stage 2: Residual RL Fine-Tuning. In the second stage, we freeze the Flow-Planner parameters θ to preserve its semantic generalization. We train the Res-Refiner parameters ϕ using RLPD Ball et al. (2023), a sample-efficient RL algorithm. The Res-Refiner interacts with the environment, receiving sparse binary rewards $r_{success}$. To accelerate learning, we maintain a Replay Buffer that mixes 50% offline demonstrations and 50% online experiences. We utilize an ensemble of N_c critics $\{Q_i\}_{i=1}^{N}$ to mitigate Q-value overestimation. The critic loss minimizes the Bellman error averaged over the ensemble:

$$\mathcal{L}_{critic}(\phi) = \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\mathcal{B}}\Big[\Big(Q_i(\mathbf{s},\mathbf{a}) - (r+\gamma \max_{\mathbf{a}'}\min_{j} Q_j(\mathbf{s}',\mathbf{a}'))\Big)^2\Big] \tag{9}$$

The actor is updated to maximize the Q-value subject to entropy regularization. Through this stage, the Res-Refiner learns to compensate for the noise and dynamic perturbations that the open-loop Flow-Planner cannot anticipate.

4 Experiments

4.1 Simulation Benchmark on Kinetix

Setup and Baselines. We utilize Kinetix Matthews et al. (2024), a benchmark specifically designed for evaluating dynamic policies. To ensure a fair comparison in this state-based environment, we adapt our Flow-Planner to an MLP-based policy predicting action chunks ($H = 8$), while the Res-Refiner serves as a lightweight residual network. Note that the optical flow module is omitted here as Kinetix provides direct state observations. We benchmark against two SOTA real-time manipulation methods: RTC Black et al. (2025a) and VLASH Tang et al. (2025) (Vision-based DynamicVLA Xie et al. (2026) cannot work in state-based Kinetix simulation benchmark). System latency is simulated by introducing an artificial delay $\delta \in \{0, 1, 2, 3, 4\}$ steps between observation and execution.

Table 2: Ablation study of average success rates on Kinetix under varying latencies (δ) and execution horizons (K). We evaluate different execution sizes for each delay setting. DSDyn-VLA demonstrates superior performance compared to ablated variants.

	Delay $\delta = 0$		Delay $\delta = 1$		Delay $\delta = 2$		Delay $\delta = 3$		Delay $\delta = 4$
Execution Horizon (K)	1	8	1	7	2	6	3	5	4
DSDyn-VLA	97.2%	97.2%	97.3%	97.1%	96.9%	97.3%	97.0%	97.3%	97.1%
w/o Future	97.0%	97.1%	93.6%	93.1%	91.6%	91.9%	88.8%	89.6%	85.6%
w/o Refiner	93.9%	90.9%	93.3%	91.7%	91.1%	91.3%	89.1%	89.1%	87.0%

Table 3: Results on our DynBench. Each method is evaluated with 100 trials for each task.

Method	Pick	Drop	Stack	Pour	Sort	Insert	Ball	Can	GreenBall	Avg
PI0.5	12%	14%	1%	10%	19%	1%	18%	10%	2%	9.7%
RTC	16%	41%	7%	13%	25%	1%	32%	25%	15%	19.4%
VLASH	19%	55%	8%	30%	37%	12%	35%	32%	16%	27.1%
DynamicVLA	28%	55%	19%	48%	43%	17%	44%	36%	19%	34.3%
DSDyn-VLA w/o Refiner	49%	56%	32%	54%	49%	19%	46%	48%	31%	42.7%
DSDyn-VLA w/o Future	37%	58%	30%	54%	53%	24%	50%	42%	36%	42.7%
DSDyn-VLA w/o Flow	36%	48%	23%	50%	33%	17%	44%	37%	30%	35.3%
DSDyn-VLA	**51%**	**62%**	**37%**	**64%**	**53%**	**25%**	**53%**	**51%**	**40%**	**48.4%**

Overall Performance. Figure 4 illustrates the aggregate success rate across 12 Kinetix tasks. Under the minimal latency ($\delta = 0$), all methods perform comparably. However, as the inference delay increases—mimicking the computation load of large VLA models—the performance of RTC and VLASH degrades rapidly. In contrast, DSDyn-VLA exhibits exceptional robustness, maintaining a success rate of over 97% even at the most severe latency setting ($\delta = 4$).

Impact of Modules. The quantitative results in Table 2 isolate the contribution of each component. Removing the future prediction mechanism (*w/o Future*) leads to a significant performance drop (-11.5% at $\delta = 4$). Furthermore, removing the fast stream (*w/o Refiner*) results in unstable execution (-10.1%), highlighting that the open-loop execution of chunks fails to correct for errors, whereas the Res-Refiner successfully stabilizes the trajectory.

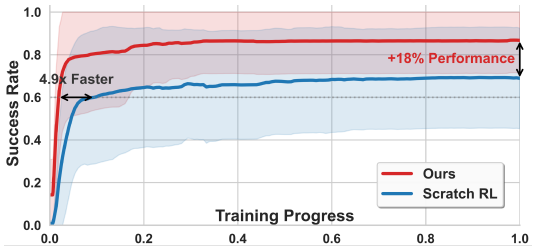

Figure 5: Training efficiency and performance comparison on Kinetix tasks. The **Res-Refiner** (Ours, red) significantly outperforms the standard RL baseline (Blue), achieving **4.9x faster convergence** and a **+18% higher success rate**.

Training Efficiency and Residual Efficacy. To verify that the performance gains stem from the Dual-Stream framework rather than simply training a strong RL policy, we conducted an analysis on 12 Kinetix tasks. Figure 5 compares the training curves of our *Res-Refiner* against a standard RL policy trained from scratch with the same network architecture. Results indicate that the Res-Refiner demonstrates significantly superior sample efficiency, achieving convergence **4.9x faster** than Scratch-RL. Moreover, the average peak success rate of Res-Refiner is substantially higher, delivering a **+18% performance gain** over the baseline. This suggests that learning a *residual* on top of Flow-planner is a much easier optimization problem than learning global control dynamics from scratch, validating the "Slow-Fast" design philosophy.

4.2 DynBench Experiments

Details of DynBench. DynBench is a MuJoCoTodorov et al. (2012)-based benchmark for dynamic manipulation, designed to evaluate the full visual pipeline of DSDyn-VLA as a complement to the state-based Kinetix benchmark. We construct DynBench by porting four real-world dynamic tasks into simulation with a Franka Panda robot arm and further expanding them into a 9-task suite, including Conveyor Picking, Dynamic Dropping, Dynamic Stacking, Mobile Pouring, category-aware conveyor sorting, cylinder insertion into a moving hole, grasping a rolling ball, grasping a rolling can, and selecting the green ball when red and green balls roll in together. Each task is accompanied by 100 rule-generated demonstrations to support consistent training and evaluation.

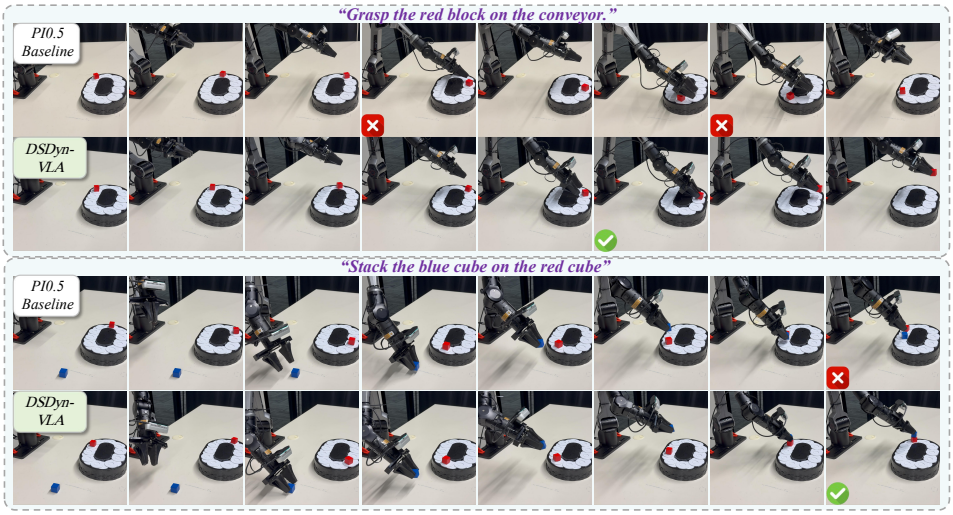

Figure 6: Case comparison of our DSDyn-VLA and baseline (PI0.5).

Results on DynBench. We further evaluate both baseline methods and ablated variants on Dyn-Bench, with 100 trials for each task. As shown in Table 3, DSDyn-VLA achieves the best average success rate of 48.4%, substantially outperforming PI0.5, RTC, VLASH, and DynamicVLA. The ablation results further verify the effectiveness of all three key components: removing the Refiner or Future module reduces the average success rate to 42.7%, while removing the Flow branch leads to a larger drop to 35.3%. These results show that Dyn-Bench provides a controlled yet visually grounded testbed, and further confirms the advantage of our full system under dynamic manipulation settings.

4.3 Real-World Experiments

Hardware and Task Setup. We validate DSDyn-VLA on a real-world robotic setup using a 6-DOF Agilex Piper arm equipped with two Intel RealSense D435 cameras. The Slow Stream is powered by a **PI0.5** backbone, fine-tuned with a chunk size of $H = 10$. The Res-Refiner operates at 30Hz. We design four challenging dynamic tasks: (1) **Conveyor Picking**, where the robot must grasp a target cube moving on a conveyor belt; (2) **Dynamic Dropping**, where the robot must release a held object into a moving container; (3) **Dynamic Stacking**, requiring the robot to pick a cube and stack it onto another moving cube; and (4) **Mobile Pouring**, a high-precision task where the robot pours water into a moving cup. In the real-world setting, the binary reward for RL is provided manually: once the task is observed to be successfully completed, a reward of 1 is assigned. RL stage takes about 30-60 mins on each task.

Performance Analysis. Table 1 quantifies the substantial gap between comparative baselines and our method over 5×50 trials per task.

Ablation on Design Choices. To justify our specific architectural decisions, we performed a fine-grained ablation study on four *Real-World* tasks (Table 4).

Table 4: Design choice ablation on four real-world task (50 trials). We compare the full model against variants with alternative visual encodings or training strategies.

Method Variant	Success Rate
DSDyn-VLA (Full)	**48.5%**
w/o Optical Flow	30.5%
w/o Polar Map	33.0%
w/o Future State Awareness	36.5%
w/o Res-Refiner	25.0%
Res-Refiner (*w/ SFT*)	37.5%

1) Impact of Optical Flow (*w/o Optical Flow*). Removing the optical flow input completely causes a catastrophic drop in success rate to 30.5%. This result confirms that standard static RGB inputs lack the necessary high-order derivative information (velocity) required to track moving objects. Without explicit motion cues, the VLA acts on outdated positional information, failing to intercept the target.

2) Impact of Polar Mapping (*w/o Polar Map*). Replacing our polar mapping with a ResNet-34 encoder that projects raw flow directly into the VLA's token space yields a suboptimal success rate of 33.0%. We attribute this to *feature space misalignment*: the projected tokens from a scratch-trained encoder lack the semantic grounding expected by the VLA backbone. In contrast, our polar mapping

translates motion into color and utilize CLIP as encoder, effectively aligning geometric dynamics with the VLA's inherent semantic understanding of the color space.

3) Impact of Future State Awareness (*w/o Future*). Disabling the Future State Awareness mechanism results in a success rate of 36.5%. Without this module, the Flow-Planner generates actions relative to the robot's state at the moment of observation (t), ignoring the system latency (T_{lat}). By the time these actions are executed at $t + T_{lat}$, the robot has moved, leading to a coordinate frame mismatch. The drop in performance validates the necessity of conditioning generation on the execution-time state.

4) Impact of Closed-Loop Correction (*w/o Res-Refiner*). Removing the *Res-Refiner* drops performance to 25.0%, highlighting the severity of the *Control Gap*. Even with perfect planning, open-loop execution of an action chunk (10 steps) leaves the robot blind to real-time perturbations. The Res-Refiner's ability to inject high-frequency corrections (30Hz) is critical for adjusting the end-effector pose during the final approach phase.

5) Impact of RL Training (Res-Refiner *w/ SFT*). Training the Res-Refiner via SFT instead of RL recovers performance only to 37.5%, significantly lower than the RL-trained counterpart. We attribute this to the distribution shift problem: SFT policies struggle to recover from states that deviate from expert demonstrations. In contrast, our RL-based refiner actively learns a robust recovery policy, proving that the reinforcement learning formulation is essential for the fast stream's reactivity.

Robustness to degraded optical flow. We further evaluate DSDyn-VLA under degraded visual conditions to test its sensitivity to imperfect optical flow. As shown in Table 7, the performance drops gradually rather than collapsing, indicating that DSDyn-VLA is robust to degraded optical flow.

Module-level runtime profiling. We profile each module on a RTX4090. The additional motion-perception and residual-correction components are lightweight compared with the base VLA backbone, introducing only modest runtime overhead, as summarized in Table 5.

Table 5: Module-level runtime profiling.

Component	Latency
VLA backbone total	73 ms
-image encoder	14 ms
-VLM prefill	32 ms
-action expert decoding	27 ms
System latency	13 ms
RAFT-Large	3 ms
Flow encoder	14 ms
Kinematics rollout	$<$ 1 ms
Flow-Planner total	103 ms
Res-Refiner total	3 ms

Table 6: Residual magnitude statistics on Dynamic Stacking and Mobile Pouring.

Statistic	Ratio
All dims $<$ 80% of range	95.8%
All dims $<$ 95% of range	99.1%

Table 7: Robustness under degraded optical-flow conditions, averaged over four real-world tasks.

Condition	Avg. Success
Clean	48.5%
20% occlusion	47.0%
Low light	45.5%
High light	46.0%
Additive noise	45.5%
50% occlusion	43.5%
70% occlusion	42.0%
Reflective stickers	42.5%
Blinking light	40.5%

Is the residual range sufficient? The Res-Refiner predicts residuals bounded to 10% of the full action range per dimension. To verify that this constraint is not overly restrictive, we measure the residual magnitudes on Dynamic Stacking and Mobile Pouring. Table 6 shows that the residual branch rarely saturates, suggesting that this budget is sufficient for local closed-loop correction.

5 Conclusion

We present DSDyn-VLA, a dual-stream framework for dynamic manipulation that integrates motion perception, future-state awareness, and real-time residual correction into a unified control paradigm. Across both simulation and real-world tasks, DSDyn-VLA consistently outperforms strong baselines. While there remains room to further broaden residual-policy generalization and expand real-world evaluation across more tasks and platforms, the current results already suggest strong potential for applications such as industrial handling and assistive robotics. More broadly, this work may broaden the applicability of robotic manipulation in dynamic settings, while also raising the need for careful handling of safety and failure risks in human-shared environments.

AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript to improve its readability and to assist with formatting references. We have not used generative AI tools for generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, interpreting results, or cleaning and reformatting datasets. We have reviewed all AI-assisted edits to the text. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide source code for DSDyn-VLA and DynBench, including the training and evaluation pipelines for the Flow-Planner and Res-Refiner, together with demonstration videos, in the supplementary materials. Section 3 describes the model architecture and two-stage training procedure. Section 4 presents the experimental setups for Kinetix, DynBench, and real-world tasks, while Appendix A.1 reports additional evaluations on LIBERO and DOMINO. Model weights will be released upon publication.

REFERENCES

Philip J Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, pp. 1577–1594. PMLR, 2023.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi0: A vision-language-action flow model for general robot control. corr, abs/2410.24164, 2024. doi: 10.48550. *arXiv preprint ARXIV.2410.24164*, 2024.

Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. *arXiv preprint arXiv:2506.07339*, 2025a.

Kevin Black, Allen Z Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking. *arXiv preprint arXiv:2512.05964*, 2025b.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Yin Cao, Xuemei Xu, and Yazheng Zhang. Vision-based reinforcement learning for robotic grasping of moving objects on a conveyor. *Machines*, 13(10):973, 2025.

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.

Frederik Ebert, Sudeep Dasari, Alex X Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. In *Conference on robot learning*, pp. 983–993. PMLR, 2018.

Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards generalizable robotic manipulation in dynamic environments. *arXiv preprint arXiv:2603.15620*, 2026.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

Dong Han, Beni Mulyana, Vladimir Stankovic, and Samuel Cheng. A survey on deep reinforcement learning algorithms for robotic manipulation. *Sensors*, 23(7):3762, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Chien-Ming Huang, Maya Cakmak, and Bilge Mutlu. Adaptive coordination strategies for human-robot handovers. In *Robotics: science and systems*, volume 11, pp. 1–10. Rome, Italy, 2015.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

Xiaoqian Huang, Mohamad Halwani, Rajkumar Muthusamy, Abdulla Ayyad, Dewald Swart, Lakmal Seneviratne, Dongming Gan, and Yahya Zweiri. Real-time grasping strategies using event camera. *Journal of Intelligent Manufacturing*, 33(2):593–615, 2022.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. pi0.6*: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi0.5: a vision-language-action model with open-world generalization, 2025. *URL https://arxiv. org/abs/2504.16054*, 1(2):3, 2025b.

Yuhua Jiang, Shuang Cheng, Yan Ding, Feifei Gao, and Biqing Qi. Asyncvla: Asynchronous flow matching for vision-language-action models. *arXiv preprint arXiv:2511.14148*, 2025.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Bin Li, Hu Cao, Zhongnan Qu, Yingbai Hu, Zhenke Wang, and Zichen Liang. Event-based robotic grasping detection with neuromorphic vision sensor and event-grasping dataset. *Frontiers in neurorobotics*, 14:51, 2020.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. *arXiv preprint arXiv:2410.07864*, 2024a.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. *arXiv preprint arXiv:2408.17355*, 2024b.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

Michael Matthews, Michael Beukman, Chris Lu, and Jakob Foerster. Kinetix: Investigating the training of general agents through open-ended physics-based control tasks. *arXiv preprint arXiv:2410.23208*, 2024.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022.

Rajkumar Muthusamy, Abdulla Ayyad, Mohamad Halwani, Dewald Swart, Dongming Gan, Lakmal Seneviratne, and Yahya Zweiri. Neuromorphic eye-in-hand visual servoing. *IEEE Access*, 9: 55853–55870, 2021.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903. IEEE, 2024.

Samuel Pfrommer, Yixiao Huang, and Somayeh Sojoudi. Reinforcement learning for flow-matching policies. *arXiv preprint arXiv:2507.15073*, 2025.

Daniel Seita, Yufei Wang, Sarthak J Shetty, Edward Yao Li, Zackory Erickson, and David Held. Toolflownet: Robotic manipulation with tools via predicting tool flow from point clouds. In *Conference on Robot Learning*, pp. 1038–1049. PMLR, 2023.

Kohei Sendai, Maxime Alvarez, Tatsuya Matsushima, Yutaka Matsuo, and Yusuke Iwasawa. Leave no observation behind: Real-time correction for vla action chunks. *arXiv preprint arXiv:2509.23224*, 2025.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, pp. 894–906. PMLR, 2022.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Bing Sun, Hongjun Ma, and Daqi Zhu. A fusion designed improved elastic potential field method in auv underwater target interception. *IEEE Journal of Oceanic Engineering*, 48(3):640–648, 2023.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. Vlash: Real-time vlas via future-state-aware asynchronous inference. *arXiv preprint arXiv:2512.01031*, 2025.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pp. 5026–5033. IEEE, 2012.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

Thomas Weng, Sujay Man Bajracharya, Yufei Wang, Khush Agrawal, and David Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, pp. 192–202. PMLR, 2022.

Rosa Wolf, Yitian Shi, Sheng Liu, and Rania Rayyes. Diffusion models for robotic manipulation: A survey. *Frontiers in Robotics and AI*, 12:1606247, 2025.

Ziwei Xia, Zhen Deng, Bin Fang, Yiyong Yang, and Fuchun Sun. A review on sensory perception for dexterous robotic manipulation. *International Journal of Advanced Robotic Systems*, 19(2): 17298806221095974, 2022.

Haozhe Xie, Beichen Wen, Jiarui Zheng, Zhaoxi Chen, Fangzhou Hong, Haiwen Diao, and Ziwei Liu. Dynamicvla: A vision-language-action model for dynamic object manipulation. *arXiv preprint arXiv:2601.22153*, 2026.

Shaopeng Zhai, Qi Zhang, Tianyi Zhang, Fuxian Huang, Haoran Zhang, Ming Zhou, Shengzhe Zhang, Litao Liu, Sixu Lin, and Jiangmiao Pang. A vision-language-action-critic model for robotic real-world reinforcement learning. *arXiv preprint arXiv:2509.15937*, 2025.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *arXiv preprint arXiv:2505.22094*, 2025.

Yongsheng Zhao, Lei Zhao, Baoping Cheng, Gongxin Yao, Xuanzhang Wen, and Han Gao. Vla-rail: A real-time asynchronous inference linker for vla models and robots. *arXiv preprint arXiv:2512.24673*, 2025.

Zhide Zhong, Haodong Yan, Junfeng Li, Xiangchen Liu, Xin Gong, Tianran Zhang, Wenxuan Song, Jiayi Chen, Xinhu Zheng, Hesheng Wang, et al. Flowvla: Visual chain of thought-based motion reasoning for vision-language-action models. *arXiv preprint arXiv:2508.18269*, 2025.

A Appendix

A.1 External Benchmark Experiments

Static Manipulation on LIBERO. Since the Res-Refiner does not take text embeddings as input, it is necessary to verify that the added streams of DSDyn-VLA do not hurt performance on general static manipulation tasks that require precise instruction following. We evaluate DSDyn-VLA on the four LIBERO Liu et al. (2023) suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-10. As shown in Table 8, DSDyn-VLA matches or outperforms the PI0.5 baseline on all four suites, indicating that the introduced motion perception and residual correction streams preserve the static manipulation and instruction-following capability of the underlying VLA. We attribute this to the fact that the Res-Refiner learns via RL to repair suboptimal actions, which also reduces failures on static tasks.

Dynamic Manipulation on DOMINO. To further validate generalization beyond Kinetix and DynBench, we evaluate DSDyn-VLA on DOMINO Fang et al. (2026), a recently proposed benchmark featuring complex object dynamics for dynamic manipulation. We compare against the PI0.5 baseline and PUMA, the state-of-the-art dynamics-aware method introduced in the same work, using the success rate (SR) and the manipulation score (MS), a continuous execution-quality metric. As shown in Table 9, DSDyn-VLA surpasses PUMA on both metrics, improving SR from 17.18% to 20.74% and MS from 35.08 to 41.64. These results on an external dynamic benchmark further confirm the effectiveness and generalizability of DSDyn-VLA.

B Related Work

B.1 RL Finetuning for VLA Models

RL has proven critical for enhancing VLA robustness by exploring corner cases uncovered by SFT. While methods like VLAC Zhai et al. (2025) and VLA-RL Lu et al. (2025) have successfully fine-tuned autoregressive VLAs, they are incompatible with Flow Matching architectures. Although some theoretical works explore RL for flow-based policies Zhang et al. (2025); Pfrommer et al. (2025), they are difficult to scale. PI0.6* Intelligence et al. (2025a) recently demonstrated real-world fine-tuning of a flow-based VLA. However, it relies on a binarized advantage, which reduces RL to a weakened form of conditional behavior cloning. Our DSDyn-VLA addresses this by training a external non-flow module via online RL to augment the VLA.

B.2 Methods for Perception Gap

For the perception gap, prior work on dynamic manipulation has mainly explored two directions. One line Muthusamy et al. (2021); Huang et al. (2022); Li et al. (2020) uses event cameras for fast visual servoing and grasping, benefiting from high temporal resolution and low latency, but these methods typically rely on specialized neuromorphic sensors and a more limited data ecosystem than standard RGB pipelines. Another line Weng et al. (2022); Seita et al. (2023) exploits optical flow for dynamic control and policy learning, showing that motion cues are valuable for manipulation; however, such methods are often task-specific and do not naturally interface with large RGB-pretrained VLA

Table 8: Success rates (%) on the four LIBERO suites (mean ± std over 3 seeds). The dynamic-oriented components of DSDyn-VLA do not degrade performance on static instruction-following tasks.

Method	Spatial	Object	Goal	LIBERO-10
PI0.5	98.67±0.23	98.30±0.10	97.87±0.42	92.60±0.17
DSDyn-VLA	**99.17±0.15**	**98.73±0.06**	**98.17±0.29**	**93.30±0.30**

Table 9: Results on the DOMINO benchmark for dynamic manipulation (mean ± std over 3 seeds).

Method	SR (%)	MS
PI0.5	9.63±0.08	26.19±0.41
PUMA	17.18±0.16	35.08±0.13
DSDyn-VLA	**20.74±0.25**	**41.64±0.66**

Table 10: Detailed ablation results on 12 Kinetix tasks under varying latencies ($\delta \in \{0, 2, 4\}$) and execution horizon of 4. We compare the full DSDyn-VLA against variants lacking future state awareness (*w/o Future*) or the residual refinement stream (*w/o Refiner*).

Type	Task	*w/o Future*			*w/o Refiner*			DSDyn-VLA		
		$\delta=0$	$\delta=2$	$\delta=4$	$\delta=0$	$\delta=2$	$\delta=4$	$\delta=0$	$\delta=2$	$\delta=4$
High-Dyn	Catapult	91.2%	49.3%	41.5%	73.1%	51.7%	56.1%	90.9%	92.2%	90.6%
	Walker	90.9%	75.4%	0.4%	87.4%	86.8%	86.9%	91.0%	89.9%	89.7%
	Catcher_V3	99.9%	99.8%	99.7%	73.5%	70.4%	21.7%	99.7%	99.7%	99.7%
Regular	Car Launch	99.5%	99.2%	99.5%	98.9%	99.4%	99.3%	99.4%	99.5%	99.4%
	Cartpole Thrust	100%	100%	100%	100%	100%	100%	100%	100%	99.9%
	Chain lander	95.2%	95.9%	97.5%	97.6%	97.9%	97.5%	94.6%	95.2%	95.3%
	Grasp Easy	99.9%	100%	100%	99.0%	99.0%	99.0%	99.9%	99.8%	100%
	Unicycle	100%	100%	100%	100%	100%	100%	100%	100%	100%
	Hard Lunar Lander	100%	100%	99.8%	98.2%	98.8%	98.6%	100%	100%	100%
	Half Cheetah	98.7%	98.0%	97.8%	98.5%	98.0%	97.9%	98.9%	97.9%	98.7%
	Swimmer	92.2%	92.6%	91.3%	92.9%	92.2%	93.4%	92.2%	92.5%	92.2%
	Trampoline	99.5%	99.5%	99.9%	98.1%	99.2%	94.1%	99.7%	99.8%	99.3%
Avg (12 Tasks)		97.2%	92.5%	85.6%	93.1%	91.1%	87.0%	97.2%	97.2%	97.1%

backbones. In contrast, our method stays within a standard RGB setup and introduces motion cues through a polar-mapped optical-flow representation, which is more compatible with CLIP-style visual encoders.

B.3 Methods for Control Gap

For the control gap, prior embodied-control methods have improved execution-time reactivity mainly through continual replanning or closed-loop action updating. Representative examples include video-prediction Finn & Levine (2017) or MPC-style replanning Ebert et al. (2018), receding-horizon control Chi et al. (2025), and test-time resampling for chunked actions Liu et al. (2024b), all of which aim to restore sensitivity to the latest observation during execution. These methods highlight the importance of closing the loop online, but often require repeated full replanning or expensive sampling. By contrast, our approach preserves the VLA's chunk-level planning ability while introducing a lightweight high-frequency residual correction module, making it more suitable for dynamic manipulation under nontrivial inference latency.

C Detailed Ablation Results on Kinetix Benchmark

We provide the detailed results of the ablation experiments on Kinetix in Table 10.

Table 11: Fairer comparison with temporally strengthened baselines and ablations on motion representation. For a fair comparison, we augment RTC, VLASH, and DYNAMICVLA with two historical frames and 1 hour of end-to-end PPO training. We further compare several motion-input variants of our DSDYN-VLA.

Setting	Avg. Success (%)
RTC	18.0
RTC + history frames + PPO	24.0
VLASH	26.5
VLASH + history frames + PPO	32.5
DynamicVLA	37.5
DynamicVLA + history frames + PPO	39.0
DSDyn-VLA	**48.5**
DSDyn-VLA w/ frame stacking	42.5
DSDyn-VLA direct encoding	33.0
DSDyn-VLA learnable MLP	37.0

D FAIRER TEMPORAL BASELINES AND OPTICAL-FLOW ENCODING VARIANTS

To provide a fairer comparison with prior dynamic-manipulation baselines, we strengthen RTC, VLASH, and DYNAMICVLA with two additional historical frames and further perform 1 hour of end-to-end PPO training for each method. This setting reduces the concern that our gains might simply come from providing richer temporal context or additional policy adaptation.

In addition, we conduct several variants on our full model DSDYN-VLA to better understand the role of motion representation. Specifically, we compare the full model against: (1) replacing optical flow with simple frame stacking, (2) directly encoding optical flow with a ResNet-based encoder, and (3) encoding optical flow with a learnable MLP adapter. The results are reported in Table 11.

We make three observations. First, adding historical frames and PPO training consistently improves all strengthened baselines: RTC improves from 18.0 to 24.0, VLASH improves from 26.5 to 32.5, and DYNAMICVLA improves from 37.5 to 39.0. This confirms that temporal context and additional policy optimization are indeed helpful for dynamic manipulation. However, even after these enhancements, all baselines remain clearly below DSDYN-VLA, which achieves 48.5 average success.

Second, replacing optical flow with naive frame stacking leads to a noticeable drop from 48.5 to 42.5. This suggests that the benefit of our method does not come merely from exposing the model to more temporal observations; rather, explicitly modeling motion is more effective than asking the network to infer it implicitly from stacked RGB frames.

Third, alternative optical-flow encoding strategies are substantially less effective than our full design. Direct optical-flow encoding yields only 33.0, while a learnable MLP adapter reaches 37.0, both significantly below the full DSDYN-VLA. These results indicate that the performance gain is not obtained by simply adding an extra flow branch. Instead, the way motion cues are represented and aligned with the visual backbone is critical for effective dynamic manipulation.

Overall, these results strengthen two conclusions: (i) our advantage cannot be explained solely by stronger temporal input or extra RL fine-tuning, and (ii) explicit and properly encoded motion representation is a key ingredient behind the performance of DSDYN-VLA.

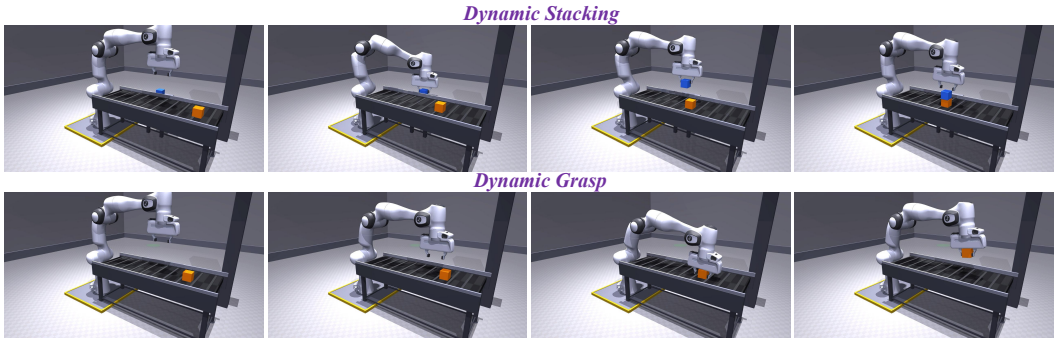

Figure 7: Qualitative rollouts of DSDyn-VLA on DynBench. Top: Dynamic Stacking. Bottom: Dynamic Grasp. From left to right, the snapshots show that DSDyn-VLA tracks moving objects and completes the task under continuous motion.