MEF-VLA: MOTION-TO-EVENT FORESIGHT FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language-action (VLA) models have advanced robot manipulation, and predictive world modeling further supports action learning through future-state prediction. Despite these advances, policies can still deviate from the instructed sequence or fail to complete individual interactions in multi-stage tasks. To improve the guidance provided by visual foresight, we revisit which future to predict and what to represent. A fixed prediction horizon can end during intermediate motion or extend beyond the next interaction, resulting in inconsistent guidance for task progress. Even with a reference aligned with task progress, complete future-frame targets retain static scene appearance while leaving the spatial change from the current state implicit. We introduce MEF-VLA (Motion-to-Event Foresight), which learns the motion remaining until the next interaction event. Event frames marking upcoming interactions define adaptive prediction horizons. Remaining event-flow represents the motion toward these shared references as state-dependent spatial displacement. Learned event queries predict remaining-motion features and condition the action expert, adding negligible inference overhead. In simulation, MEF-VLA improves success rates over the vanilla VLA by 1.6 percentage points on LIBERO, 8.9 on zero-shot LIBERO-Plus, and 9.0 on five-task sequences in CALVIN D→D. Across three real-world task families, MEF-VLA trained on one- and two-stage instructions achieves 58.9% average success on held-out three-stage instructions, compared with 34.4% for the vanilla VLA. Ablations further demonstrate the benefits of both event-aligned horizons and remaining-motion supervision for task success and zero-shot robustness.

1 INTRODUCTION

Vision-language-action (VLA) models have advanced language-conditioned robot manipulation by leveraging pretrained visual and linguistic representations (Kim et al., 2025b; Black et al., 2025a). Predictive world modeling further supports action learning by predicting future observations or their representations (Tian et al., 2025; Zhang et al., 2025; Yang et al., 2026; Sun et al., 2026; Lin et al., 2026b). Despite these advances, policies can still deviate from the instructed sequence or fail to complete individual interactions in multi-stage manipulation. To improve the guidance provided by visual foresight, we revisit two questions (Fig. 1): *which future to predict, and what to represent?*

Many predictive policies define future targets using fixed frame offsets or temporal windows. These targets range from future frames in Seer (Tian et al., 2025) and Mantis (Yang et al., 2026) to point trajectories in FoMoVLA (Li et al., 2026) and a scene-flow prior in LaMP (Wang et al., 2026). Because interaction timing varies across executions, a fixed window can capture only part of the motion preceding an interaction or extend into the following operation. The temporal reference therefore determines which future state is targeted and which motion is supervised (Fig. 1, left).

Recent methods organize future references around task progress. StructVLA (Jin et al., 2026) predicts sparse structured frames as progress anchors, while StageWAM (Liu et al., 2026a) predicts a latent representation of the next inferred stage. Observations approaching the same grasp, for example, can share a future reference while differing in the displacement needed to reach it. Supervising the reference state leaves this displacement implicit. Targets derived from complete future frames also retain static scene appearance, which can divert model capacity from learning the motion needed to complete the next interaction.

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

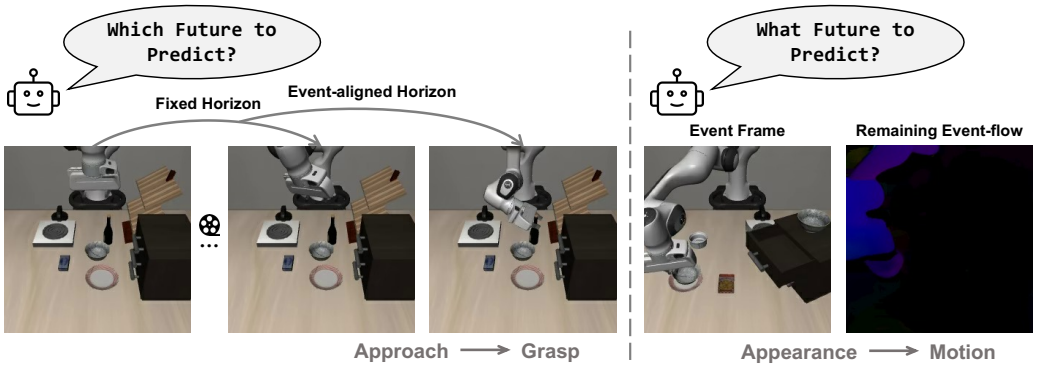

Figure 1: **Motion-to-event foresight for multi-stage manipulation.** Left: fixed-horizon prediction can terminate during intermediate motion, whereas event alignment selects the next interaction as a trajectory-adaptive reference. Right: a complete event frame retains scene appearance; remaining event-flow instead represents the state-dependent displacement from the current observation to that event. MEF-VLA uses this shared reference to define the temporal endpoint and motion target.

We introduce MEF-VLA (Motion-to-Event Foresight), a VLA framework that learns the motion remaining from the current observation to the next interaction event. Event frames provide a shared reference for observations approaching the same interaction, allowing the prediction horizon to adapt to execution. Remaining event-flow encodes the direction and extent of displacement from each current state to this reference, focusing predictive supervision on task-relevant motion rather than static appearance. Learned event queries predict features of this motion representation and condition the action expert. All other predictive components are used only during training, resulting in negligible inference overhead.

Motion-to-event foresight improves task success and zero-shot robustness: compared with a vanilla VLA using the same backbone and action expert, MEF-VLA improves success by 1.6 percentage points on LIBERO (Liu et al., 2023) and 8.9 points on LIBERO-Plus (Fei et al., 2026). Ablations on both benchmarks show that event alignment and remaining-motion supervision each contribute to these gains. The benefits extend to sequential instruction following on CALVIN D$\rightarrow$D (Mees et al., 2022), where five-task success increases from 58.1% to 67.1%. Real-world experiments further test generalization to longer instruction compositions across Pill Sorting, Sandwich Making, and Cup Stacking. Despite training only on one- and two-stage instructions, MEF-VLA raises average success on held-out three-stage instructions from 34.4% to 58.9%.

In summary, our contributions are as follows:

(i) We formulate motion-to-event foresight for VLA policies, using the next interaction as a shared, trajectory-adaptive reference for predicting the motion remaining from the current state.
(ii) We introduce remaining event-flow to represent this motion as state-dependent spatial displacement, focusing predictive supervision on execution-relevant changes. Learned event queries encode these motion features to condition action generation.
(iii) Experiments on LIBERO, LIBERO-Plus, and CALVIN demonstrate improved task success, zero-shot robustness, and sequential instruction following, with ablations establishing the benefits of event alignment and remaining-motion supervision. Real-world evaluations further demonstrate generalization from one- and two-stage training instructions to held-out three-stage instructions across three task families.

2 Related Work

2.1 Vision-Language-Action Models

VLA models adapt pretrained vision-language models (VLMs) to language-conditioned robot control. RT-2 (Zitkovich et al., 2023) and OpenVLA (Kim et al., 2025b) formulate action prediction as

autoregressive generation of discrete tokens, connecting vision-language pretraining with learning from robot demonstrations. For continuous action generation, π_0 (Black et al., 2025a) combines a VLM backbone with a flow-matching action expert, and GR00T N1 (NVIDIA et al., 2025) conditions a diffusion transformer on vision-language features to generate action chunks. Building on this paradigm, MEF-VLA augments action learning with motion prediction.

2.2 Visual Foresight for Robot Manipulation

Visual foresight introduces future observations or their representations as additional prediction targets for action learning. Seer (Tian et al., 2025) jointly predicts future images and actions, Mantis (Yang et al., 2026) couples future-frame reconstruction with action learning through learned queries, and SSM-VLA (Cai et al., 2026) cascades visual prediction and action generation. Beyond RGB prediction, DreamVLA (Zhang et al., 2025) forecasts dynamic regions, depth, and semantic features, while VLA-JEPA (Sun et al., 2026) models future latent states and JEPA-WAM (Lin et al., 2026b) learns joint current–future representations. InternVLA-A1.5 (Physical Intelligence Team, Shanghai AI Laboratory, 2026) learns foresight queries with supervision from a frozen video generator over the action horizon. ViPRA (Routray et al., 2026) jointly predicts future observations and motion-centric latent actions during video pretraining. These approaches specify future targets through frame offsets, action horizons, or predefined future sequences.

Beyond fixed-horizon prediction, recent methods organize visual foresight around task progress. StructVLA (Jin et al., 2026) generates sparse structured frames as progress anchors, while Stage-WAM (Liu et al., 2026a) predicts a next-stage latent to condition short-horizon video and action generation. These methods represent task progress through future images or latent states. PALM (Liu et al., 2026b) combines affordance foresight with explicit within-subtask progress prediction. MEF-VLA instead uses the next interaction event as a reference for predicting the spatial change remaining from the current observation.

2.3 Motion Representations for Manipulation

Motion representations capture scene changes that support action learning. TraceVLA (Zheng et al., 2025) augments policy inputs with historical point traces. LAPA (Ye et al., 2025) learns discrete latent actions from image pairs for VLA pretraining, while UniVLA (Bu et al., 2025) learns task-centric latent actions from visual feature transitions. Other approaches explicitly predict motion to support visual foresight and action generation. HiF-VLA (Lin et al., 2026a) integrates historical and predicted future motion representations into action generation. FlowVLA (Zhong et al., 2025) predicts optical flow as an intermediate step in future-frame generation during world-model pretraining. FoMoVLA (Li et al., 2026) jointly supervises future-frame features and 2D point trajectories over the action horizon, while LaMP (Wang et al., 2026) conditions action generation on a fixed-horizon dense 3D scene-flow prior. MEF-VLA defines the motion target relative to the next interaction event, allowing the prediction horizon to adapt to task progress. Observations sharing an event reference receive distinct displacement targets determined by their current states.

3 Method

MEF-VLA learns the motion remaining from the current observation to the next interaction event (Fig. 1). We construct event references from demonstrations and represent the displacement toward each reference as remaining event-flow. As illustrated in Fig. 2, event-flow preprocessing constructs these targets, and joint event-foresight and policy learning uses them to supervise event queries that condition action generation.

3.1 Problem Formulation

We learn a language-conditioned policy $\pi_\theta(a_{t:t+H-1} \mid o_t, s_t, \ell)$ from demonstrations, where o_t contains the current RGB observations, s_t is the robot state when provided, ℓ is the instruction, and $a_{t:t+H-1}$ is an H-step action chunk. For each valid training sample, MEF-VLA additionally supervises the spatial change from o_t to an event frame at $\tau(t) > t$ selected from the demonstration. The action horizon H specifies the predicted chunk length, while $\tau(t) - t$ defines the event-aligned

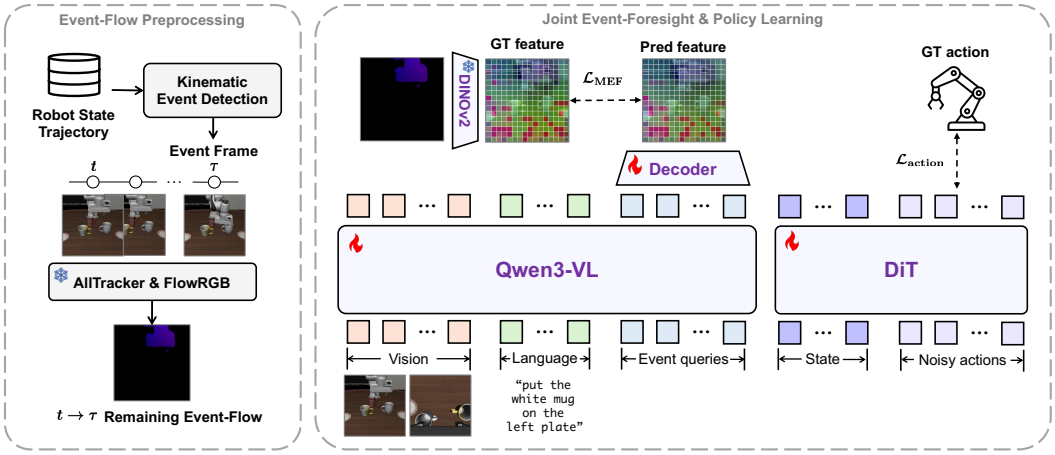

Figure 2: **Overview of MEF-VLA.** Left: event-flow preprocessing constructs event references and remaining event-flow targets, which are encoded as FlowRGB and cached once per dataset. Right: joint event-foresight and policy learning supervises event queries with features extracted by a frozen DINOv2 encoder. The queries predict remaining-motion features from current observations and the instruction, and condition the DiT action expert. The target encoder and patch decoder are used only during training.

prediction horizon. The future demonstration supplies the supervision target; the policy learns to infer its features from the current observations and instruction.

3.2 Event-Aligned Prediction Horizon

To align predictive supervision with task progress, we use the next interaction as a shared reference for the ongoing manipulation. We construct event references using a kinematic detector that identifies the onsets of persistent low-speed intervals in demonstration trajectories. These onsets provide cues for interactions such as grasp stabilization and placement. Detection uses trajectory-adaptive speed thresholds and a minimum interval duration. Let $\mathcal{E}$ denote the resulting event times, including the final frame as a terminal reference. For each nonterminal time step, we select the first event strictly after the current state,

$$\tau(t) = \min\{e \in \mathcal{E} : e > t\}, \qquad \Delta_t = \tau(t) - t. \tag{1}$$

Observations between consecutive events therefore share a future reference, while Δ_t decreases as execution approaches that event. At an event time, the reference advances to the following event. This assignment provides a common prediction reference within each phase and updates it as execution progresses, allowing its temporal distance to vary across states and demonstrations. Appendix A.4 specifies the detector and boundary handling. On 80 LIBERO demonstrations, the detected events achieve 83.9% F1 against human-annotated interactions within a ± 1 s tolerance (Appendix D).

3.3 Remaining Event-Flow Supervision

The event reference specifies the upcoming interaction toward which the policy should act; the supervision target describes the spatial changes needed to reach it from the current observation. An event RGB frame retains scene appearance, including background and texture, while giving the same future-image target to observations that share that event. We represent the motion remaining until the next event as remaining event-flow, a dense field of current-to-event displacement. Its dependence on both the event reference and the current observation provides state-dependent supervision.

Current-to-event correspondence. Let I_t denote the primary-camera image in o_t. We apply the frozen dense point tracker AllTracker (Harley et al., 2025) to the clip $I_{t:\tau(t)}$. For a pixel $x_t = x$ in

the current image, its tracked position $x_{\tau(t)}$ in the event frame defines

$$u_{t\to\tau}(x) = x_{\tau(t)} - x_t, \qquad M_{t\to\tau}(x) = \mathbf{1}[V_{t\to\tau}(x) \geq \gamma], \tag{2}$$

where V is the tracker visibility estimate and M is a visibility mask used to suppress unreliable displacement estimates when constructing FlowRGB. The tracker uses intermediate frames to estimate long-range correspondences, while the target stores only the net displacement between the current and event endpoints. Each vector describes the direction and extent of the spatial change at its current-image location. Thus, two observations leading to the same grasp can share an event frame but have different remaining-flow targets as the gripper and object configurations change.

Spatial feature supervision. We encode displacement direction and magnitude as a three-channel image $R_t^{\text{rem}} = \text{FlowRGB}(u_{t\to\tau}, M)$, preserving the spatial layout of the remaining change. Features of this motion representation provide the prediction target for policy learning. During training, a frozen DINOv2 model (Oquab et al., 2024) encodes each cached FlowRGB field on the fly to obtain the patchwise target

$$Z_t^* = \text{sg}[\phi_{\text{DINO}}(R_t^{\text{rem}})] \in \mathbb{R}^{P\times d_z}, \tag{3}$$

where sg denotes stop-gradient, P is the number of patches, and d_z is the feature dimension. We retain the patch grid so that the predictive loss compares features at corresponding spatial locations. The encoder remains frozen, providing a fixed mapping from remaining event-flow to the feature space used for supervision. These fields are precomputed and cached once per dataset; normalization and color encoding are specified in Appendix A.4.

3.4 Joint Event-Foresight and Policy Learning

Event-query prediction. Following query-based foresight (Yang et al., 2026; Zhang et al., 2025), we append K learned event queries Q^E to the image–language sequence to predict remaining-motion features for action generation. The VLM produces

$$H_t = \text{VLM}_\theta(o_t, \ell, Q^E), \qquad H_t^E = H_t[Q^E] \in \mathbb{R}^{K\times d}. \tag{4}$$

Here, H_t contains the full multimodal hidden sequence and H_t^E denotes the states at the event-query positions. The learned query embeddings are shared across samples, while their output states depend on the current visual and linguistic context. During training, a lightweight decoder maps H_t^E to the spatial grid of the teacher target. Its P learned patch queries, associated with spatial positions, cross-attend to the event-query states:

$$\widehat{Z}_t = D_\psi(H_t^E) \in \mathbb{R}^{P\times d_z}. \tag{5}$$

We supervise the prediction with

$$\mathcal{L}_{\text{MEF}} = \frac{1}{|\mathcal{V}|}\sum_{t\in\mathcal{V}} \text{SmoothL1}_\beta\left(\widehat{Z}_t, Z_t^*\right), \tag{6}$$

where $\mathcal{V}$ contains samples with a valid next-event target and Smooth L1 is averaged over patches and feature dimensions. The decoder serves as a training-time readout; action conditioning uses the event-query states H_t^E themselves.

Action conditioning. The complete hidden sequence H_t, including H_t^E, and the robot state when used condition the GR00T-style action expert (StarVLA Community, 2026). Following conditional flow matching (Lipman et al., 2023), we interpolate between Gaussian noise ϵ and the demonstrated action chunk a as $a_r = (1-r)\epsilon + ra$, with r drawn from the training-time distribution specified in Appendix A, and optimize

$$\mathcal{L}_{\text{action}} = \mathbb{E}_{a,\epsilon,r}\left[\|v_\theta(a_r, r; H_t, s_t) - (a-\epsilon)\|_2^2\right]. \tag{7}$$

Here, v_θ is the action expert's velocity prediction and r is the flow-matching time. Conditioning on the full sequence gives the action expert access to both the current multimodal context and the event-query representation.

Joint training and inference. We jointly optimize the predictive and action objectives,

$$\mathcal{L} = \mathcal{L}_{\text{action}} + \lambda_{\text{MEF}}\mathcal{L}_{\text{MEF}}, \tag{8}$$

where λ_{MEF} weights the predictive loss. The predictive loss trains the query states to encode remaining-motion features, while the action loss trains their use in control through the shared conditioning path. Both losses optimize the event queries and shared VLM. At deployment, event-query states are computed from the current observations and instruction, and the action expert integrates its predicted velocity field from Gaussian noise to generate an action chunk, using robot state when provided. The target encoder and patch decoder serve only as training-time supervision modules; deployment retains the event queries that condition the action expert. Appendix B summarizes the components used during preprocessing, training, and inference, and reports the associated preprocessing and inference costs.

4 Experiments

We evaluate MEF-VLA on standard manipulation tasks, long-horizon instruction sequences, and zero-shot perturbations. Real-world experiments further assess multi-stage instructions at seen and unseen composition lengths. Controlled ablations examine the roles of event alignment and remaining-motion supervision.

4.1 Experimental Setup

Benchmarks and metrics. We evaluate MEF-VLA on LIBERO (Liu et al., 2023), CALVIN (Mees et al., 2022), and LIBERO-Plus (Fei et al., 2026). LIBERO contains four suites—Spatial, Object, Goal, and Long—and reports success rates for each suite and the average across suites. CALVIN D→D evaluates sequential instruction following in environment D after training on demonstrations from the same environment. Following the official evaluation protocol, we evaluate on 1,000 five-task instruction sequences and report success rates for completing the first one to five consecutive tasks and the average completed sequence length (Avg. Len.). LIBERO-Plus applies seven controlled perturbation families to LIBERO tasks. For LIBERO-Plus, we report the benchmark Total, which aggregates successful trials over all perturbed instances and therefore accounts for the different category sizes.

Table 1: LIBERO success rates (%).

Method	Spatial	Object	Goal	Long	Avg.
π_0	98.0	96.8	94.4	88.4	94.4
$\pi_{0.5}$	<u>98.8</u>	98.2	<u>98.0</u>	92.4	96.9
UniVLA	96.5	96.8	95.6	92.0	95.2
FlowVLA	93.2	95.0	91.6	72.6	88.1
DreamVLA	97.5	94.0	89.5	89.5	92.6
Mantis	<u>98.8</u>	99.2	94.4	94.2	96.7
StructVLA	95.4	98.0	93.4	92.2	94.8
VLA-JEPA	96.2	**99.6**	97.2	**95.8**	<u>97.2</u>
JEPA-WAM	95.6	<u>99.4</u>	97.2	<u>94.6</u>	96.7
MEF-VLA	**99.4**	98.6	**98.6**	**95.8**	**98.1**

Training and evaluation protocol. We initialize the VLM from Qwen3-VL-4B-Instruct (Bai et al., 2025) and the GR00T-style action expert from scratch. The full policy is optimized jointly with the predictive branch. A single policy is trained across the four LIBERO suites and evaluated zero-shot on LIBERO-Plus, while a separate policy is trained on language-annotated demonstrations from CALVIN environment D. All cross-method comparisons use the full MEF-VLA policy. Vanilla VLA denotes an action-only reference with the same VLM backbone and action-expert architecture, without event queries or remaining-motion supervision. Training configurations are provided in Appendix A.

4.2 Simulation Experiments

Tables 1–3 compare MEF-VLA with representative robot policies on standard manipulation, sequential instruction following, and zero-shot perturbations. Comparisons include $\pi_{0.5}$ (Physical Intelligence et al., 2025) and OpenVLA-OFT (Kim et al., 2025a), alongside the methods discussed in Sec. 2. Results for prior methods are drawn from published papers and official reports. Best results are bold, and second-best results are underlined where indicated.

Table 2: CALVIN D→D results. Success rates (%) for completing the first one to five tasks consecutively and average completed sequence length (Avg. Len.).

Method	Tasks completed in a row					Avg. Len. ↑
	1	2	3	4	5	
OpenVLA	71.6	38.5	18.0	8.8	4.2	1.411
π_0	84.8	70.4	55.9	46.6	37.7	2.954
$\pi_{0.5}$	**92.5**	84.0	76.6	71.0	64.4	3.885
Vanilla VLA	89.8	79.9	71.8	64.9	58.1	3.645
MEF-VLA	91.5	**84.4**	**77.9**	**73.7**	**67.1**	**3.946**

Table 3: Zero-shot comparison on LIBERO-Plus. Success rates (%) are reported; Total is the overall success rate across all perturbed instances. The VLA-JEPA and JEPA-WAM Totals are recomputed from published category rates and official category sizes.

Method	Camera	Robot	Lang.	Light	Backgr.	Noise	Layout	Total
OpenVLA	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
OpenVLA-OFT	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
π_0	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
UniVLA	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
DreamVLA	26.2	17.6	67.0	77.5	71.5	53.6	43.5	48.9
VLA-JEPA	63.3	**67.1**	85.4	**95.6**	93.6	66.3	**85.1**	78.0
JEPA-WAM	**79.2**	59.2	68.2	93.3	94.6	**83.6**	76.1	77.9
MEF-VLA	61.2	55.7	**87.4**	**95.6**	**96.6**	82.4	79.3	**78.2**

LIBERO performance. MEF-VLA achieves an average success rate of 98.1% across the four LIBERO suites, outperforming the strongest prior method in Table 1 by 0.9 percentage points. Performance remains strong on both short- and long-horizon tasks.

Long-horizon evaluation on CALVIN. We evaluate sequential instruction following on CALVIN D→D (Table 2). MEF-VLA achieves an average completed sequence length of 3.946, compared with 3.645 for Vanilla VLA. Success on complete five-task sequences increases from 58.1% to 67.1%, while first-task success increases from 89.8% to 91.5%. The larger gap on longer sequences indicates stronger long-horizon execution performance.

Zero-shot robustness. On LIBERO-Plus, MEF-VLA achieves 78.2% Total success without additional training, remaining competitive with recent predictive methods (Table 3). Section 4.4 examines the contributions of event alignment and remaining-motion supervision under shared training settings.

4.3 Real-World Experiments

Platform and tasks. We deploy policies on a mobile dual-arm platform equipped with dexterous hands, one head RGB camera, and two wrist RGB cameras (Fig. 3, left). The three task families emphasize complementary capabilities within a shared multi-stage setting. *Pill Sorting* tests precise grasping of small objects by placing color-specified pills into a tray in the instructed order. *Sandwich Making* tests bimanual manipulation by stacking an instructed sequence of ingredients on a plate. *Cup Stacking* tests spatial understanding by placing specified cups over other cups in sequence, with the positions of four differently colored cups randomized within the workspace. Each pill placement, ingredient placement, or cup-stacking operation defines one stage. Each rollout follows a single instruction comprising $N \in \{1, 2, 3\}$ stages in a specified order. We vary instruction length using familiar objects and manipulation primitives.

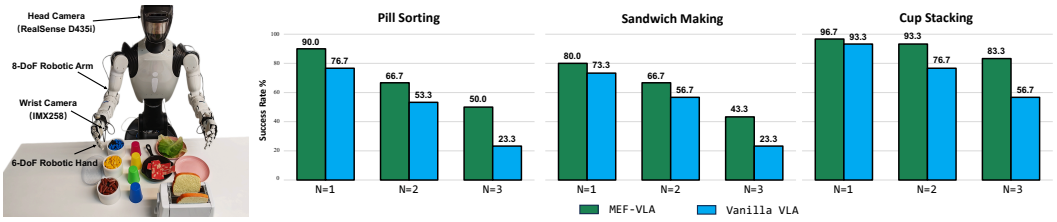

Figure 3: **Real-world setup and evaluation.** Left: the mobile dual-arm platform and its workspace. Right: instruction success rates (%) over 30 trials per task family, method, and instruction length N, measured in stages. Training demonstrations contain one or two stages; three-stage instructions test generalization to longer compositions. MEF-VLA outperforms Vanilla VLA in all nine settings.

Table 4: Predictive target ablations on LIBERO and zero-shot LIBERO-Plus. A1 replaces the event-aligned horizon with the action horizon while retaining flow supervision; A2 retains event alignment and predicts event-frame features. Success rates (%) are reported.

Configuration	LIBERO					LIBERO-Plus
	Spatial	Object	Goal	Long	Avg.	Total
Vanilla VLA	97.8	98.8	97.4	92.0	96.5	69.3
MEF-VLA (Full)	**99.4**	98.6	**98.6**	**95.8**	**98.1**	**78.2**
Action horizon (A1)	98.2	98.8	96.8	93.6	96.9	73.3
Event frame (A2)	98.8	**99.6**	95.4	95.6	97.4	73.7

Training and evaluation. We train a separate policy for each task family using teleoperated demonstrations of one- and two-stage instructions. The datasets contain 300 Pill Sorting, 500 Sandwich Making, and 300 Cup Stacking demonstrations; all three-stage compositions are withheld. Within each task family, MEF-VLA and Vanilla VLA use the same training demonstrations and real-time chunking (RTC) (Black et al., 2025b) for inference. Evaluation covers seen instruction lengths ($N = 1, 2$) and held-out three-stage compositions ($N = 3$), with 30 trials per task family, length, and method under matched instructions and initial configurations. Instruction success requires completing all N stages in order. Appendix E provides implementation and metric details.

Execution at seen instruction lengths. MEF-VLA improves success over Vanilla VLA in every one- and two-stage setting (Fig. 3). Averaged across task families, success increases from 81.1% to 88.9% for one-stage instructions and from 62.2% to 75.6% for two-stage instructions.

Generalization to longer instructions. For held-out three-stage instructions, MEF-VLA achieves 50.0% success on Pill Sorting, 43.3% on Sandwich Making, and 83.3% on Cup Stacking, compared with 23.3%, 23.3%, and 56.7% for Vanilla VLA. Average success across task families increases from 34.4% to 58.9%. The larger improvement on three-stage instructions supports the use of motion-to-event foresight for longer instruction compositions. The failure analysis also shows fewer instruction-following errors on three-stage instructions (Appendix E.3).

4.4 Ablation Study

We compare the full motion-to-event target with two alternatives: motion over the action horizon (A1) and the RGB state at the next event (A2) (Table 4). All three models share the event-query interface, decoder capacity, and optimization settings, and are trained on LIBERO and evaluated on LIBERO and zero-shot LIBERO-Plus. Compared with Vanilla VLA, the full model improves average success on LIBERO from 96.5% to 98.1% and Total success on LIBERO-Plus from 69.3% to 78.2%. The largest gain on LIBERO occurs on Long, where success increases from 92.0% to 95.8%. Appendix C provides configuration details and results for all seven perturbation categories.

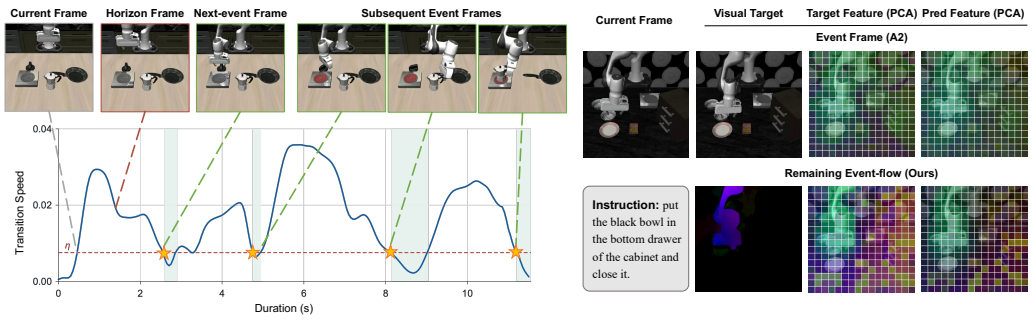

Figure 4: **Event-aligned horizons and remaining-motion supervision on LIBERO-Long.** Left: the speed profile and threshold η are shown with detected persistent low-speed intervals (shaded); their onsets define event references (stars). The fixed-horizon frame precedes the next interaction, while later frames show subsequent events. Right: in a separate example, A2 and MEF-VLA share the current observation and next event but predict features of the event RGB frame (top) and remaining event-flow (bottom), respectively. Each row shows the supervision target and PCA visualizations of the target and predicted features, overlaid on the current observation.

Effect of event-aligned prediction horizons. A1 retains flow-feature supervision but uses a fixed prediction horizon equal to the action horizon. Replacing this fixed-horizon target with the next-event target increases LIBERO average success from 96.9% to 98.1% and LIBERO-Plus Total from 73.3% to 78.2%. Figure 4 (left) illustrates this distinction on a LIBERO-Long trajectory: the fixed-horizon frame lies within an intermediate movement, while the next-event frame marks the upcoming interaction boundary. The event reference therefore adapts to the duration of the ongoing phase. These results support using the next interaction as a shared prediction reference for observations within a manipulation phase.

Effect of remaining-motion supervision. A2 uses the same event references and prediction architecture as the full model, but predicts DINOv2 features of the event RGB frames. Replacing these event-frame targets with remaining-motion features increases LIBERO average success from 97.4% to 98.1% and LIBERO-Plus Total from 73.7% to 78.2%. The full model also outperforms A2 across all seven perturbation categories (Table 10). Figure 4 (right) compares the two supervision targets and their predicted features on a LIBERO-Long example with the same current observation and next event. In A2's PCA maps, the robot arm and parts of the static scene appear in similar colors. Remaining event-flow supervision yields a clearer separation between the moving arm and static regions, a spatial pattern also retained in the predicted features. These gains support using state-dependent motion toward the next event as supervision for action learning.

5 Conclusion and Limitations

We presented MEF-VLA, a VLA framework that learns motion-to-event foresight to guide action generation toward upcoming interactions. Event-aligned horizons provide task-relevant references, and remaining event-flow supervises the state-dependent spatial change toward them. Experiments on LIBERO, LIBERO-Plus, and CALVIN show improvements in task success, zero-shot robustness, and sequential instruction following. Across three real-world task families, policies trained on one- and two-stage instructions generalize to held-out three-stage compositions. Together with controlled ablations, these results support aligning motion prediction with interaction events as a useful training objective for multi-stage manipulation.

Our current event construction uses kinematic cues from demonstration trajectories. Extending it to interactions with less pronounced motion pauses is a direction for future work. Remaining event-flow provides image-space motion supervision, without explicitly modeling contact forces. Complementing this supervision with force or tactile signals could support manipulation requiring precise contact control.

AI USE STATEMENT

We used generative AI tools to refine the language and improve the readability of the manuscript. We also used these tools to identify relevant literature, summarize prior work, and compare related methods. For research ideation and analysis, we used AI assistance to discuss the method formulation, obtain feedback on experimental and ablation designs, and interpret quantitative and qualitative results. The authors make the final research decisions and are responsible for verifying AI-assisted text, references, and analyses, and for the final content and claims of this paper.

REPRODUCIBILITY STATEMENT

We describe the model architecture and training objectives in Section 3, and the benchmarks, evaluation protocols, and metrics in Section 4. Appendix A provides model initialization, optimization settings, data preprocessing, and the configurations for event detection and remaining event-flow construction. Appendix C details the ablation settings and feature visualization procedure, Appendix D specifies the event reference validation protocol, and Appendix E documents the real-world evaluation protocol and success criteria.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025a. doi: 10.15607/RSS.2025.XXI.010.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems*, volume 38, 2025b.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. UniVLA: Learning to act anywhere with task-centric latent actions. In *Robotics: Science and Systems*, 2025.

Zhejia Cai, Yandan Yang, Xinyuan Chang, Shiyi Liang, Ronghan Chen, Feng Xiong, Mu Xu, and Ruqi Huang. Seeing space and motion: Enhancing latent actions with geometric and dynamic awareness for vision-language-action models. In *IEEE International Conference on Robotics and Automation*, 2026.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations*, 2024.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 38574–38583, 2026.

Adam W. Harley, Yang You, Xinglong Sun, Yang Zheng, Nikhil Raghuraman, Yunqi Gu, Sheldon Liang, Wen-Hsuan Chu, Achal Dave, Suya You, Rares Ambrus, Katerina Fragkiadaki, and Leonidas Guibas. AllTracker: Efficient dense point tracking at high resolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 5253–5262, 2025.

Minghao Jin, Mozheng Liao, Mingfei Han, Zhihui Li, and Xiaojun Chang. Beyond dense futures: World models as structured planners for robotic manipulation. *arXiv preprint arXiv:2603.12553*, 2026.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025a.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P. Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713, 2025b.

Wei Li, Peijin Jia, Yuan Ma, Xuefeng Jiang, Titong Jiang, Sheng Sun, Yujian Li, Xin Wen, Han Hong, Zhikang Liu, Bailin Li, and Kun Zhan. FoMoVLA: Bridging visual foresight and motion guidance for vision-language-action models. *arXiv preprint arXiv:2607.14739*, 2026.

Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. HiF-VLA: Hindsight, insight and foresight through motion representation for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026a.

Yihan Lin, Jiawei He, Shifeng Bao, Chen Zhao, Yang Li, Xiaobo Wang, Yan Wang, Cheng Chi, and Jing Zhang. JEPA-WAM: Learning vision-language-action policies with joint-embedding world modeling. *arXiv preprint arXiv:2608.09381*, 2026b.

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

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

Xiao Liu, Yuguang Yang, Xi Wang, Kai Jiang, Cheng Chi, Yong Xu, Wenchao Ding, Yilun Chen, and Yan Wang. StageWAM: Joint-embedding stage prediction for world-action models in robot manipulation. *arXiv preprint arXiv:2608.10780*, 2026a.

Yuanzhe Liu, Jingyuan Zhu, Yuchen Mo, Gen Li, Xu Cao, Jin Jin, Yifan Shen, Zhengyuan Li, Tianjiao Yu, Wenzhen Yuan, Fangqiang Ding, and Ismini Lourentzou. PALM: Progress-aware policy learning via affordance reasoning for long-horizon robotic manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 2022.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, et al. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Physical Intelligence Team, Shanghai AI Laboratory. InternVLA-A1.5: Unifying understanding, latent foresight, and action for compositional generalization. *arXiv preprint arXiv:2607.04988*, 2026.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. ZeRO: Memory optimizations toward training trillion parameter models. In *International Conference for High Performance Computing, Networking, Storage and Analysis*, 2020.

Sandeep Kumar Routray, Hengkai Pan, Unnat Jain, Shikhar Bahl, and Deepak Pathak. ViPRA: Video prediction for robot actions. In *International Conference on Learning Representations*, 2026.

StarVLA Community. StarVLA: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. VLA-JEPA: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.

Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. In *International Conference on Learning Representations*, 2025.

Xinkai Wang, Chenyi Wang, Yifu Xu, Mingzhe Ye, Fucheng Zhang, Jialin Tian, Xinyu Zhan, Lifeng Zhu, Cewu Lu, and Lixin Yang. LaMP: Learning vision-language-action policy with 3d scene flow as latent motion prior. In *European Conference on Computer Vision*, 2026.

Yi Yang, Xueqi Li, Yiyang Chen, Jin Song, Yihan Wang, Zipeng Xiao, Jiadi Su, You Qiaoben, Pengfei Liu, and Zhijie Deng. Mantis: A versatile vision-language-action model with disentangled visual foresight. *arXiv preprint arXiv:2511.16175*, 2026.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *International Conference on Learning Representations*, 2025.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, Li Yi, Wenjun Zeng, and Xin Jin. DreamVLA: A vision-language-action model dreamed with comprehensive world knowledge. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *International Conference on Learning Representations*, 2025.

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

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183, 2023.

APPENDIX OVERVIEW

Appendix A provides implementation and training details, Appendix B summarizes component usage and reports preprocessing and inference costs, and Appendix C presents additional analyses of prediction targets. Appendix D evaluates the temporal alignment of the constructed event references, while Appendix E documents the real-world evaluation protocol and failure modes.

A IMPLEMENTATION AND TRAINING DETAILS

A.1 MODEL INITIALIZATION AND OPTIMIZATION

We initialize the VLM from Qwen3-VL-4B-Instruct and randomly initialize a GR00T-style DiT-B action expert with 16 layers and hidden dimension 1024. We jointly optimize all parameters of the VLM, action expert, event queries, and patch decoder. The DINOv2 target encoder remains frozen during training; AllTracker is used only during preprocessing and remains frozen. Table 5 summarizes the shared configuration, and Table 6 lists dataset-specific settings. We use the final training checkpoint for evaluation.

Table 5: Shared model and optimization settings. Dataset-specific values are listed in Table 6.

Configuration	Value
VLM initialization	Qwen3-VL-4B-Instruct
Action expert	DiT-B, random initialization
Optimization	All parameters of the policy and patch decoder
Optimizer	AdamW (Loshchilov & Hutter, 2019)
Optimizer (β_1, β_2) / ϵ	$(0.9, 0.95)$ / 10^{-8}
Weight decay	10^{-8}
VLM learning rate	10^{-5}
Action-expert learning rate	10^{-4}
Patch-decoder learning rate	4×10^{-5}
Learning-rate schedule	Linear warmup, cosine decay
Minimum base learning rate	10^{-6}
Maximum gradient norm	1.0
Image / target resolution	224×224
Observation history	Current observations only
Training precision	BF16 mixed precision
Training seed	42

We train on NVIDIA A100 80GB GPUs with DeepSpeed ZeRO-2 (Rajbhandari et al., 2020) and gradient checkpointing. Each optimizer update uses one global batch, with no gradient accumulation. For flow matching, we draw $u \sim \mathrm{Beta}(1.5, 1.0)$ and use $r = (0.999 - \min(u, 0.999))/0.999$. We repeat time and noise sampling eight times per observation on LIBERO and real-world tasks and four times on CALVIN.

The dataset-specific auxiliary weights were selected based on the relative gradient magnitudes of the action and MEF losses at convergence. The base learning rate applies to parameters outside the module-specific groups in Table 5. Policy images are resized to 224×224. On LIBERO and real-world tasks, image augmentation uses a crop fraction of 0.95 and brightness, contrast, saturation, and hue settings of 0.3, 0.4, 0.5, and 0.08, respectively. Policy image augmentation is disabled on CALVIN.

A.2 SIMULATION TRAINING

LIBERO. We jointly train one policy on the Spatial, Object, Goal, and Long suites using the no-op-filtered demonstrations. Dataset and trajectory balancing are disabled. The policy receives global and wrist RGB observations together with the current end-effector pose and gripper state, and predicts seven-dimensional actions comprising translation, rotation, and a gripper command.

Table 6: Training settings by dataset. Pill, Sandwich, and Cup denote Pill Sorting, Sandwich Making, and Cup Stacking. Real-world training budgets and optimization settings are shared by MEF-VLA and Vanilla VLA.

Configuration	LIBERO	CALVIN	Pill	Sandwich	Cup
Training steps	30,000	30,000	10,000	15,000	10,000
Warmup steps	5,000	5,000	1,667	2,500	1,667
Global batch size	128	32	64	64	64
Number of GPUs	8	8	4	4	4
Base learning rate	4×10^{-5}	2.5×10^{-5}	4×10^{-5}	4×10^{-5}	4×10^{-5}
Action chunk length	16	8	16	16	16
Flow-matching samples per observation	8	4	8	8	8
Auxiliary weight λ_{MEF}	0.05	0.075	0.05	0.05	0.05
Image augmentation	Yes	No	Yes	Yes	Yes
Trajectory balancing	No	No	Yes	Yes	Yes

LIBERO-Plus uses the same final checkpoint for zero-shot evaluation. The full model and A1/A2 variants use the same optimization settings in Tables 5 and 6.

CALVIN. We train a single multitask policy on 5,124 language-annotated episodes from environment D. The policy receives static-camera and gripper-camera RGB observations and the instruction, with robot-state conditioning disabled. Robot poses are used for offline event-reference construction. Training uses the dataset's seven-dimensional relative actions, including its gripper action command. Dataset and trajectory balancing are disabled. On CALVIN, Vanilla VLA uses the same global batch size, action horizon, number of flow-matching samples per observation, robot-state inputs, and training budget as the full model.

A.3 Real-World Training

We train separate policies for Pill Sorting, Sandwich Making, and Cup Stacking on 300, 500, and 300 demonstrations, respectively. All training instructions contain one or two stages. The training data are resampled to 10 Hz, and trajectory balancing is enabled. Within each task family, MEF-VLA and Vanilla VLA use the same data, optimizer settings, training budget, and input and action masks.

Pill Sorting and Cup Stacking use head and right-wrist RGB images and right-arm and hand state; left-arm inputs and action supervision are masked. Sandwich Making uses the head and both wrist views, with state inputs and action supervision for both arms and hands. The robot state comprises two six-dimensional end-effector poses and two continuous hand values. Actions contain translations and $\mathrm{SO}(3)$ rotation vectors relative to the current observed pose, together with continuous hand commands. All steps in an action chunk use the same current-pose reference. For the active arms, pose actions are normalized using per-dimension 1st and 99th percentiles, while pose states use min–max normalization. Hand values $h \in [0, 1]$ are mapped to $2h - 1$.

A.4 Event-Reference and Motion-Target Construction

Event references. We smooth pose trajectories with an edge-padded centered moving average. Table 7 reports the window radius, episode-level speed quantile, and minimum low-speed duration. Simulation uses six-dimensional end-effector poses. For the real-world robot, poses are scaled using fixed per-dimension 1st and 99th percentiles, independently of policy normalization. Events are detected separately for each arm and merged by taking their temporal union. Let $s_t^{(g)}$ denote one of G non-overlapping kinematic state groups in a trajectory of length T. For the smoothed state $\bar{s}_t^{(g)}$, transition speeds and episode-adaptive thresholds are

$$v_t^{(g)} = \left\| \bar{s}_{t+1}^{(g)} - \bar{s}_t^{(g)} \right\|_2, \qquad \eta^{(g)} = Q_q\left(\{v_t^{(g)}\}_{t=0}^{T-2} \right), \tag{9}$$

where Q_q denotes the quantile at level q. A low-speed interval is a maximal interval $[b, c]$, with $b > 0$, containing at least p consecutive transitions satisfying $v_t^{(g)} \le \eta^{(g)}$ and preceded by $v_{b-1}^{(g)} > \eta^{(g)}$. The onset b of each persistent low-speed interval defines an event. Events from all groups are merged and deduplicated, and the final frame is appended as a terminal reference:

$$\mathcal{E} = \mathrm{UniqueSort}\left(\bigcup_{g=1}^{G} \mathcal{E}^{(g)} \cup \{T-1\} \right). \tag{10}$$

Equation 1 assigns each nonterminal frame to the next event strictly after it; the terminal frame itself is excluded from the auxiliary objective.

Table 7: Event-reference construction settings. A smoothing radius w corresponds to $2w+1$ frames.

Dataset	Radius w	Quantile q	Persistence p
LIBERO	2	0.20	3
CALVIN	2	0.22	3
Pill Sorting	2	0.22	4
Sandwich Making	4	0.25	4
Cup Stacking	4	0.25	3

Motion targets and feature prediction. We construct remaining event-flow targets using the global camera in LIBERO, the static camera in CALVIN, and the head camera in all three real-world tasks, with the official AllTracker weights. The visibility threshold is $\gamma = 0.5$. FlowRGB represents the net current-to-event displacement. For image size $H_I \times W_I$, let $d_I = \sqrt{H_I^2 + W_I^2}$. We normalize the masked displacement magnitude as

$$r(x) = M(x)\frac{\|u_{t\to\tau}(x)\|_2}{d_I}, \qquad \widetilde{r}(x) = \mathrm{clip}\left(\frac{r(x)}{s_{\mathrm{flow}}}, 0, 1\right), \tag{11}$$

where s_{flow} is the dataset-specific 99th percentile of $r(x)$ over visible training pixels. The field is encoded in HSV and converted to RGB:

$$\begin{aligned} \mathrm{Hue}(x) &= \frac{\mathrm{atan2}(u_y(x), u_x(x)) + \pi}{2\pi}, \\ \mathrm{Sat}(x) = M(x), &\qquad \mathrm{Val}(x) = \widetilde{r}(x). \end{aligned} \tag{12}$$

Hue represents displacement direction, saturation marks reliable correspondences, and value represents normalized displacement magnitude. Invalid locations are encoded as zero-valued FlowRGB pixels before feature extraction. We extract 256 patch features of dimension 768 after the final layer normalization of the frozen DINOv2 ViT-B/14 encoder with registers (Darcet et al., 2024). A two-layer patch decoder with hidden dimension 512, eight attention heads, and zero dropout predicts these features from 16 event queries. Its learned patch queries use fixed two-dimensional positional encodings and cross-attend to the event-query states. The Smooth L1 parameter is $\beta = 0.1$, and auxiliary loss weights are listed in Table 6.

B Computational Efficiency

B.1 Preprocessing, Training, and Inference

Table 8 summarizes the components used during preprocessing, training, and inference. FlowRGB targets are precomputed and cached once per dataset, then encoded by the frozen DINOv2 target encoder during training. At inference, the policy uses the VLM backbone, learned event queries, and action expert.

Table 8: Components used during preprocessing, training, and inference.

Component	Preprocessing	Training	Inference
Kinematic event detector	✓	–	–
AllTracker point tracker	✓	–	–
DINOv2 target encoder	–	✓	–
Patch decoder	–	✓	–
Event queries	–	✓	✓
VLM backbone and action expert	–	✓	✓

B.2 Inference and Preprocessing Costs

Policy inference. Table 9 compares action-chunk inference latency under identical inputs and action-sampling settings. We benchmark both policies using 40 observations sampled across the four LIBERO suites. MEF-VLA increases mean latency from 118.29 to 119.55 ms per chunk, an overhead of approximately 1.3 ms (1.1%). The deployed policy retains the learned event queries, while the target encoder and patch decoder are used only during training. These measurements support the low inference overhead of motion-to-event foresight.

Table 9: Action-chunk inference latency on LIBERO. Results are mean $\pm$ standard deviation over 1,000 inference calls using identical preprocessed inputs and action-sampling settings on a single NVIDIA A100 80GB GPU.

Method	Inference latency (ms/chunk) $\downarrow$
Vanilla VLA	118.29 ± 0.86
MEF-VLA	119.55 ± 1.05

One-time preprocessing. Constructing the remaining event-flow targets for all four LIBERO suites takes 4.69 hours on four NVIDIA A100 80GB GPUs (18.76 GPU-hours), covering 1,693 demonstrations and 271,772 valid targets. This measurement includes event detection, correspondence estimation, flow encoding, and cache generation. The targets are cached once per dataset and reused during training; target feature encoding by DINOv2 is performed during training and is excluded from this preprocessing time.

C Additional Analysis of Prediction Targets

C.1 Ablation Configurations

We compare the full model with two variants that change either the prediction horizon or the supervised representation. Action horizon (A1) uses a fixed horizon equal to the action horizon, $\tau(t) = \min(t + H, T - 1)$, and predicts DINOv2 features of the resulting FlowRGB targets. The flow-magnitude scale is recalibrated for these fixed-horizon targets. Event frame (A2) retains the event-aligned horizon and predicts DINOv2 features of the event RGB frame. All three models use $K = 16$ event queries conditioned on the current observations and instruction, and share the VLM, action expert, decoder capacity, optimization settings, training budget, and rollout protocol. They are trained on the standard LIBERO demonstrations and evaluated zero-shot on LIBERO-Plus under the same protocol as the full-model comparison.

Feature visualization. For the feature visualizations in Fig. 4 (right), PCA is performed jointly on the target and predicted patch features for each supervision type, ensuring a shared projection basis within each target–prediction pair. The first three principal components are mapped to RGB using a shared color scale within each pair.

Table 10: Predictive target ablations across LIBERO-Plus perturbations. All configurations are trained on LIBERO and evaluated zero-shot. Success rates (%) are reported; Total is the overall success rate across all perturbed instances. Best and second-best results are bold and underlined, respectively.

Configuration	Camera	Robot	Lang.	Light	Backgr.	Noise	Layout	Total
Vanilla VLA	41.0	**61.2**	<u>86.5</u>	91.3	91.6	53.8	73.4	69.3
MEF-VLA (Full)	**61.2**	<u>55.7</u>	**87.4**	**95.6**	**96.6**	**82.4**	**79.3**	**78.2**
Action horizon (A1)	56.2	42.9	83.5	<u>94.2</u>	<u>96.0</u>	76.9	<u>76.5</u>	73.3
Event frame (A2)	<u>56.5</u>	46.4	84.7	93.1	95.6	<u>78.0</u>	74.0	<u>73.7</u>

C.2 Zero-Shot Results on LIBERO-Plus

Table 10 reports the results by perturbation category. The full model reaches 78.2% Total, improving over A1 and A2 by 4.9 and 4.5 percentage points, respectively, and outperforms both variants in all seven categories. Relative to the vanilla VLA, it improves six categories, including gains of 28.6 points in Noise and 20.2 points in Camera. Robot initial-state perturbations remain challenging (55.7% versus 61.2% for Vanilla VLA). This gap may reflect limited generalization of the learned motion prior to perturbed initial configurations.

D Event Reference Validation

MEF-VLA uses the next event as the temporal reference for remaining-motion prediction. We evaluate the alignment between these references and human-annotated interactions across the four LIBERO suites, and examine how this alignment varies with the event-detection parameters.

D.1 Annotation and Evaluation Protocol

Human annotations. We randomly sample two demonstrations per task, yielding 80 trajectories across 40 tasks. A single annotator marks a representative frame for each identified interaction, producing 200 event annotations. The annotations are subsequently reviewed for internal consistency. The same human annotations are used for all evaluated detector configurations.

Evaluation protocol. We match detected events to annotated interactions within each trajectory using a ± 1 s tolerance. Matching is one-to-one, maximizing the number of matched events and then minimizing their total absolute temporal difference. Kinematic events are evaluated using micro-averaged precision, recall, and F1. Interaction coverage additionally evaluates the complete reference set, including the terminal reference: each terminal frame can match at most one remaining annotation within the same tolerance. A terminal frame that is already a kinematic event is counted only once. Confidence intervals are estimated using 1,000 bootstrap resamples of trajectories, stratified by suite.

D.2 Temporal Alignment Results

Table 11 summarizes the results. The kinematic detector matches 183 of 200 annotated interactions, achieving 91.5% recall, 77.5% precision, and 83.9% F1. Incorporating the terminal reference covers 12 additional interactions, bringing the coverage of the event-reference set to 97.5% (95% CI: 95.4–99.5%). Coverage reaches 100.0% on Spatial and Object, 97.7% on Goal, and 94.7% on Long. Across the 195 matched reference–annotation pairs, the mean absolute temporal difference is 0.293 s.

The kinematic detector produces 236 events, of which 53 remain unmatched to the annotations; the complete reference set leaves five annotated interactions uncovered. Overall, the constructed references provide broad temporal coverage of annotated interactions across the sampled tasks. This

Table 11: Event-reference alignment with human-annotated interactions on LIBERO. Precision, recall, and F1 evaluate kinematic events within a ± 1 s tolerance. Coverage additionally includes matches provided by the terminal reference. Overall results are micro-averaged. All rates are percentages.

Suite	Annotated events	Kinematic events			Complete reference set
		Precision	Recall	F1	Coverage
Spatial	40	81.6	100.0	89.9	100.0
Object	40	77.6	95.0	85.4	100.0
Goal	44	77.1	84.1	80.4	97.7
Long	76	75.6	89.5	81.9	94.7
Overall	200	77.5	91.5	83.9	97.5

correspondence supports their use as interaction-aligned endpoints for remaining-motion supervision and complements the policy-level gains from event alignment reported in Sec. 4.4.

D.3 Sensitivity to Event-Detection Parameters

We examine the sensitivity of event alignment to the speed quantile q, minimum low-speed duration p, and smoothing radius w. Starting from the configuration used to construct the policy-training targets, $(q, p, w) = (0.20, 3, 2)$, we vary one parameter at a time while keeping the others fixed. All configurations are evaluated on the same 80 demonstrations using the annotation and matching protocol described above.

Table 12: Sensitivity to event-detection parameters on LIBERO. The first row gives the default configuration; subsequent rows vary one parameter at a time. Precision, recall, and F1 (%) evaluate kinematic events against the same human annotations within a ± 1 s matching tolerance.

q	p	w	Precision	Recall	F1
0.20	3	2	77.5	91.5	83.9
0.15	3	2	82.0	82.0	82.0
0.25	3	2	73.2	93.0	81.9
0.20	2	2	72.4	94.5	82.0
0.20	4	2	81.2	86.5	83.8
0.20	3	1	75.9	93.0	83.6
0.20	3	3	76.6	90.0	82.8

Table 12 shows that event-detection F1 remains within 2.0 percentage points of the default across all tested settings, ranging from 81.9% to 83.9%. Increasing q raises recall at the expense of precision, whereas increasing p yields the opposite trade-off. Varying the smoothing radius has a smaller effect, with F1 ranging from 82.8% to 83.9%.

E Real-World Evaluation and Failure Analysis

E.1 Data and Rollout Protocol

Data. Training configurations are summarized in Appendix A. Cup Stacking contains 75 one-stage and 225 two-stage demonstrations, with each stage specifying which cup to place over another. The workspace contains four differently colored cups with randomized positions.

Rollout protocol. The robot provides a head view, two wrist views, and two-arm proprioceptive state at 10 Hz; task-specific masks exclude inactive-arm inputs and actions. Evaluation instructions

specify the ordered operations for each task family and composition length. Both methods receive the same ordered instructions and initial configurations, sampled from the demonstration workspace distribution. Both methods use real-time chunking (RTC) (Black et al., 2025b) for all three task families, with a 16-step action chunk and a target execution horizon of 8 steps. A trial terminates upon task completion, an unrecoverable ordering error, a safety intervention, or a 160 s time limit.

E.2 METRICS

For each task family, length, and method, we conduct $M = 30$ trials. Instruction success rate is

$$\mathrm{SR}_N = \frac{1}{M} \sum_{i=1}^{M} \mathbf{1}[\text{trial } i \text{ completes all } N \text{ stages in order}] . \tag{13}$$

Figure 3 reports SR_N for each task family. For each N, the cross-task average gives equal weight to the three task families; with 30 trials per family, this is also the success rate pooled over 90 trials. Across Pill Sorting, Sandwich Making, and Cup Stacking, respectively, the success counts for MEF-VLA are $(27, 24, 29)$ at $N = 1$, $(20, 20, 28)$ at $N = 2$, and $(15, 13, 25)$ at $N = 3$. The corresponding counts for Vanilla VLA are $(23, 22, 28)$, $(16, 17, 23)$, and $(7, 7, 17)$. We compute cross-task averages and percentage-point gains from these counts before rounding. The average gains at $N = 1, 2, 3$ are 7.8, 13.3, and 24.4 percentage points, respectively.

E.3 FAILURE MODES

We analyze failures across the three real-world task families, covering 270 trials per method across all instruction lengths and the subset of 90 trials with held-out three-stage instructions.

Failure categories. We group failures into three categories: *timeout*, when the task remains incomplete at the 160 s time limit; *instruction-following error*, when execution deviates from the instructed operations or their order; and *collision*, involving unintended contact between the robot and its environment. Timeout cases mainly involve repeated unsuccessful manipulation attempts or failure to place the target object at the required location. Figure 5 illustrates the three categories in Sandwich Making, distinguishing incomplete execution from deviations from the instruction and unintended physical contact. Table 13 reports failure counts and category-specific failure rates across all evaluated trials in each group.



(a) Timeout (b) Instruction-following error (c) Collision

Figure 5: **Failure cases in Sandwich Making.** (a) The robot does not complete the operation within the time limit. (b) The robot performs an operation inconsistent with the instruction. (c) The robot makes unintended contact with the plate. Red crosses mark failures; the green check in (b) indicates the instructed action.

Table 13: Real-world failure modes. Entries report failure counts; parentheses give percentages of all evaluated trials within each group (270 trials per method for All; 90 for $N = 3$). All includes $N = 1, 2, 3$; the $N = 3$ rows are a subset of All.

Scope	Method	Timeout	Instruction-following error	Collision	Total failures
All	Vanilla VLA	54 (20.0%)	44 (16.3%)	12 (4.4%)	110 (40.7%)
	MEF-VLA	30 (11.1%)	31 (11.5%)	8 (3.0%)	69 (25.6%)
$N = 3$	Vanilla VLA	21 (23.3%)	35 (38.9%)	3 (3.3%)	59 (65.6%)
	MEF-VLA	12 (13.3%)	23 (25.6%)	2 (2.2%)	37 (41.1%)

Across all instruction lengths, MEF-VLA reduces the number of failed trials from 110 to 69, with fewer failures in each category (Table 13). The largest reduction is in timeouts, from 54 to 30, followed by instruction-following errors, from 44 to 31.

For held-out three-stage instructions, instruction-following errors are the largest failure category for both methods. MEF-VLA reduces the instruction-following error rate from 38.9% to 25.6%, accounting for 12 of the 22 fewer failed trials relative to Vanilla VLA. Within MEF-VLA, instruction-following errors occur in 8 of 180 trials at seen lengths and 23 of 90 trials at $N = 3$. Maintaining the instructed operations and their order across longer compositions therefore remains a key challenge alongside the observed generalization gains.