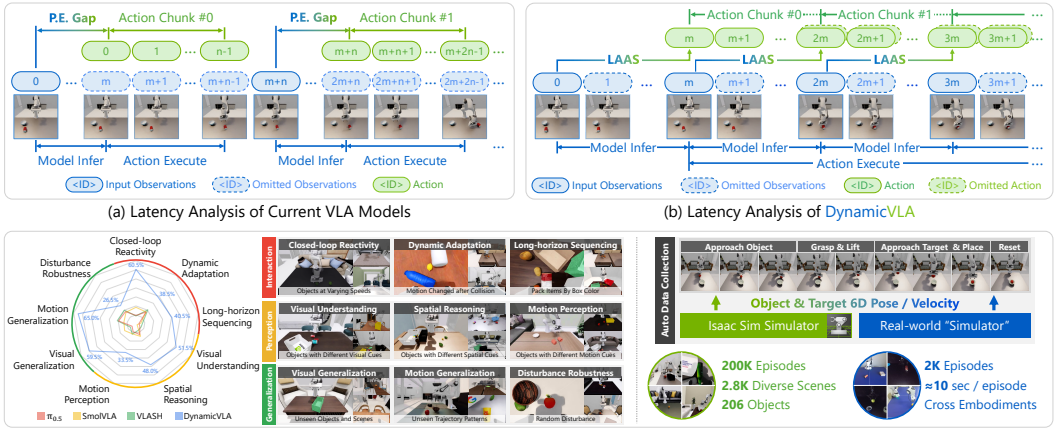
DynamicVLA: A Vision-Language-Action Model for Dynamic Object Manipulation

Haozhe Xie* **Beichen Wen*** **Jiarui Zheng** **Zhaoxi Chen** **Fangzhou Hong** **Haiwen Diao** **Ziwei Liu** ✉
S-Lab, Nanyang Technological University
https://haozhexie.com/project/dynamic-vla



(a) Latency Analysis of Current VLA Models

(b) Latency Analysis of DynamicVLA

(c) Dynamic Object Manipulation (DOM) Benchmark

Figure 1: **(a)** Current VLA models suffer from inter-chunk waiting and perception–execution (P.E.) mismatch, leading to delayed or misaligned reactions to moving objects. **(b)** DynamicVLA uses a continuous inference schedule to maintain a non-blocking action stream, while LAAS discards latency-invalid prefixes and executes only temporally valid suffixes to preserve action-time alignment under dynamic object motion. **(c)** The Dynamic Object Manipulation (DOM) benchmark features 2.8K scenes and 206 objects for evaluating Perception, Interaction, and Generalization, while its data collection pipeline enables gathering of 200K synthetic and 2K real-world episodes.

Abstract

Manipulating dynamic objects remains an open challenge for Vision-Language-Action (VLA) models. Although recent VLAs generalize well in static manipulation, dynamic scenes introduce a latency-induced perception–execution mismatch: object states continue to evolve during inference, making actions predicted from past observations stale at execution time. We present DynamicVLA, a latency-aware VLA model for dynamic object manipulation. It combines a compact 0.4B architecture and convolutional vision encoder for efficient multimodal inference with a continuous inference schedule that overlaps reasoning and execution for non-blocking control. Latency-aware Action Streaming then discards latency-invalid action prefixes and executes only the temporally valid suffix of each predicted chunk, preserving action-time alignment under dynamic object motion. To fill the missing foundation of dynamic manipulation data, we introduce the Dynamic Object Manipulation (DOM) benchmark, built with an automated collection pipeline that gathers 200K synthetic episodes across 2.8K scenes and 206 objects, and enables fast collection of 2K real-world episodes without teleoperation. Extensive evaluations in simulation and on real robots show that DynamicVLA improves dynamic manipulation success under changing object motion, perception-heavy instructions, and unseen motion patterns.

* Equal contribution. ✉ Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Dynamic object manipulation is a fundamental yet underexplored frontier in robotics. Real-world interaction often involves objects in motion, such as handing, repositioning, or stabilizing items, requiring robots to perceive, predict, and act under changing conditions. Even minor latency may cause failure, making dynamic manipulation substantially more challenging than static grasping [1].

Robots have long been evaluated on moving-target tasks such as throwing [2], soccer [3], and table tennis [4, 5], but typically rely on reactive control and handcrafted perception in structured settings. Recent manipulation methods [6, 7] handle moving objects, yet remain limited to predictable, conveyor-belt–like motion. Concurrent VLAs [8, 9, 10] demonstrate real-time interaction with fast-moving targets, but mainly in striking or interception-style tasks with coarse contact and large spatial margins. Dynamic object manipulation instead requires precise 6DoF grasping and placement under uncertain motion, making tight perception–action alignment essential and still underexplored.

VLA models [11] have achieved strong results in static manipulation, where object states remain nearly unchanged during inference and latency is often tolerable. Even large 3B–7B VLAs [12, 13] perform well despite slow inference, while recent designs [14, 15] further improve efficiency. Asynchronous action-chunking methods [16, 17] keep action chunks running and improve chunk-to-chunk continuity, but they do not guarantee temporal alignment between predicted actions and evolving object states. As shown in Figure 1a, object states keep evolving during inference, making early predicted actions temporally invalid before execution. This latency-induced perception–execution gap is mostly hidden in static benchmarks but becomes critical for dynamic objects.

To address these issues, we propose DynamicVLA, a latency-aware VLA model for dynamic object manipulation. It uses a compact 0.4B-parameter architecture with a convolutional vision encoder for efficient spatial compression and visual encoding. During execution, we use a continuous inference schedule that overlaps prediction and control, avoiding inter-chunk waiting and letting newer chunks supersede older ones at overlapping steps. Yet continuous inference maintains an uninterrupted action stream; it does not correct the temporal-index shift caused by inference latency. Since the earliest actions in each chunk are indexed to execution times that may have already elapsed, Latency-aware Action Streaming discards these latency-invalid prefixes and executes only the temporally valid suffix of each chunk, preserving action-time alignment under dynamic object motion.

A key barrier to developing and evaluating VLA policies for dynamic scenarios is the lack of large-scale data and standardized protocols. Existing robotic datasets [18, 19] focus mostly on static scenes, while teleoperation fails when objects move faster than humans can reliably track. To close this gap, we introduce Dynamic Object Manipulation (DOM), a benchmark with an automated data-collection pipeline validated on Franka Emika Panda and AgileX PiPER. In simulation, Isaac Sim [20] provides real-time 6D object poses and velocities to drive an automatic demonstrator, yielding 200K episodes across 2.8K simulation-ready 3D scenes and 206 objects. For real-world collection, a "real-world" simulator exposes the same state interface using estimated object position and velocity, enabling the same demonstrator to collect 2K episodes without teleoperation.

Our contributions are summarized as follows.

- We identify latency-induced perception–execution gap as a failure mode for VLA policies in dynamic manipulation, and present DynamicVLA, a latency-aware VLA model that combines multimodal inference with non-blocking inference and action-time-aligned streaming.
- We introduce Dynamic Object Manipulation (DOM), a large-scale dynamic manipulation benchmark with an automated data-collection pipeline, comprising 200K synthetic episodes, 2.8K scenes, 206 objects, and 2K real-world episodes across multiple embodiments.
- We conduct extensive simulation and real-robot evaluations on DOM and 16 real-robot tasks, showing that DynamicVLA improves dynamic manipulation under changing object motion, perception-heavy instructions, and unseen objects, scenes, and motion regimes.

2 Related Work

Vision-Language-Action Models. Inspired by the success of Large Language Models (LLMs) [21, 22, 23] and Vision-Language Models (VLMs) [24, 25, 26], Vision-Language-Action (VLA) models extend VLMs with action generation. Transformer-based methods [16, 27] use Transformers

to model state-action-reward sequences. LLM/VLM-based methods [28, 12] treat VLA tasks as sequence-to-sequence problems for action generation. Diffusion-based methods [29, 30] model policies as denoising diffusion models. LLM and diffusion model methods [31, 13] combine LLMs for representation and diffusion models for action generation. Video generation with inverse kinematics methods [32, 33, 34] generate motion sequences and convert them into actions. However, existing VLA models are often too slow for precise or rapid execution.

Robot Learning Datasets. Real-world datasets [35, 36, 19, 37] provide high-fidelity interactions but are costly and hard to scale, while simulated datasets [38, 18, 39, 40, 41] offer scalability yet suffer from the sim-to-real gap. Most benchmarks focus on simple tabletop manipulation (*e.g.*, pick-and-place, pushing) with limited task diversity, though recent work explores long-horizon [18, 42], language-conditioned [43, 44], and tactile-rich [45, 46] settings. Generative models [47, 48, 32, 49] introduce interactive elements but remain constrained by artifacts, low frame rates, and memory. Despite progress in standardization and multi-embodiment learning, current datasets lack dynamic objects, limiting applicability to environments with independent motion.

Robot Dynamic Manipulation. Robots have long handled moving targets in specialized settings such as table tennis [4, 5], but mostly through specialized systems. Recent moving-object manipulation methods mainly consider structured or conveyor-like motion [6, 7], while concurrent VLA efforts have begun to explore real-time or dynamic interaction [8, 9, 10]. However, general dynamic object manipulation with fine contact requirements, evaluated broadly in both simulation and the real world, remains less studied.

3 The DynamicVLA Model

3.1 Problem Formulation

We study dynamic object manipulation, where object states evolve continuously during perception, reasoning, and execution. At time step t, the VLA policy π receives a temporal window of visual observations $\mathbf{O}_t = \{\mathbf{o}_{t-k}, \ldots, \mathbf{o}_t\}$, a language instruction $\mathbf{L}_t$, and its proprioceptive state $\mathbf{P}_t$, and predicts an action sequence $\mathbf{A}_t = \{\mathbf{a}_t, \ldots, \mathbf{a}_{t+n}\}$, *i.e.*, $\mathbf{A}_t = \pi(\mathbf{O}_t, \mathbf{L}_t, \mathbf{P}_t)$.

The physical environment includes a latent object state $\mathbf{s}_t$, describing the object's 6D pose and motion. Crucially, object motion does not pause during inference: while the model reasons over $\mathbf{O}_t$, the object transitions from $\mathbf{s}_t$ to $\mathbf{s}_{t+m}$, where m denotes inference latency, leading to potential misalignment between perception and execution.

3.2 The DynamicVLA Architecture

Since inference latency directly affects whether a robot can react to moving objects, we design a 0.4B VLA model for fast and spatially efficient multimodal reasoning, illustrated in Figure 2.

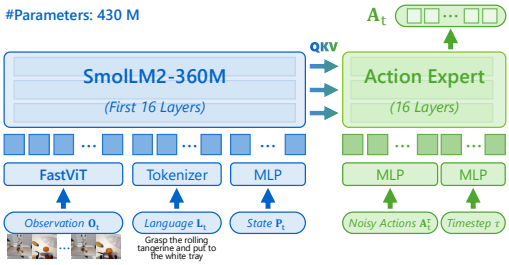


Figure 2: **The DynamicVLA architecture.** Multimodal features condition a diffusion-based action expert for action-chunk prediction.

Vision–Language Backbone. We adopt SmolLM2-360M [50] as the language backbone, resulting in an overall tiny model size. Unlike VLMs that rely on transformer-based vision encoders, we employ a convolutional vision encoder, FastViT [51], which performs efficient spatial compression and avoids quadratic token growth when processing multi-frame visual inputs. Following SmolVLA [14], we use the first 16 transformer layers of the language backbone to balance latency and multimodal reasoning capacity.

Diffusion-Based Action Expert. The action expert $\mathcal{E}_\theta$ predicts an action chunk $\mathbf{A}_t$ conditioned on the features produced by the VLM backbone. Following diffusion-style action modeling [52, 53], we instantiate $\mathcal{E}_\theta$ as a conditional Flow Matching Transformer [13] and train it using the objective

$$\ell^\tau(\theta) = \mathbb{E}_{p(\mathbf{A}_t|\mathbf{f}_t), q(\mathbf{A}_t^\tau|\mathbf{A}_t)} \left[\| \mathcal{E}_\theta(\mathbf{A}_t^\tau, \mathbf{O}_t) - \mathbf{u}(\mathbf{A}_t^\tau|\mathbf{A}_t) \| \right] \quad (1)$$

where superscript $\tau \in [0, 1]$ denotes flow matching timesteps. $q(\mathbf{A}_t^\tau|\mathbf{A}_t) = \mathcal{N}(\tau\mathbf{A}_t, (1-\tau)\mathbf{I})$. $\mathbf{f}_t$ represents the VLM features extracted from $\mathbf{O}_t$, and $\mathbf{A}_t^\tau = \tau\mathbf{A}_t + (1-\tau)\epsilon$, with $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Under this objective, $\mathcal{E}_\theta(\mathbf{A}_t^\tau, \mathbf{O}_t)$ learns to match the denoising vector field $\mathbf{u}(\mathbf{A}_t^\tau \mid \mathbf{A}_t) = \epsilon - \mathbf{A}_t$.

Multi-modal Fusion and Projection. We use lightweight linear projections to align representations across modules, including **1)** embedding robot states into the multimodal feature space, **2)** adapting action representations to the diffusion-based action expert, and **3)** matching feature dimensions between the VLM backbone and the action expert.

3.3 Continuous Inference Schedule

At time step t, the VLA model predicts an action sequence $\mathbf{A}_t = \{\mathbf{a}_t, \ldots, \mathbf{a}_{t+n}\}$. In standard chunk-based VLA execution [54, 13, 55], the next inference call is triggered only after the current chunk is fully consumed, causing inter-chunk waiting when objects continue to move.

To avoid this waiting, we use a greedy asynchronous schedule that launches the next inference call immediately after the previous one finishes. Let m denote the inference delay. Although this delay may vary across cycles in practice, we use a uniform m for notational simplicity, so new chunks become available at $t, t+m, t+2m, \ldots$. When a new chunk overlaps with actions remaining from an older chunk, the newer prediction replaces the older one at the same execution timestep.

3.4 Latency-aware Action Streaming

When inference starts at time t, the VLA model predicts $\mathbf{A}_t$ from observations $\mathbf{O}_t$. However, this chunk becomes available at $t+m$, by which time the object state has evolved from $\mathbf{s}_t$ to $\mathbf{s}_{t+m}$. Consequently, the prefix $\{\mathbf{a}_t, \ldots, \mathbf{a}_{t+m-1}\}$ is indexed to execution times that have elapsed before the chunk can be executed. Executing these actions at later physical timesteps would re-time commands predicted for past object states to the current dynamic scene, causing a perception–execution gap.

Latency-aware Action Streaming addresses this temporal-index validity problem by starting execution from $\mathbf{a}_{t+m}$, the first action whose intended execution time has not elapsed. It discards the latency-invalid prefix and streams only the valid suffix $\{\mathbf{a}_{t+m}, \ldots, \mathbf{a}_{t+n}\}$. This rule preserves action-time alignment by ensuring that each executed action corresponds to a timestep that remains valid when the chunk becomes available, which is critical when object states evolve during inference.

4 The Dynamic Object Manipulation Benchmark

4.1 Overview

Dynamic Object Manipulation (DOM) is a large-scale benchmark for dynamic object manipulation, addressing the lack of standardized evaluation on moving objects. It provides scalable, fully automated data collection in simulation and the real world, yielding 200K synthetic and 2K real-world episodes where teleoperation is impractical due to human reaction limits. DOM further structures scenarios by interaction, perception, and generalization factors, enabling consistent comparison across algorithms and robot embodiments.

4.2 Benchmark Dimensions

As illustrated in Figure 1c, DOM evaluates dynamic manipulation ability across three dimensions:

Interaction. This dimension evaluates how a policy responds to evolving object motion. *1) Closed-loop reactivity*, which measures how quickly the robot adjusts to objects moving at different speeds; *2) Dynamic adaptation*, where the policy must handle abrupt motion changes such as direction shifts or unexpected disturbances; *3) Long-horizon sequencing*, which assesses whether the policy maintains coherent behavior over extended interactions and prioritizes actions as motion events unfold.

Perception. This dimension evaluates how a policy perceives and grounds visual and linguistic cues in environments. *1) Visual understanding*, which measures the ability to distinguish objects with similar shapes, textures, or materials; *2) Spatial reasoning*, which examines whether the policy can infer object positions and relative arrangements in cluttered or changing scenes; *3) Motion perception*, which assesses how the policy interprets object motion cues such as speed and direction.

Generalization. This dimension evaluates how robustly a policy transfers across novel objects, scenes, and motion patterns. *1) Visual generalization*, which measures adaptation to unseen shapes,

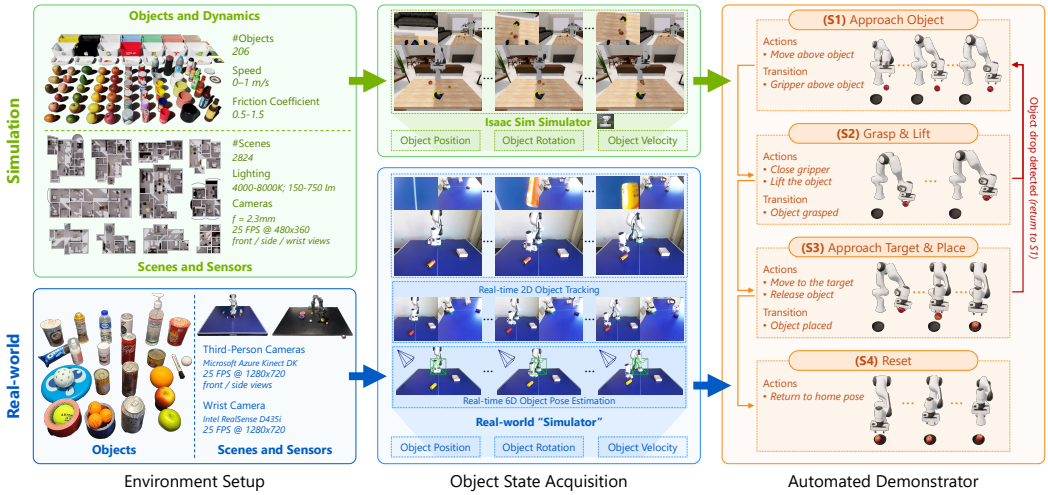


Figure 3: **Automatic Simulation and Real-world Data Collection.** *Environment Setup:* simulation and real-world settings share objects, scenes, and synchronized cameras. *Object State Acquisition:* simulation provides ground-truth 6D object states, while real-world RGB observations are converted into a real-world "simulator" interface that enables automatic dynamic-manipulation data collection without teleoperation.

appearances, and scene layouts; *2) Motion generalization*, which assesses whether the policy can handle new speed ranges, altered friction conditions, and trajectory patterns that differ from those observed during training; *3) Disturbance Robustness*, which tests the ability to maintain stable behavior under external perturbations such as unexpected pushes, collisions, or sensor noise.

4.3 Simulation Data Collection

Our simulation framework serves two purposes: 1) scaling dynamic manipulation data for VLA pretraining, and 2) providing a reproducible benchmark for fair evaluation. As shown in Figure 3, we build a high-throughput Isaac Sim pipeline [20] that combines scene and object sampling, multi-view RGB rendering, simulator-state acquisition, and automatic demonstration generation.

Objects and Dynamics. We use 206 everyday Objaverse objects [56], including fruits, vegetables, containers, and household items, with texture augmentation for visual diversity. Object speeds are sampled from 0–1 m/s (0.15–0.75 m/s for training, with some static objects; see Appendix D), and friction coefficients from 0.5–1.5. Multiple objects are placed in the workspace to allow natural interactions during motion.

Scenes and Sensors. We use 2.8K 3D-FRONT scenes [57], curated for clean, flat tabletops and realistic, non-self-occluding object placements. Each scene has three 25-FPS RGB cameras at 480×360 resolution: two third-person views 1 m from the robot and one wrist-mounted camera, with Azure-Kinect-aligned intrinsics. We randomize illumination with color temperature 4000–8000 K, intensity 150–750 lm, and light positions sampled from $x, y \in [-50, 50]$ m and $z \in [10, 20]$ m.

Object State Acquisition. The simulator maintains ground-truth 6D object states throughout each episode. Isaac Sim propagates object motion through the physics engine, from which we extract per-object position, rotation, and linear/angular velocity at 25 Hz. These states form a simulator-style interface for driving the automatic demonstrator during simulation data collection.

Automatic Demonstrator. The automatic demonstrator consumes real-time object pose and velocity, together with the target pose, to generate demonstrations through four stages: approach, grasp-and-lift, place, and reset. It predicts short-horizon object motion during approach, continuously updates the end-effector target, stabilizes residual motion during grasping, and returns the robot to the home pose after placement. This yields scalable dynamic manipulation trajectories without exposing privileged states, such as object poses and velocities, to learned policies.

4.4 Real-World Data Collection

Teleoperation is unreliable for dynamic manipulation because human reaction is too slow for fast-moving objects, while real-world setups lack the ground-truth 6D object states used by simulator pipelines. To address both issues, we build a real-world "simulator" that estimates simulator-style object states from commodity RGB sensors, enabling teleoperation-free dynamic data collection at $\approx$10 s per episode on both Franka and PiPER.

Environment Setup. We use 25 physical household objects spanning containers, food items, bottles, and tools, with multiple objects per episode, including pick/place targets and natural distractors. The scene is captured by two synchronized third-person RGB cameras (Azure Kinect DK) placed at front and side viewpoints, along with a wrist-mounted RealSense D435i, matching the simulation geometry and supplying synchronized, calibrated RGB streams for state estimation.

Object State Acquisition. To replicate the simulator's state interface, we build a real-world "simulator" that estimates object position and velocity. EfficientTAM [58] supplies per-view object masks from the synchronized third-person cameras, and a geometric triangulation step recovers the 3D centroid. Linear velocity is estimated by least-squares fitting over a short temporal window, and for curved motion the turning rate of the velocity direction is additionally estimated to extrapolate curved paths, producing a smooth, low-latency state stream that matches the simulator-style interface. Object orientation and angular velocity are not estimated in the real world: the demonstrator derives grasp orientation from the motion direction and per-object grasp priors, which is sufficient for the rotationally symmetric or top-down-graspable objects used in real-world collection.

Automatic Demonstrator. The same automatic demonstrator used in simulation runs unchanged in the real world, consuming estimated object states and target poses to generate trajectories. These estimated states are used only for data collection and are not exposed to learned policies.

5 Experiments

Our experiments evaluate DynamicVLA on dynamic object manipulation under real-time constraints. We benchmark DynamicVLA against VLA baselines across interaction, perception, and generalization challenges, and further analyze how model efficiency, asynchronous execution, and action-time alignment affect performance. Specifically, we study the following research questions:

1. How well can DynamicVLA interact with fast-moving objects and maintain stable closed-loop behavior over long horizons?
2. How well does DynamicVLA ground appearance, spatial, and motion cues?
3. How well does DynamicVLA generalize to unseen objects, scenes, and motion regimes?
4. How do model efficiency, non-blocking execution, and action-time-aligned streaming contribute to performance under inference latency?

5.1 Evaluation Protocols

Experimental Setup. DynamicVLA is evaluated on the Dynamic Object Manipulation (DOM) benchmark in simulation and the real world. Experiments are conducted in Isaac Sim with a Franka Emika Panda arm, and on real-world Franka and AgileX PiPER arms. In real-world trials, a secondary robot arm induces object motion using predefined launch trajectories, ensuring comparable dynamics across methods. The evaluated policy controls only the manipulation arm, with access limited to RGB observations, language instructions, and proprioception. Each real-world task is evaluated over three paired motion–position configurations, with 20 trials per configuration, resulting in 60 trials per task and 960 real-world trials per method. All methods are evaluated under identical conditions within each environment.

Baselines. In simulation, we evaluate ACT [16], Diffusion Policy [29], OpenVLA-OFT [54], $\pi_{0.5}$ [55], SmolVLA [14], GR00T-N1.5 [59], RTC [17], VLA-Adapter-Pro [15], and VLASH [10], covering general-purpose VLAs, lightweight adaptation-based models, and latency-aware designs. In real-world experiments, we evaluate $\pi_{0.5}$, SmolVLA, and VLASH under identical physical setups. All baselines are initialized from publicly available pretrained weights and adapted to DOM using the same training data, following official recipes when available.

Table 1: **Dynamic Object Manipulation Simulation Benchmark Results.** Average success rates (%) are reported across nine evaluation sub-dimensions, organized under three categories: Interaction, Perception, and Generalization. In addition, overall average success rate (Success, %) and task completion time (Time, seconds) are reported. Each method is evaluated over 1,800 trials (10 scenes $\times$ 9 dimensions $\times$ 20 trials). All baseline models are fine-tuned on the DOM dataset using their official implementations and released pretrained weights. Best results are highlighted in bold.

Methods	Interaction			Perception			Generalization			Average	
	CR	DA	LS	VU	SR	MP	VG	MG	DR	Success ↑	Time ↓
Diffusion Policy [29]	0.50	0.50	0.00	1.00	0.00	0.00	1.00	0.50	0.00	0.38	10.89
ACT [16]	2.50	1.50	0.00	1.00	1.50	0.50	2.50	2.50	0.00	1.33	10.80
OpenVLA-OFT [54]	3.50	0.50	0.50	0.00	1.50	0.50	3.50	2.00	0.00	1.33	10.83
$\pi_{0.5}$ [55]	9.50	17.50	3.50	5.00	12.50	9.00	5.00	19.50	18.00	11.06	10.62
RTC [17]	12.00	15.00	4.00	7.50	13.50	8.50	6.50	14.00	20.00	11.22	10.63
SmolVLA [14]	18.50	17.50	5.50	1.50	14.50	11.50	14.50	13.50	17.00	12.67	10.65
GR00T-N1.5 [59]	10.50	12.00	4.00	9.50	13.50	14.00	14.50	19.50	20.00	13.05	10.56
VLA-Adapter-Pro [15]	21.00	15.50	6.00	6.50	16.50	10.50	15.00	18.50	13.00	13.61	9.98
VLASH [10]	9.00	20.50	7.50	6.50	7.50	12.00	7.00	21.00	20.00	12.33	10.60
DynamicVLA	**60.50**	**38.50**	**40.50**	**51.50**	**48.00**	**33.50**	**59.50**	**65.00**	**26.50**	**47.06**	**8.53**

CR: Closed-loop Reactivity; DA: Dynamic Adaptation; LS: Long-horizon Sequencing; VU: Visual Understanding; SR: Spatial Reasoning; MP: Motion Perception; VG: Visual Generalization; MG: Motion Generalization; DR: Disturbance Robustness

Implementation Details. DynamicVLA is trained on 32 NVIDIA A100 GPUs with a batch size of 40 per GPU. We use the AdamW optimizer with a learning rate of 1×10^{-4}, β coefficients $(0.9, 0.95)$, $\epsilon = 1 \times 10^{-8}$, and weight decay of 1×10^{-10}. A cosine learning rate schedule with 1000 warm-up steps is employed. The models are trained for approximately two weeks, with three stages as 2 days for pre-training, 10 days for mid-training, and 2 days for post-training. DynamicVLA requires 1.8GB of GPU memory and performs asynchronous action-chunk inference at 0.226 second per chunk on an NVIDIA RTX A6000. Actions are indexed at the 25 Hz observation timestep with a fixed action horizon of $n = 20$, and the inference latency m is measured at runtime and allowed to vary across hardware and execution cycles (Sec. 5.5).

Evaluation Metrics. We evaluate all methods using two metrics: *Success Rate*, the fraction of trials that complete the instructed manipulation without object drop or timeout, and *Task Completion Time*, the elapsed time from the onset of object motion to task termination, including successful completion, timeout, or object drop. We report both metrics in simulation and only success rate in real-world experiments. All metrics are averaged over multiple trials.

Execution Constraints. To ensure safe real-world operation, we restrict the robot workspace within predefined bounds. If the predicted end-effector position exceeds a predefined safety threshold, the robot aborts the current attempt and returns to a safe home pose, and the trial is marked as failure.

5.2 Dynamic Interaction and Reactivity

For the *Interaction* dimension, the performance primarily depends on timely correction to evolving object motion. Across CR/DA/LS (Table 1), prior VLAs exhibit consistently low success under dynamic motion, while DynamicVLA reaches 60.5/38.5/40.5%. The gains are especially pronounced in long-horizon sequencing, where DynamicVLA improves over the strongest baseline by 33.0 percentage points. This suggests that the main difficulty is not reacting once to a moving object, but maintaining temporal coordination as the object state continues to evolve during execution.

The same pattern appears in real-world trials (Figure 4a). Baselines often commit to actions based on outdated observations or lose coordination after trajectory changes, whereas DynamicVLA more reliably maintains perception–execution alignment under tight timing constraints.

5.3 Multimodal Spatial-Temporal Reasoning

For the *Perception* dimension, visual-language grounding must be performed under object motion. In simulation (Table 1, VU/SR/MP), prior VLAs degrade substantially across all three subdimensions, while DynamicVLA improves success to 51.5/48.0/33.5%. Perception-heavy dynamic tasks expose a trade-off between semantic grounding and real-time execution: policies must resolve

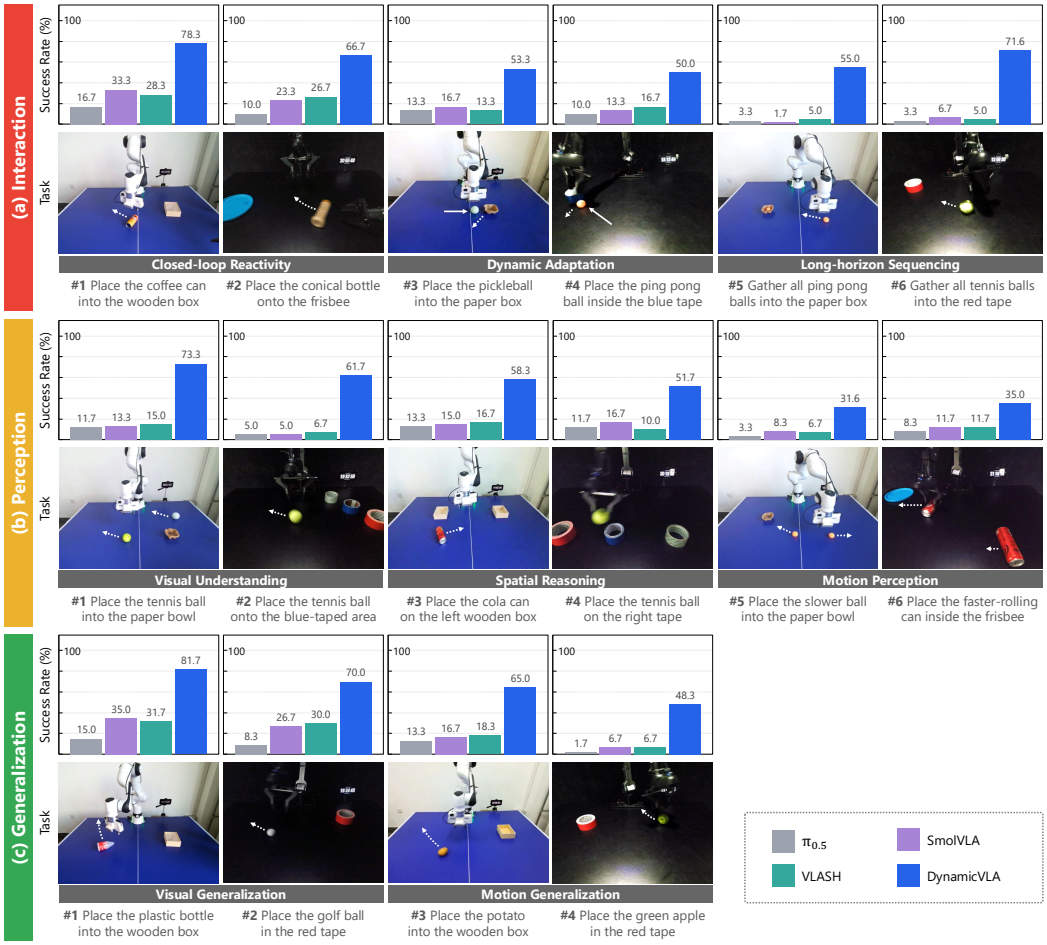


Figure 4: **Real-world Evaluation.** We compare VLA models on 16 real-world dynamic manipulation tasks using Franka and PiPER as the manipulation robot. Each method–task pair is evaluated over 60 trials, comprising 20 trials for each of three paired motion–position configurations. An additional robot arm initializes object motion, providing comparable motion conditions across trials.

appearance, spatial, and motion cues while acting before the scene evolves. Motion perception remains the most difficult case, suggesting that dynamic VLA policies still struggle when language grounding depends directly on object trajectories.

Real-world results show a similar trend (Figure 4b). DynamicVLA maintains a large advantage in visual and spatial grounding tasks, while motion-based instructions remain challenging, reflecting difficulty of estimating object motion, grounding language cues, and executing timely corrections.

5.4 Generalization under Distribution Shift

For the *Generalization* dimension, we evaluate whether policies remain effective under distribution shifts in object appearance, scene layout, motion patterns, and disturbances. In simulation (Table 1, VG/MG/DR), DynamicVLA achieves 59.5/65.0/26.5% success, consistently outperforming prior VLAs across all three settings. These results indicate that DynamicVLA does not merely fit the training configurations, but retains its advantage under unseen objects, scenes, and motion regimes.

Disturbance robustness remains substantially more challenging, with a lower success rate than visual and motion generalization. This suggests that abrupt perturbations and contact changes still pose failure cases for current dynamic VLA policies. Real-world results show similar trends for visual and motion shifts (Figure 4c). We omit real-world disturbance trials because perturbation strength and contact conditions are difficult to reproduce reliably across methods.

Table 2: **Ablation of key design choices.** Effects of FastViT vision encoding (FViT), Continuous Inference (CI), and Latency-aware Action Streaming (LAAS) on success rate (SR) on the DOM benchmark.

	FViT	CI	LAAS	SR (%) ↑
[1]	✓	✗	✗	30.27
[2]	✓	✗	✓	36.11
[3]	✓	✓	✗	39.72
[4]	✗	✓	✓	28.89
[5]	✓	✓	✓	**47.06**

Table 3: **Alternative overlap-handling strategies.** Naive async and Time Ensemble use the trained DynamicVLA policy with LAAS disabled, while RTC uses a separately trained inpainting policy. All methods are evaluated under the same observation, action, and evaluation protocol. We report jerk and success rate (SR) on the DOM benchmark.

Strategy	Jerk ↓	SR (%) ↑
Naive async	165.22	39.72
Time Ensemble [16]	160.69	39.72
RTC [17]	**155.36**	**39.83**

5.5 Ablation Studies

To analyze DynamicVLA's main design choices, we study visual encoding, continuous inference and overlap handling, LAAS, and the action horizon–latency relation. All variants are evaluated on DOM under identical protocols, with results summarized in Tables 2 and 3.

Vision Encoding. We ablate the visual encoder by replacing FastViT with a transformer-based encoder following the SmolVLM configuration [60], while keeping all other components fixed. Measured on a single NVIDIA RTX A6000 GPU, the FastViT-based variant reduces end-to-end per-chunk inference time from 0.523s to 0.226s and improves success from 28.89% to 47.06% (Table 2, [4] and [5]). This supports the use of efficient visual encoding for multi-view, multi-frame dynamic manipulation under tight latency constraints.

Continuous Inference and Overlap Handling. We evaluate the role of the continuous inference schedule by disabling it while keeping all other components unchanged (Table 2, [2] and [5]). CI uses naive asynchronous execution, launching each inference call immediately after the previous one finishes and selecting latest-chunk actions without smoothing. Disabling CI introduces inter-chunk waiting and reduces success from 47.06% to 36.11%. As shown in Table 3, Time Ensemble and RTC strategies slightly reduce jerk, measured as the time derivative of acceleration, but leave success nearly unchanged relative to naive async execution, remaining below the full LAAS variant in Table 2 [5]. This suggests that learned chunks are already temporally consistent enough for naive async execution, while overlap smoothing alone does not address latency-invalid actions under dynamic object motion.

Latency-aware Action Streaming. We evaluate LAAS as a mechanism for correcting the perception–execution mismatch caused by inference latency. As shown in Table 2, removing latency-invalid prefixes improves success even without Continuous Inference ([1] vs. [2]: 30.27% to 36.11%), indicating that actions indexed to already elapsed timesteps can hurt execution. Under Continuous Inference, disabling LAAS reduces success from 47.06% to 39.72% ([5] vs. [3]), showing that non-blocking execution alone does not remove this temporal misalignment. These results suggest that CI and LAAS address complementary latency effects: CI reduces inter-chunk waiting, while LAAS removes stale prefixes to preserve action-time alignment under dynamic object motion.

Action Horizon n and Inference Latency m. Actions are indexed at the 25 Hz observation timestep, so $n = 20$ spans 0.8 s and the 0.226 s per-chunk latency corresponds to $m \approx 6$ steps. LAAS uses the online-measured latency to discard a variable-length latency-invalid prefix from each completed chunk. Since $n \geq 2m$ on our hardware, the remaining suffix of length $n - m$ still covers the next inference delay, avoiding inter-chunk waiting. A sensitivity study in Appendix C (Table 7) injects J extra steps of delay into the same policy: success is unchanged while $n \geq 2(m + J)$ holds and degrades once the valid suffix no longer covers the next inference call. This also delimits the scope of CI and LAAS: their benefit requires a backbone whose latency satisfies the non-blocking condition, and each executed action still rests on observations m steps old, which is why integrating them into existing VLAs (Appendix C) yields a clear gain for SmolVLA but only a limited gain for the higher-latency $\pi_{0.5}$.

6 Conclusion and Future Work

This work shows that dynamic object manipulation exposes a key failure mode for VLA models: latency-induced temporal misalignment between observation and action execution, which is largely hidden in static manipulation. We present DynamicVLA, a latency-aware VLA model that combines efficient multimodal inference, non-blocking continuous inference, and Latency-aware Action Streaming to discard latency-invalid prefixes and preserve action-time alignment. We further introduce DOM, a large-scale benchmark built from scalable simulation and real-world data collection pipelines with automated demonstrators. Looking forward, several limitations of the current study point to promising directions for future work:

Toward Latency-scalable VLA Architectures. Dynamic manipulation exposes a fundamental tension between semantic understanding and real-time responsiveness. Future VLAs should improve inference speed without simply sacrificing visual-language reasoning, through adaptive token selection, compact temporal context, and efficient action generation.

Toward Longer-horizon Dynamic Tasks. DOM evaluates long-horizon sequencing through sustained interaction with moving objects. Future dynamic tasks can further extend to multi-stage settings where goals, object motion, and contact outcomes evolve over longer horizons, requiring memory and planning under real-time constraints.

Toward Richer Physical Dynamics. Moving rigid objects provide a tractable setting for studying perception–execution alignment. Future work can extend dynamic manipulation to deformable, articulated, granular, or fluid objects, where state prediction becomes more challenging.

Acknowledgments

We thank Prof. David Hsu (NUS) and Prof. Shengping Zhang (HIT) for their support and for providing access to the Franka Emika Panda robot utilized in this work. This work was supported by the Singapore Ministry of Education under its Academic Research Fund Tier 2 (MOE-T2EP20221-0012, MOE-T2EP20223-0002), and by cash and in-kind contributions from NTU S-Lab and industry partner(s).

References

[1] S. Bai, W. Song, J. Chen, Y. Ji, Z. Zhong, J. Yang, H. Zhao, W. Zhou, W. Zhao, Z. Li, P. Ding, C. Chi, H. Li, C. Xu, X. Zheng, D. Wang, S. Zhang, and B. Chen, "Towards a unified understanding of robot manipulation: A comprehensive survey," *arXiv 2510.10903*, 2025.

[2] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. A. Funkhouser, "TossingBot: learning to throw arbitrary objects with residual physics," *IEEE T-RO*, vol. 36, no. 4, pp. 1307–1319, 2020.

[3] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, and E. Osawa, "RoboCup: the robot world cup initiative," in *International Conference on Autonomous Agents*, 1997.

[4] J. Ren, Y. Li, K. Zhang, P. Fu, H. Jiang, Y. Pan, G. Zeng, T. Huang, W. Guo, P. Lu, T. Li, J. Wang, L. Chen, H. Li, and P. Luo, "SMASH: mastering scalable whole-body skills for humanoid ping-pong with egocentric vision," *arXiv 2604.01158*, 2026.

[5] P. Dürr, M. El Gheche, G. J. Maeda, N. Mukai, N. Takahashi, S. Heusser, H. Sahloul, Y. Saraiji, P. Adodin, and Y. e. a. Bi, "Outplaying elite table tennis players with an autonomous robot," *Nature*, vol. 652, no. 8111, pp. 886–891, 2026.

[6] Y. Zhang, R. Wang, and X. Chen, "Dynamic behavior cloning with temporal feature prediction: Enhancing robotic arm manipulation in moving object tasks," *IEEE RA-L*, vol. 10, no. 6, pp. 5209–5216, 2025.

[7] Z. Li, X. Wu, Z. Xu, and H. Zhao, "Train once, deploy anywhere: Realize data-efficient dynamic object manipulation," *arXiv 2508.14042*, 2025.

[8] S. Liu, B. Li, K. Ma, L. Wu, H. Tan, X. Ouyang, H. Su, and J. Zhu, "RDT2: exploring the scaling limit of umi data towards zero-shot cross-embodiment generalization," *arXiv 2602.03310*, 2026.

[9] Y. Ma, Y. Zhou, Y. Yang, T. Wang, and H. Fan, "Running VLAs at real-time speed," *arXiv 2510.26742*, 2025.

[10] J. Tang, Y. Sun, Y. Zhao, S. Yang, Y. Lin, Z. Zhang, J. Hou, Y. Lu, Z. Liu, and S. Han, "VLASH: real-time VLAs via future-state-aware asynchronous inference," *arXiv 2512.01031*, 2025.

[11] Y. Zhong, F. Bai, S. Cai, X. Huang, Z. Chen, X. Zhang, Y. Wang, S. Guo, T. Guan, K. N. Lui, Z. Qi, Y. Liang, Y. Chen, and Y. Yang, "A survey on Vision-Language-Action models: An action tokenization perspective," *arXiv 2507.01925*, 2025.

[12] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. P. Foster, P. R. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "OpenVLA: An open-source Vision-Language-Action model," in *CoRL*, 2025.

[13] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, and B. I. et al., "π_0: A Vision-Language-Action flow model for general robot control," in *RSS*, 2025.

[14] M. Shukor, D. Aubakirova, F. Capuano, P. Kooijmans, S. Palma, A. Zouitine, M. Aractingi, C. Pascal, M. Russi, A. Marafioti, S. Alibert, M. Cord, T. Wolf, and R. Cadène, "SmolVLA: A Vision-Language-Action model for affordable and efficient robotics," *arXiv 2506.01844*, 2025.

[15] Y. Wang, P. Ding, L. Li, C. Cui, Z. Ge, X. Tong, W. Song, H. Zhao, W. Zhao, P. Hou, S. Huang, Y. Tang, W. Wang, R. Zhang, J. Liu, and D. Wang, "VLA-Adapter: an effective paradigm for tiny-scale Vision-Language-Action model," in *AAAI*, 2026.

[16] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," in *RSS*, 2023.

[17] K. Black, M. Y. Galliker, and S. Levine, "Real-time execution of action chunking flow policies," in *NeurIPS*, 2025.

[18] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "LIBERO: benchmarking knowledge transfer for lifelong robot learning," in *NeurIPS*, 2023.

[19] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, and A. J. et al., "Open X-Embodiment: robotic learning datasets and RT-X models," in *ICRA*, 2024.

[20] M. Mittal, P. Roth, J. Tigue, A. Richard, O. Zhang, P. Du, A. Serrano-Muñoz, X. Yao, R. Zurbrügg, and N. R. et al., "Isaac Lab: A gpu-accelerated simulation framework for multi-modal robot learning," *arXiv 2511.04831*, 2025.

[21] OpenAI, "GPT-4 technical report," *arXiv 2303.08774*, 2023.

[22] Llama Team, "The llama 3 herd of models," *arXiv 2407.21783*, 2024.

[23] D. Guo, D. Yang, H. Zhang, J. Song, P. Wang, Q. Zhu, R. Xu, R. Zhang, S. Ma, and X. B. et al., "DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning," *Nature*, vol. 645, no. 8081, pp. 633–638, 2025.

[24] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," in *NeurIPS*, 2023.

[25] H. Liu, C. Li, Y. Li, and Y. J. Lee, "Improved baselines with visual instruction tuning," in *CVPR*, 2024.

[26] Qwen Team, "Qwen2.5-VL technical report," *arXiv 2502.13923*, 2025.

[27] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, and J. H. et al., "RT-1: robotics transformer for real-world control at scale," in *RSS*, 2023.

[28] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, and A. W. et al., "RT-2: Vision-Language-Action models transfer web knowledge to robotic control," in *CoRL*, 2023.

[29] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, "Diffusion policy: Visuomotor policy learning via action diffusion," in *RSS*, 2023.

[30] P. L. Moo Jin Kim, Chelsea Finn, "Fine-tuning Vision-Language-Action models: Optimizing speed and success," in *RSS*, 2025.

[31] D. Ghosh, H. R. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, Q. Vuong, T. Xiao, P. R. Sanketi, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," in *RSS*, 2024.

[32] S. Yang, Y. Du, S. K. S. Ghasemipour, J. Tompson, L. P. Kaelbling, D. Schuurmans, and P. Abbeel, "Learning interactive real-world simulators," in *ICLR*, 2024.

[33] H. Wu, Y. Jing, C. Cheang, G. Chen, J. Xu, X. Li, M. Liu, H. Li, and T. Kong, "Unleashing large-scale video generative pre-training for visual robot manipulation," in *ICLR*, 2024.

[34] Z. Ren, Y. Wei, X. Guo, Y. Zhao, B. Kang, J. Feng, and X. Jin, "VideoWorld: Exploring knowledge learning from unlabeled videos," in *CVPR*, 2025.

[35] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, S. Savarese, and L. Fei-Fei, "ROBOTURK: A crowdsourcing platform for robotic skill learning through imitation," in *CoRL*, 2018.

[36] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, A. Lee, K. Fang, C. Finn, and S. Levine, "BridgeData V2: A dataset for robot learning at scale," in *CoRL*, 2023.

[37] K. Wu, C. Hou, J. Liu, Z. Che, X. Ju, Z. Yang, M. Li, Y. Zhao, Z. Xu, and G. Y. et al., "RoboMIND: benchmark on multi-embodiment intelligence normative data for robot manipulation," in *RSS*, 2025.

[38] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE RA-L*, vol. 7, no. 3, pp. 7327–7334, 2022.

[39] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, W. Ai, and B. J. M. et al., "BEHAVIOR-1K: A human-centered, embodied AI benchmark with 1, 000 everyday activities and realistic simulation," *arXiv 2403.09227*, 2024.

[40] X. Li, K. Hsu, J. Gu, O. Mees, K. Pertsch, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao, "Evaluating real-world robot manipulation policies in simulation," in *CoRL*, 2024.

[41] Y. Mu, T. Chen, Z. Chen, S. Peng, Z. Lan, Z. Gao, Z. Liang, Q. Yu, Y. Zou, M. Xu, L. Lin, Z. Xie, M. Ding, and P. Luo, "RoboTwin: dual-arm robot benchmark with generative digital twins," in *CVPR*, 2025.

[42] S. Zhang, Z. Xu, P. Liu, X. Yu, Y. Li, Q. Gao, Z. Fei, Z. Yin, Z. Wu, Y. Jiang, and X. Qiu, "VLABench: A large-scale benchmark for language-conditioned robotics manipulation with long-horizon reasoning tasks," in *ICCV*, 2025.

[43] K. Zheng, X. Chen, O. C. Jenkins, and X. E. Wang, "VLMbench: A compositional benchmark for vision-and-language manipulation," in *NeurIPS*, 2022.

[44] Y. Jiang, A. Gupta, Z. Zhang, G. Wang, Y. Dou, Y. Chen, L. Fei-Fei, A. Anandkumar, Y. Zhu, and L. Fan, "VIMA: general robot manipulation with multimodal prompts," *arXiv 2210.03094*, 2022.

[45] J. Xu, S. Kim, T. Chen, A. R. Garcia, P. Agrawal, W. Matusik, and S. Sueda, "Efficient tactile simulation with differentiability for robotic manipulation," in *CoRL*, 2022.

[46] I. Akinola, J. Xu, J. Carius, D. Fox, and Y. Narang, "TacSL: A library for visuotactile sensor simulation and learning," *IEEE T-RO*, vol. 41, pp. 2645–2661, 2025.

[47] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu, "RoboCasa: Large-scale simulation of everyday tasks for generalist robots," in *RSS*, 2024.

[48] Y. Wang, Z. Xian, F. Chen, T. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan, "RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation," in *ICML*, 2024.

[49] H. Xie, B. Wen, Z. Chen, F. Hong, and Z. Liu, "3D scene generation: A survey," *IJCV*, vol. 134, p. 418, 2026.

[50] L. B. Allal, A. Lozhkov, E. Bakouch, G. M. Blázquez, G. Penedo, L. Tunstall, A. Marafioti, H. Kydlícek, A. P. Lajarín, V. Srivastav, J. Lochner, C. Fahlgren, X. Nguyen, C. Fourrier, B. Burtenshaw, H. Larcher, H. Zhao, C. Zakka, M. Morlon, C. Raffel, L. von Werra, and T. Wolf, "SmolLM2: when smol goes big - data-centric training of a small language model," *arXiv 2502.02737*, 2025.

[51] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, "FastViT: A fast hybrid vision transformer using structural reparameterization," in *ICCV*, 2023.

[52] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *ICLR*, 2023.

[53] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, D. Podell, T. Dockhorn, Z. English, and R. Rombach, "Scaling rectified flow transformers for high-resolution image synthesis," in *ICML*, 2024.

[54] M. J. Kim, C. Finn, and P. Liang, "Fine-tuning Vision-Language-Action models: Optimizing speed and success," in *RSS*, 2025.

[55] P. Intelligence, "$\pi_{0.5}$: a Vision-Language-Action model with open-world generalization," in *CoRL*, 2025.

[56] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi, "Objaverse: A universe of annotated 3D objects," in *CVPR*, 2023.

[57] H. Fu, B. Cai, L. Gao, L. Zhang, J. Wang, C. Li, Q. Zeng, C. Sun, R. Jia, B. Zhao, and H. Zhang, "3D-FRONT: 3D furnished rooms with layouts and semantics," in *ICCV*, 2021.

[58] Y. Xiong, C. Zhou, X. Xiang, L. Wu, C. Zhu, Z. Liu, S. Suri, B. Varadarajan, R. Akula, F. N. Iandola, R. Krishnamoorthi, B. Soran, and V. Chandra, "Efficient track anything," *arXiv 2411.18933*, 2024.

[59] J. Bjorck, F. Castañeda, N. Cherniadev, X. Da, R. Ding, Linxi, Y. Fang, D. Fox, F. Hu, and S. H. et al., "GR00T N1: an open foundation model for generalist humanoid robots," *arXiv 2503.14734*, 2025.

[60] A. Marafioti, O. Zohar, M. Farré, M. Noyan, E. Bakouch, P. Cuenca, C. Zakka, L. B. Allal, A. Lozhkov, N. Tazi, V. Srivastav, J. Lochner, H. Larcher, M. Morlon, L. Tunstall, L. von Werra, and T. Wolf, "SmolVLM: redefining small and efficient multimodal models," *arXiv 2504.05299*, 2025.

[61] M. Byeon, B. Park, H. Kim, S. Lee, W. Baek, and S. Kim, "COYO-700M: image-text pair dataset," https://github.com/kakaobrain/coyo-dataset, 2022.

A Model Architecture Details

VLM Backbone. The RGB images in the temporal observation window $\mathbf{O}_t$ are concatenated and encoded by FastViT [51] using a hierarchical multi-stage design. Each input image is resized to 384×384, and the encoder progressively increases channel width across stages $(96, 192, 384, 768, 1536)$ with corresponding block depths $(2, 12, 24, 4, 2)$. FastViT applies aggressive spatial compression via a large initial patch size of 64 and strided downsampling, using RepMixer-style token mixing in early stages and attention in later ones. The encoder outputs 36 visual tokens of fixed dimension 960, aligned with the language embedding space, achieving substantial token reduction while preserving manipulation-relevant spatial structure. In addition to visual inputs, the robot proprioceptive state $\mathbf{P}_t$ is incorporated as an explicit conditioning signal. The 32-dimensional state vector, containing Cartesian position and orientation with zero padding for unused entries, is linearly projected into the language embedding space and represented as a single 960-dimensional state token. Language instructions $\mathbf{L}_t$ are tokenized into a variable number of language tokens depending on prompt length. All visual, language, and state tokens are concatenated and processed jointly by the language backbone. Multimodal reasoning is performed by SmolLM2-360M [60], where only the first 16 transformer layers are used to reduce inference latency, following the practice adopted in SmolVLA [14]. The backbone outputs key–value representations for all processed tokens, which are cached and reused across inference cycles.

Action Expert. Action generation is handled by a dedicated diffusion-based action expert, instantiated as a lightweight transformer copied from the language backbone and truncated to the first 16 layers. The expert predicts an action chunk with horizon $n = 20$, which is sufficient under Continuous Inference while keeping inference latency low. Each action is a 32-dimensional vector representing end-effector pose and gripper state (with zero padding), and the noisy action input has shape $(n, 32)$ during training and pure noise during inference. The action expert uses a reduced hidden dimension of 720 ($0.75 \times$ the language embedding size) to lower computation. Noisy action tokens are projected into this space and combined with diffusion timestep embeddings, and denoising updates are generated by querying the cached key–value representations, without re-encoding perceptual inputs.

B The Training Scheme

Pre-training Stage. The vision–language backbone combines a convolutional visual encoder (FastViT) and a compact language model (SmolLM2-360M), both initialized from their respective pretrained weights. To align visual and linguistic representations, we first perform large-scale vision–language pre-training using 150M English image–text pairs sampled from COYO-700M [61].

Mid-training Stage. After vision–language pre-training, the full VLA model is trained on the synthetic Dynamic Object Manipulation (DOM) dataset (Sec. 4). Each episode provides temporally evolving multi-view visual observations, from which the model uses a wrist-mounted camera on the end-effector and a fixed third-person camera facing the manipulator. To capture short-term dynamics, the temporal observation window is instantiated as $\mathbf{O}_t = \{\mathbf{o}_{t-2}, \mathbf{o}_t\}$. Using two views per timestep, this results in four images per input step, which are concatenated channel-wise and processed jointly by the vision encoder. In this stage, DynamicVLA is optimized using minibatches formed by randomly sampling episode timesteps from shuffled manipulation demonstrations. For each minibatch, the model is trained on tuples $(\mathbf{O}_t, \mathbf{L}_t, \mathbf{P}_t)$, while the action expert is trained to denoise a noisy action chunk $\mathbf{A}_t^\tau$ under the objective defined in Eq. 1.

Post-training stage. In the post-training stage, the model is fine-tuned on robot-specific real-world demonstrations using the same objective as in mid-training, enabling adaptation to new embodiments and sensing configurations.

C More Discussion

Temporal Visual Context. We conduct an ablation study to analyze the impact of temporal visual context by varying the composition of the observation window $\mathbf{O}_t$ within the same DynamicVLA architecture. As described in Sec. 3, our default setting feeds the model $\mathcal{M}$ with a sparse temporal window $\mathbf{O}_t = \{\mathbf{o}_{t-2}, \mathbf{o}_t\}$, which is designed to facilitate implicit object velocity perception. As

Table 4: **Ablation on Temporal Visual Context.** The temporal observation window is varied by enabling different visual frames at time steps $\{t-3, t-2, t-1, t\}$, while keeping the model architecture, inference frequency, and execution pipeline fixed. Note that SR, T.Time, and I.Time represent the success rate (in %), task completion time (in seconds), and inference time (in seconds, measured on an NVIDIA RTX A6000 GPU), respectively.

$t-3$	$t-2$	$t-1$	t	SR ↑	T.Time ↓	I.Time ↓
✗	✗	✗	✓	38.22	9.52	**0.225**
✗	✗	✓	✓	43.39	8.77	0.226
✗	✓	✗	✓	47.06	8.53	0.226
✓	✗	✗	✓	46.89	8.51	0.226
✗	✓	✓	✓	**47.11**	**8.46**	0.228
✓	✓	✓	✓	47.06	8.53	0.229

Table 5: **Ablation on LLM Depth.** Different LLM depths are evaluated by retaining the first l transformer layers. Note that SR, T.Time, I.Time, and #Param denote success rate (%), task completion time (seconds), inference time (seconds, measured on an NVIDIA RTX A6000 GPU), and parameter count (in millions), respectively.

#Layers	SR ↑	T.Time ↓	I.Time ↓	#Param ↓
8	44.17	8.92	**0.127**	**303**
16	47.06	8.53	0.226	430
24	**48.44**	8.43	0.317	558
32	42.11	**8.39**	0.373	685

summarized in Table 4, different temporal configurations lead to negligible differences in inference latency and parameter count. We observe that using a single-frame input $\{\mathbf{o}_t\}$ results in a clear drop in task success rate, as a single observation lacks the temporal cues necessary for estimating object motion and dynamics. However, expanding the temporal window beyond two frames does not yield further noticeable gains, indicating diminishing returns from additional visual redundancy. Moreover, compared to $\{\mathbf{o}_{t-2}, \mathbf{o}_t\}$, the setting $\{\mathbf{o}_{t-1}, \mathbf{o}_t\}$ achieves lower success rates, suggesting that a larger temporal interval provides more informative motion cues for velocity estimation. Overall, these results demonstrate that sparse but sufficiently spaced temporal context is critical for effective dynamic manipulation, even without increasing inference frequency.

Depth of LLM Backbone. Following the backbone truncation strategy [14], we reduce inference latency by retaining only the first l transformer layers of the LLM during inference. To examine whether this design choice remains effective in the DynamicVLA setting, we evaluate multiple backbone depths ($l = 8, 16, 24$) and compare them against the full model ($l = 32$). As shown in Table 5, increasing the backbone depth leads to a modest increase in inference latency. However, this additional latency can be largely amortized by Continuous Inference and Latency-aware Action Streaming, and does not translate into a noticeable improvement in task success rate. In contrast, aggressively truncating the backbone significantly improves inference speed, but at the cost of reduced model capacity, resulting in a substantial degradation in success rate. Overall, this ablation confirms that a 16-layer backbone strikes the optimal balance between efficiency and robustness.

Cross-Model Analysis of CI and LAAS To evaluate the generality of the proposed execution mechanisms, Continuous Inference (CI) and Latency-aware Action Streaming (LAAS) are integrated into existing VLA models, including SmolVLA and $\pi_{0.5}$, without altering their backbone architectures. As shown in Table 6, consistent performance improvements are observed on SmolVLA, indicating that CI and LAAS effectively enhance closed-loop responsiveness under moderate inference latency. In contrast, $\pi_{0.5}$ exhibits only marginal gains, as its substantially larger backbone incurs high inference latency, which limits the effectiveness of overlapping inference and temporally aligned execution. These results suggest that CI and LAAS are broadly applicable, while their practical benefit is maximized when paired with a low-latency backbone.

Sensitivity to Additional Inference Latency. Although the main formulation uses a constant m for notation, our implementation measures latency online for each inference call. When a chunk

Table 6: **Cross-Model Analysis of CI and LAAS.** CI and LAAS are integrated into existing VLA models without backbone modification or retraining. Note that SR and T.Time represent the success rate (in %) and task completion time (in seconds), respectively. † indicates inference-time integration of CI and LAAS.

Method	SR (%) ↑	T.Time (s) ↓
$\pi_{0.5}$† [55]	15.89	9.95
SmolVLA† [14]	25.56	9.77
DynamicVLA	**47.06**	**8.53**

Table 7: **Robustness to increased inference latency.** J denotes the number of extra control steps inserted after each inference call before the predicted action chunk is released. Success rate (SR) and task completion time (T.Time) are reported on the DOM benchmark.

Delay J	SR (%) ↑	T.Time ↓
2	**47.06**	**8.53**
4	**47.06**	**8.53**
6	44.44	8.79
8	41.94	8.97

launched at time t becomes available after $\hat{m}_t$ control steps, LAAS discards the first $\hat{m}_t$ actions and streams the remaining valid suffix. To test robustness to increased inference latency, we inject a fixed extra delay after each inference call. Specifically, before releasing each predicted chunk, we wait for an additional J control steps, so the effective latency becomes $\hat{m}_t = m_t + J$. We evaluate $J \in \{2, 4, 6, 8\}$ under the same DOM protocol. On a single NVIDIA RTX A6000 GPU, the nominal latency $m_t \approx 6$ control steps. With $n = 20$, the executable suffix remains long enough to bridge the next inference call up to approximately $J = 4$ to satisfy $n \geq 2\hat{m}_t$. Once the effective latency grows further, the suffix length $n - m_t$ becomes insufficient to cover the next inference interval, leading to inter-chunk waiting and lower success rates. As shown in Table 7, performance remains stable under moderate additional delay and degrades only when the non-blocking condition is violated.

D Detailed Evaluation Setup

In this section, we provide comprehensive details of the simulation data split and the real-world evaluation setups used in our experiments, including task specifications and object configurations.

Simulation Data Split and Evaluation Protocol. The training pool consists of 201 of the 206 objects and 2,821 of the 2,824 scenes, with object speeds from 0.15 to 0.75 m/s plus static objects and friction coefficients from 0.5 to 1.5. Language instructions are generated from attribute tags (appearance, size, spatial position, distance, and speed) ranked by per-episode object states, and test instructions use held-out object–target combinations. *Visual Generalization* evaluates on 5 held-out objects and 3 held-out scenes that are disjoint from the training pool by asset ID. *Motion Generalization* evaluates on out-of-range speeds (0.75 to 1.0 m/s) and out-of-range friction coefficients (0 to 0.5 and 1.5 to 2.0). *Disturbance Robustness* applies scripted pushes and collisions at evaluation time only. The remaining six sub-dimensions use training-distribution objects and scenes with unseen per-trial initial states. Each sub-dimension uses 10 test environments with 20 trials each. An environment specifies a scene asset, its objects, and pre-generated initial object poses, velocities, and friction values that are reused identically across all methods, and disturbance events are scripted per trial. No object or scene asset ID appears in both a training set and its held-out evaluation set; the exact ID lists and per-trial specifications ship with the benchmark release.

Real-world Object Launching. Each real-world task is executed under standardized conditions to ensure repeatability and fair comparison across different policies. Objects are launched by a secondary robot arm that executes a position-controlled push-and-release primitive with a trapezoidal velocity profile (maximum acceleration 2.0 m/s^2), releasing the object 2 cm above the tabletop. Release speeds range from 0.2 to 0.8 m/s and are verified post hoc by the tracking system. Evaluation is conducted across three predefined paired motion–position configurations, each combining an initial motion profile with a corresponding target container position; the three configurations use release speeds of 0.3, 0.5, and 0.7 m/s. Trajectory families cover straight rolls, curved rolls (rotationally asymmetric objects such as the conical bottle), and collision-deflected trajectories. Across repeated launches of the same configuration, the standard deviation is 0.03 m/s in release speed and 3.5 cm in trajectory endpoint, confirming comparable motion conditions across methods.

Real-world Interaction Evaluation (Sec. 5.2)

- **Place the coffee can into the wooden box.** The robot must track and grasp a rolling Nescafé coffee can and place it into a wooden box. This task evaluates closed-loop reactivity to continuously moving targets.
- **Place the conical bottle onto the frisbee.** The robot must grasp a conical roasted sesame bottle whose rolling motion follows a curved trajectory and place it onto a blue frisbee. This task evaluates closed-loop reactivity under non-linear object motion.
- **Place the pickleball into the paper box.** The robot must grasp a moving pickleball and place it into a paper box, where the ball is designed to collide with the box and undergo trajectory deflection. This task evaluates adaptive manipulation under contact-induced motion changes.
- **Place the ping pong ball inside the blue tape.** The robot must grasp a moving ping pong ball and place it within a blue-taped region, where impacts with the tape are designed to deflect the ball's trajectory. This task evaluates adaptive placement under perturbed object motion.
- **Gather all ping pong balls into the paper box.** The robot must continuously collect ping pong balls that repeatedly appear on the tabletop and place them into a paper box. This task evaluates long-horizon task sequencing under sustained dynamic inputs.
- **Gather all tennis balls into the red tape.** The robot must continuously collect tennis balls that repeatedly appear on the tabletop and return them to a red-taped region. This task evaluates long-horizon planning and execution in dynamic environments.

Real-world Perception Evaluation (Sec. 5.3)

- **Place the tennis ball into the paper bowl.** The robot must identify and grasp the moving tennis ball among multiple simultaneously thrown objects (a tennis ball and a pickleball), and place it into a paper bowl. This task evaluates object-level visual understanding for identifying and manipulating the correct target under dynamic motion.
- **Place the tennis ball onto the blue-taped area.** The robot is required to catch a rolling tennis ball and place it precisely within the region marked by blue tape, among multiple visually similar tape markings (red, blue, and transparent). This task evaluates visually grounded target understanding and precise placement under continuous object motion.
- **Place the cola can on the left wooden box.** The robot must grasp a moving cola can and place it on a wooden box located to its left, evaluating its ability to handle spatial placement under varying object motions. This task evaluates spatial understanding for target localization and placement relative to the robot's viewpoint.
- **Place the tennis ball on the right tape.** The robot must grasp a moving tennis ball and place it on a tape located to its right, evaluating spatial awareness and placement precision. This task evaluates spatial understanding for interpreting directionally specified targets and executing accurate placement.
- **Place the slower ball into the paper bowl.** The robot must grasp the ping pong ball specified by its lower moving speed and place it into the paper bowl. This task evaluates motion-based target understanding, where the target is specified by its movement direction.
- **Place the faster-rolling can inside the frisbee.** The robot must grasp the cola can specified by its higher rolling speed and place it inside the blue frisbee. This task evaluates motion-based target understanding, where the target is specified by its relative motion speed.

Real-world Generalization Evaluation (Sec. 5.4)

- **Place the plastic bottle into the wooden box.** The robot must grasp a rolling plastic bottle with an unseen appearance and a regular curved trajectory, and place it into a wooden box. This task evaluates visual generalization to unseen object appearances under dynamic motion.
- **Place the golf ball in the red tape.** The robot must grasp a rolling golf ball with an unseen appearance and place it within a red-taped region. This task evaluates visual generalization to unseen object instances during dynamic manipulation.

- **Place the potato into the wooden box.** The robot must grasp a moving potato whose motion follows irregular patterns and place it into a wooden box. This task evaluates motion generalization to irregular object dynamics.
- **Place the green apple in the red tape.** The robot must grasp a moving green apple whose motion exhibits irregular and unpredictable patterns, and place it onto a red-taped region. This task evaluates motion generalization to irregular object trajectories.

E Broader Impacts

DynamicVLA may support research on more reliable robotic manipulation in settings with moving objects, such as logistics, manufacturing, and other structured service scenarios. By identifying latency-induced perception–execution mismatch and introducing DOM, this work may also help the community evaluate VLA policies under more realistic timing constraints. As with other physical robot learning systems, downstream deployment requires careful safety validation, especially in unstructured environments or around humans. Our experiments are conducted in controlled simulation and laboratory settings with bounded workspaces and abort conditions; the model is intended as a research prototype rather than an unsupervised system for safety-critical deployment.

F Existing Assets and Licenses

We use several existing models, datasets, simulators, and perception tools. All external assets are used under their respective licenses or terms of use, and we cite original creators in the main paper.

- **Pretrained models and tools.** DynamicVLA uses SmolLM2 as the language backbone, FastViT as the visual encoder, and EfficientTAM for real-world object mask tracking. SmolLM2 and EfficientTAM are released under Apache-2.0, while FastViT is used under Apple's ML-FastViT license terms.
- **Pretraining and simulation data.** We use COYO-700M for vision-language pretraining, Objaverse objects for simulation assets, and 3D-FRONT scenes for simulated environments. COYO-700M is distributed under CC-BY-4.0, while its underlying image/text content remains subject to the licenses of the original content providers; Objaverse is distributed under ODC-By v1.0, with individual objects under their corresponding Creative Commons licenses; 3D-FRONT is used under its dataset terms of use.
- **Simulation platform.** We use Isaac Sim for physics simulation and rendering. The Isaac Sim source code is released under Apache-2.0, while additional components such as Omniverse Kit SDK, models, and textures are governed by separate NVIDIA license terms.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions: identifying latency-induced perception–execution mismatch, proposing DynamicVLA with continuous inference and LAAS, introducing DOM, and evaluating on simulation and real robots. These claims are directly supported by the method, benchmark, and experimental sections.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper discusses limitations and future directions, including the latency–understanding trade-off, limited focus on short- to medium-horizon dynamics, and the current assumption of rigid-body dynamics. These points define the scope of the proposed method and indicate important cases not yet addressed.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: The paper does not present formal theoretical results, theorems, or proof-based claims. The mathematical formulation is used to define the dynamic manipulation setting and proposed execution mechanism rather than to establish theoretical guarantees.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper provides the model architecture, data-collection pipeline, benchmark dimensions, evaluation protocols, baselines, metrics, and implementation details needed to understand and reproduce the main experiments. The experimental setup specifies the simulation and real-robot platforms, trial counts, input modalities, baseline adaptation, and evaluation metrics.

   Guidelines:

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

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: The code and data are open-sourced on GitHub at https://github.com/hzxie/DynamicVLA, with instructions to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the training stages, input construction, optimizer, learning rate, batch size, compute setup, inference configuration, action horizon, and evaluation protocol. The appendix further reports temporal-context and model-depth ablations, supporting the interpretation of the main results.

   Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: We report success rates and task completion times over fixed simulation and real-world trial sets, but do not include error bars or formal significance tests. Estimating variance over independent evaluation replications would require repeatedly resampling scenes, motion conditions, object states, and contact outcomes, which is beyond the scope of the current benchmark evaluation.

Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Section 5.1 reports the compute resources used for training and inference, including GPU type and count, per-GPU batch size, training time, inference GPU, memory usage, action timestep rate, and action-chunk inference latency. Baseline and ablation experiments follow the same compute setup unless otherwise specified.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

Justification: The research is conducted on simulated and controlled real-robot manipulation tasks and includes safety constraints for real-world execution. In particular, the robot workspace is bounded, unsafe predicted end-effector positions trigger abort and return-to-home behavior, and failed trials are explicitly counted as failures.

Guidelines:

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

10. **Broader impacts**

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

Answer: [Yes]

Justification: The paper discusses broader impacts in the appendix, including potential benefits for dynamic robotic manipulation and risks associated with deploying VLA policies on physical robots. It also clarifies that the experiments are conducted in controlled settings and that deployment in human-populated or safety-critical environments would require additional safety validation and safeguards.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release high-risk generative models, scraped web-scale datasets, or dual-use models of the kind highlighted in the checklist. For physical robot experiments, the paper nevertheless describes operational safeguards such as workspace limits and abort-and-return behavior during unsafe predictions.

Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the creators of all existing assets used in this work, including pretrained models, datasets, simulators, and tracking tools. Their licenses or terms of use are summarized in the appendix and release documentation.

    Guidelines:

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

13. **New assets**

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

    Answer: [Yes]

    Justification: The paper introduces the DOM benchmark, including synthetic and real-world episodes, simulation-ready scenes, object sets, benchmark dimensions, and evaluation tasks. The data-collection pipeline, object dynamics, sensors, task categories, and real-world evaluation tasks are documented in the main paper and appendix.

    Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human-subject experiments. Data are collected through simulation and automated robot data-collection pipelines rather than through human participants.

Guidelines:

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

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

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

Answer: [N/A]

Justification: The work does not involve human subjects, crowdsourcing workers, or participant studies, so IRB approval is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: The paper describes the use of SmolLM2-360M as an existing pretrained language backbone in DynamicVLA, using its first 16 transformer layers for efficient multimodal reasoning. The LLM is not newly developed in this work, but it is a core component of the proposed VLA architecture and is therefore declared.

Guidelines:

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