FOCUSWAM: ADAPTIVE MULTI-VIEW FUTURE PREDICTION FOR WORLD ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

World action models (WAMs) use future prediction to guide robot manipulation, but the views needed to anticipate an interaction vary during task execution. Global prediction can miss fine-grained gripper–object dynamics, while predicting all views at every step incurs additional computational cost even when local details contribute little to control. We introduce FocusWAM, a world action model that uses global future features to select when to predict wrist views. It first anticipates scene-level dynamics and determines whether to activate wrist-view prediction for the upcoming action chunk. When activated, the local stream predicts wrist-view dynamics conditioned on global features. The resulting local features refine the global future representation. We supervise the refined representation with a global future prediction objective and use the global and local features to condition action generation. FocusWAM achieves 55.0% average success on RoboCasa GR1 and 50.40% on the randomized split of RoboTwin 2.0 after training on clean demonstrations only, together with 76.50% across four real-world tasks. Across the clean and randomized RoboTwin evaluations, FocusWAM improves average success over always-on local prediction using the same checkpoint from 65.0% to 68.3% while achieving a 1.50× inference speedup.

1 INTRODUCTION

World action models (WAMs) learn visual dynamics to support robot action generation (Shen et al., 2025; Ye et al., 2026b). Recent approaches use predictive visual representations to condition robot actions (Hu et al., 2025; Ma et al., 2026). In multi-view manipulation, however, the information needed for control changes during task execution. Global views capture scene context, whereas wrist views reveal the gripper–object relationships needed for precise manipulation (Pan et al., 2026; Zhao et al., 2026). This raises a central design question for WAMs: how to adapt the predicted views to the interaction that the robot is about to perform.

Consider grasping an object and transporting it to a target. Wrist observations reveal gripper–object alignment before grasp closure, but the held object can subsequently obscure much of the wrist-camera view. During transport, the global view retains the object–target relationship, while the wrist view may add little information. Local details become valuable again during placement or insertion. The utility of wrist-view prediction thus depends on the upcoming interaction and camera visibility.

Predicting only the global view can miss fine-grained contact geometry. Predicting every view at every step incurs local computation even when the wrist view contributes little beyond the available global context. Fixed multi-view configurations, including RGB-future generation in Flex-π (Yan et al., 2026a) and joint inference in DreamWAM (Yuan et al., 2026a), predict the configured views at every planning step. Under substantial occlusion, additional local predictions may also contain uncertain details with limited relevance to control. This motivates selecting when to predict local dynamics while retaining global context throughout execution.

Selecting when to predict local views addresses only part of this challenge. Wrist observations reveal nearby geometry, but their limited field of view may hide the target or scene context needed to anticipate an interaction. Global predictive features can provide this context when local prediction is activated.

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.

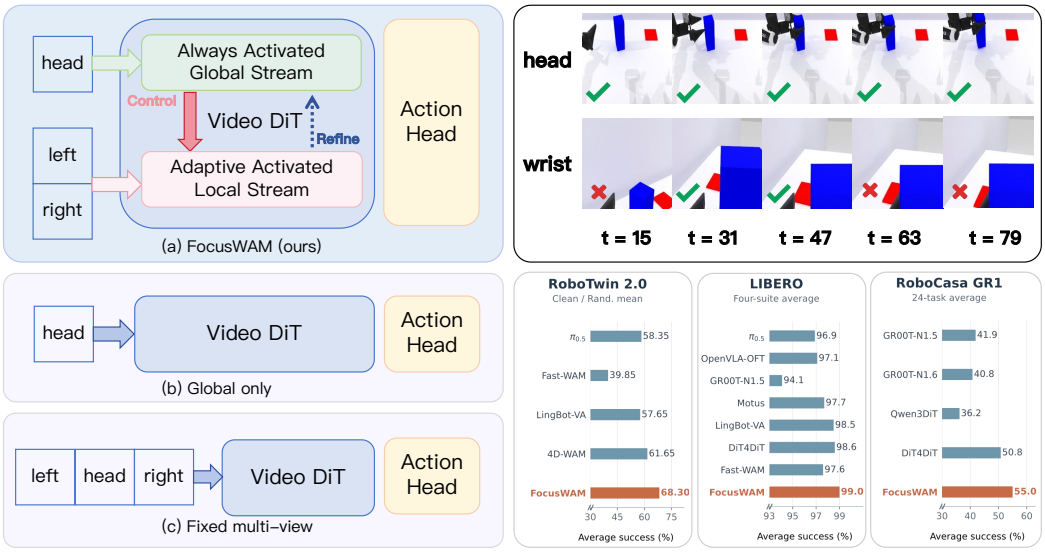

Figure 1: **Adaptive multi-view future prediction with FocusWAM.** Left: (a) FocusWAM retains global prediction, selectively invokes globally guided local prediction, and refines global features with local information within a shared Video DiT; (b) global-only and (c) fixed multi-view alternatives use a fixed set of prediction views. Upper right: a head- and wrist-view sequence illustrates selective local prediction at $t = 31$ and $t = 47$, while global prediction remains active throughout. Checkmarks indicate enabled prediction and crosses indicate omission. Lower right: average success on RoboTwin 2.0 (Clean/Rand. mean), LIBERO (four-suite mean), and RoboCasa GR1 (24-task mean); the success-rate axes are truncated.

A further challenge arises after local prediction: a small misalignment may determine whether an insertion succeeds or displaces the object. The initial global prediction, computed before processing wrist observations, cannot incorporate this evidence. Local prediction should therefore update the global future representation as well as inform action generation.

We introduce **FocusWAM** (Figure 1), a world action model for adaptive multi-view future prediction. From global future features, FocusWAM predicts whether fine-grained interaction will occur during the upcoming action chunk, determining whether to process wrist observations. When activated, the local stream predicts wrist dynamics conditioned on intermediate global features. Its predictions then refine the global representation under a global future prediction objective. The action model uses either global features alone or the refined global and local features. Local computation thus resolves fine-grained dynamics and revises the scene-level prediction used for control.

FocusWAM achieves 55.0% average success on RoboCasa GR1 and 76.50% across four real-world tasks. Training on clean RoboTwin 2.0 demonstrations yields 86.20% success in clean scenes and 50.40% under domain randomization. Using the same trained checkpoint, adaptive inference averages 68.3%, compared with 59.6% when local prediction is disabled and 65.0% when it is always enabled. It invokes the local stream for 10.3% of chunks and reduces latency by 33.5% relative to always-on prediction. Ablations assess both directions of cross-view interaction; additional LIBERO results appear in the appendix.

Our contributions are threefold:

- We introduce adaptive multi-view future prediction for robot manipulation, retaining global prediction at every step and selectively activating wrist-view prediction for fine-grained interaction.
- We condition local future prediction on intermediate global predictive features, providing scene-level context for wrist-view dynamics within a shared video backbone.
- We introduce local-to-global future refinement, using local predictive features to update the initial global representation under global future supervision before action generation.

2 Related Work

World action models. WAMs couple future prediction with action learning. VideoVLA (Shen et al., 2025) and DreamZero (Ye et al., 2026b) jointly model visual futures and actions, while VPP (Hu et al., 2025) conditions control on predictive video representations. DiT4DiT (Ma et al., 2026) extracts action-conditioning features in one video-backbone pass; Fast-WAM (Yuan et al., 2026b) further separates video co-training from test-time imagination. Latent approaches include task-centric action learning in UniVLA (Bu et al., 2025), joint-embedding transition prediction in JEPA-WAM (Lin et al., 2026), and stage-level prediction in StageWAM (Liu et al., 2026).

Other methods enrich predictive supervision. UP-VLA combines understanding and future prediction (Zhang et al., 2025a); DreamVLA predicts dynamic, spatial, and semantic information (Zhang et al., 2025b). Flex-π and DreamWAM incorporate semantic and geometric futures (Yan et al., 2026a; Yuan et al., 2026a), while Robust-WAM aligns action representations with future semantics (Yan et al., 2026b). SG-WAM introduces text-grounded and spatial guidance (He et al., 2026), and OmniVTA models future contact through visual and tactile streams (Zheng et al., 2026). These approaches study predictive content and supervision. FocusWAM studies when to predict local views and how their predictions update global features used for control.

Multi-view prediction and inference efficiency. World-to-Wrist conditions wrist-future modeling on multi-view observations and task instructions (Pan et al., 2026). GWM-VLA predicts target-view latents from geometry-aware multi-view context, using the wrist view as its prediction target (Zhao et al., 2026). Both motivate contextualized local prediction. FocusWAM additionally selects when to invoke wrist prediction and uses the resulting local features to refine the preceding global prediction. The RGB-future configuration of Flex-π and joint inference in DreamWAM instead predict their configured views at each planning step (Yan et al., 2026a; Yuan et al., 2026a).

Efficiency can also be improved along other dimensions. Vid2WAM distills video priors offline into a compact policy (Qiu et al., 2026). Adaptive-WAM selects backbone exits for driving (Ang et al., 2026), whereas TempoWAM adapts action execution and replanning (Ye et al., 2026a). FocusWAM keeps backbone depth fixed and adapts prediction views within each planning step. Its local invocation enables both global guidance of wrist prediction and local-to-global refinement before action generation.

3 Method

3.1 Overview

FocusWAM combines adaptive multi-view future prediction with global-to-local guidance and local-to-global refinement. The global stream runs at every planning step and determines whether to activate the local stream, which processes the wrist-camera observations. When activated, the local stream predicts wrist-view dynamics conditioned on global features. A refinement module then uses the local features to update the global representation before action generation.

Problem formulation. At planning step t, the available observations comprise a global image I_t^g, wrist images $\mathcal{I}_t^w = \{I_t^{w,j}\}_{j=1}^M$, a language instruction c, and proprioceptive state s_t. Here, M is the number of available wrist cameras. The policy produces an action chunk $A_t = [a_t, \ldots, a_{t+H-1}] \in \mathbb{R}^{H \times d_a}$ with $H = 16$. It always processes the global observation and selects whether to additionally process the wrist views for this chunk. Both streams model future dynamics over the same prediction horizon.

Model architecture. Figure 2 illustrates the overall architecture. Following DiT4DiT (Ma et al., 2026), a frozen VAE encodes observations and training future frames into latents. Video DiT learns future visual dynamics by predicting a velocity field from noisy future latents under a flow-matching objective (Lipman et al., 2023). For action conditioning, each active stream performs one Video DiT forward pass from Gaussian future noise, conditioned on current observations and the instruction. We call the extracted hidden representations future features and supply them directly to the Action DiT. VAE decoding is used for video visualization and image-quality evaluation. The streams share

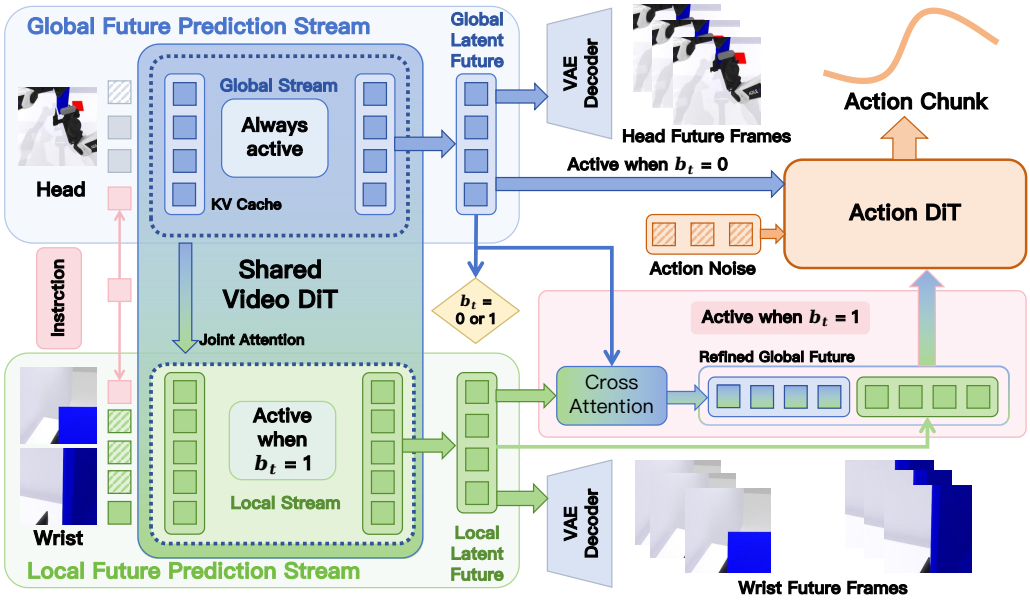

Figure 2: **Overview of FocusWAM.** Global future features determine local activation and guide wrist-view prediction through cached intermediate features. When activated, local predictive features refine the global representation before both streams condition the Action DiT; otherwise, actions use global features alone. Each active stream uses one Video DiT forward pass for action conditioning. The local stream accepts the available wrist views (two illustrated). VAE decoders render future videos for visualization and image-quality evaluation; the Action DiT reads hidden features before decoding.

the Video DiT parameters and a projection into the action conditioning space. The Action DiT conditions on proprioception and either global features alone or refined global features concatenated with local features.

3.2 Adaptive View Selection for Future Prediction

FocusWAM retains global prediction for every action chunk and activates local prediction for fine-grained interaction. The global stream provides features for action generation, local activation, and guidance of the local stream.

Given the global image and instruction, the global stream extracts

$$h_t^g = \mathcal{H}\big[V_\Theta\big(\epsilon_t^g, \tau_f \mid E_v(I_t^g), c\big)\big], \qquad F_t^g = P(h_t^g), \tag{1}$$

where E_v is the frozen VAE encoder, $\epsilon_t^g \sim \mathcal{N}(0, I)$ supplies noise at future latent positions, and τ_f is the fixed noise level used for action conditioning. $\mathcal{H}$ extracts hidden features from the Video DiT V_Θ, and P projects them into $F_t^g \in \mathbb{R}^{N_g \times d}$ for action generation and refinement.

Local activation. A lightweight head f_p uses the available global future features to select whether to activate local prediction:

$$q_p = \sigma(f_p(h_t^g)), \qquad b_t = \mathbb{I}[q_p > \eta], \tag{2}$$

where σ is the sigmoid function and η is the activation threshold. Binary labels indicate whether the upcoming chunk overlaps an annotated interaction segment, so q_p estimates interaction occurrence. This provides a supervised criterion for local activation; success and computation are evaluated jointly at a fixed threshold. At inference, $b_t = 1$ activates wrist processing and local prediction, conditioned on intermediate global features $\{G_t^\ell\}_{\ell \in \mathcal{L}_{gw}}$ retained from the same pass; otherwise, the Action DiT uses F_t^g directly. During training, label-conditioned sampling exposes each label class

to both action-conditioning paths, including global-only interaction chunks and local-enabled non-interaction chunks. Appendix B.1 defines the labels, and Appendix B.2 specifies sampling and loss aggregation.

3.3 Cross-View Prediction and Future Refinement

Global-to-local prediction supplies scene-level context for wrist-view dynamics. Local-to-global refinement subsequently incorporates the predicted local interaction into the initial global representation. The two components operate before action generation whenever the local stream is active.

Global-to-local prediction. Wrist observations capture local geometry, while the global prediction provides context for how the interaction may evolve. We therefore condition wrist-view prediction on the global features retained in Section 3.2. The wrist images are encoded, augmented with view embeddings, and concatenated into a local token sequence. The shared Video DiT extracts

$$h_t^w = \mathcal{H}\big[V_\Theta\big(\epsilon_t^w, \tau_f \mid E_v(\mathcal{I}_t^w), c, \{G_t^\ell\}_{\ell \in \mathcal{L}_{gw}}\big)\big], \tag{3}$$

where ϵ_t^w supplies the corresponding future latent noise. The shared projection produces $F_t^w = P(h_t^w) \in \mathbb{R}^{N_w \times d}$. A wrist-view future prediction objective trains this stream to represent local dynamics.

At selected backbone layers, local hidden features W_t^ℓ attend to the intermediate global features:

$$\widetilde{W}_t^\ell = W_t^\ell + \alpha_\ell \, \mathrm{CrossAttn}\big(\mathrm{LN}(W_t^\ell), \mathrm{LN}(G_t^\ell), G_t^\ell\big), \tag{4}$$

where LN denotes layer normalization and α_ℓ is a learned residual scale. Each local token can thus retrieve global context relevant to its prediction. The initial global pass provides this guidance without requiring an additional global backbone evaluation.

Local-to-global refinement. The global features used for activation and guidance are computed before the model processes wrist observations. Local prediction can reveal details, such as gripper–object alignment, that affect subsequent scene dynamics. We incorporate these details into the global representation through a residual update:

$$\widetilde{F}_t^g = F_t^g + R_\rho(F_t^g, F_t^w, \tau_f), \tag{5}$$

where the transformer R_ρ uses global features as queries and local features as keys and values. This update is applied once after local prediction. The Action DiT then attends to $[\widetilde{F}_t^g; F_t^w]$, combining a global representation informed by the local interaction with the local features themselves.

We train refinement with an auxiliary global future prediction objective. A flow-prediction head maps the refined global features to the velocity field in VAE latent space, encouraging the update to capture how local dynamics affect the scene. The original global prediction objective is retained to train the global stream for independent use. For the auxiliary objective, global features are evaluated at sampled noise levels and refined using local features from the observation-conditioned path; R_ρ receives the global noise level as a condition. The action path uses the fixed noise level τ_f for both streams. Appendix B specifies the training paths and flow targets. This supervision gives the cross-view update a predictive objective in addition to the action objective.

Training objectives. We jointly optimize action generation, future prediction, refinement, and local activation:

$$\mathcal{L} = \mathcal{L}_{\mathrm{act}} + \lambda_g \mathcal{L}_g + \lambda_w \mathcal{L}_w + \lambda_r \mathcal{L}_r + \lambda_p \mathcal{L}_p. \tag{6}$$

Here, $\mathcal{L}_{\mathrm{act}}$ is the action flow-matching loss; $\mathcal{L}_g$ and $\mathcal{L}_w$ supervise global and wrist future prediction; $\mathcal{L}_r$ supervises the refined global prediction; and $\mathcal{L}_p$ is binary cross-entropy for the interaction label. All three visual objectives regress flow velocities in VAE latent space, with no pixel-space reconstruction loss. The combined objective trains the enabled modules, while the VAE remains frozen. Activation is learned through $\mathcal{L}_p$; the discrete branch decision is not differentiated. Appendix B.3 gives implementation settings, and Appendix B.4 specifies the parameters held fixed in controlled ablations.

4 EXPERIMENTS

We evaluate FocusWAM on RoboCasa GR1, RoboTwin 2.0, and four real-world manipulation tasks. The experiments assess three aspects of adaptive multi-view prediction: control performance under changes in the visual environment, the contribution of each direction of cross-view interaction, and the success–latency trade-off of selective local prediction. All ablation studies use RoboTwin 2.0.

4.1 EXPERIMENTAL SETUP

Benchmarks and evaluation. We use the 24-task GR1 evaluation of RoboCasa (Nasiriany et al., 2024; NVIDIA et al., 2025b). Following Ma et al. (2026), we evaluate each task over 50 rollouts with a maximum horizon of 720 environment steps and average success across tasks. For RoboTwin 2.0 (Chen et al., 2026), we follow the Clean2Rand setting: models train on clean demonstrations and are evaluated under domain randomization. We train one multi-task policy on 50 clean demonstrations for each of the 50 Aloha-AgileX tasks, totaling 2,500 demonstrations, and evaluate 100 episodes per task in each of the clean and randomized settings. Clean and Rand. average success over the 50 tasks; Average is their mean. All RoboTwin control ablations follow this protocol. Task-wise RoboCasa results and additional LIBERO evaluation appear in Appendix A.

Baselines and controlled comparisons. On RoboCasa GR1, we compare with GR00T-N1.5, GR00T-N1.6, Qwen3DiT, and DiT4DiT using the results of Ma et al. (2026). RoboTwin baselines comprise $\pi_{0.5}$ (Black et al., 2025), Fast-WAM (Yuan et al., 2026b), LingBot-VA (Li et al., 2026), and 4D-WAM (Yang et al., 2026); task-level scores for the first, second, and fourth follow the official co-training leaderboard. External results retain their source camera, pretraining, and optimization settings. Our main-text ablations use one trained FocusWAM checkpoint with fixed parameters. We vary the cross-view connections or the local invocation rule at inference, keeping the observation interface, action horizon, and sampling configuration unchanged. Appendix B.4 specifies both comparisons.

Implementation and timing. Controlled comparisons use the same demonstrations, available cameras, RGB preprocessing, action horizon ($H = 16$), and four-step action sampling. We use $\eta = 0.5$ for local activation. Simulation latency is reported per action chunk on a single NVIDIA A100 GPU at batch size one; real-world deployment uses an NVIDIA RTX 4090. We time the model path from observation encoding through activation, prediction, refinement, and action generation, with task-level language embeddings cached. VAE video decoding is outside this timed path. Measurements use warmup and device synchronization. Binary interaction labels supervise local activation; Appendix B.1 describes their construction, and Appendix B.2 specifies branch sampling during training. Further implementation settings are given in Appendix B.3.

4.2 MAIN RESULTS

Table 1: Success rates (%, ↑) on the main simulation benchmarks. In (b), all policies train on clean demonstrations only. Rand. evaluates domain randomization; Avg. averages Clean and Rand.

(a) RoboCasa GR1

Method	Avg.
GR00T-N1.5	41.9
GR00T-N1.6	40.8
Qwen3DiT	36.2
DiT4DiT	50.8
FocusWAM	55.0

(b) RoboTwin 2.0 Clean2Rand

Method	Clean	Rand.	Avg.
$\pi_{0.5}$	70.70	46.00	58.35
Fast-WAM	77.80	1.90	39.85
LingBot-VA	80.70	34.60	57.65
4D-WAM	81.50	41.80	61.65
FocusWAM	86.20	50.40	68.30

RoboCasa GR1. Table 1(a) reports average success across the 24 tasks. FocusWAM achieves 55.0%, compared with 50.8% for DiT4DiT and 41.9% for GR00T-N1.5. This improves on the video-to-action architecture used as our starting point. Task-wise results in Table A1 show gains across both closing and object-placement tasks, supporting the applicability of adaptive multi-view prediction across different manipulation settings.

RoboTwin 2.0 Clean2Rand. Table 1(b) compares policies trained exclusively on clean demonstrations and evaluated on both Clean and Rand. scenes. FocusWAM reaches 86.20% in Clean scenes and 50.40% under domain randomization, averaging 68.30%. The Rand. score exceeds $\pi_{0.5}$ at 46.00% and 4D-WAM at 41.80%, while the overall average improves on 4D-WAM by 6.65 percentage points. This Clean2Rand protocol measures transfer to randomized scenes from a common clean training split. Task-level scores for methods with available breakdowns are given in Appendix A.6. The following ablations examine how cross-view interaction and selective prediction contribute to control under this protocol.

Success and inference efficiency. Table 3 compares three inference configurations of the same trained policy. FocusWAM reaches 68.3% average success, compared with 65.0% for always-on local prediction, while reducing latency from 267.5 to 177.8 ms. This corresponds to a 1.50× speedup and a 33.5% reduction in latency. Relative to global-only prediction, success increases by 7.7 and 9.7 percentage points on Clean and Rand., respectively, with 11.1 ms of additional latency. Adaptive invocation therefore provides a better success–latency trade-off than either fixed invocation rule for this checkpoint.

4.3 Ablation Studies

We first examine the two directions of cross-view interaction, then compare fixed and adaptive view prediction and assess visual prediction quality. The variants use the same checkpoint trained on clean demonstrations; we report Clean, Rand., and their mean. Model parameters remain fixed while we bypass selected interaction modules or change the local invocation rule at inference. Appendix A provides task-wise results and an activation-input comparison.

Table 2: Bidirectional interaction ablation on RoboTwin 2.0 Clean2Rand. Success rates are in %. All variants use the same trained checkpoint, retaining both streams and adaptive activation. Selected interaction modules are bypassed at inference. Guidance and refinement denote the two directions of interaction.

Variant	Guidance	Refinement	Clean	Rand.	Average
w/o both	–	–	81.0	43.6	62.3
w/o global guidance	–	Yes	83.4	46.6	65.0
w/o local-to-global refinement	Yes	–	84.0	46.4	65.2
FocusWAM	Yes	Yes	86.2	50.4	68.3

Global-to-local guidance. Every row in Table 2 uses the same trained FocusWAM checkpoint. We bypass the global-to-local attention residuals at inference while retaining the local stream and refinement (Appendix B.4). Removing global guidance reduces Clean success from 86.2% to 83.4% and Rand. success from 50.4% to 46.6%, lowering Average from 68.3% to 65.0%. Wrist observations, local prediction, and refinement remain available, so this comparison assesses global conditioning within the shared architecture. Guidance also raises local prediction PSNR from 19.48 to 21.12 dB and SSIM from 0.724 to 0.783 (Table 4). Global context therefore benefits local future modeling and control in both evaluation settings.

Local-to-global refinement. Bypassing refinement supplies the action model with the original global features concatenated with local features. The trained weights and activation rule remain unchanged. Clean and Rand. success fall to 84.0% and 46.4%, averaging 65.2%. Retaining the learned refinement update improves success by 2.2 and 4.0 percentage points over bypassing it. Its global predictions also improve in PSNR and SSIM (Table 4). This inference ablation measures the contribution of the learned update to the trained policy.

Combining the two directions. Removing both components yields 81.0% on Clean and 43.6% on Rand., averaging 62.3%, despite retaining both streams as action inputs. Either direction improves both scores, and combining them gives the highest success: 86.2% and 50.4%. The gains of 5.2 and 6.8 percentage points over the variant without either component support exchanging predictive information across views. Global guidance supplies context for local prediction, while refinement returns the resulting interaction details to the global representation.

Table 3: Inference-time view-prediction ablation on RoboTwin 2.0 Clean2Rand using the same trained checkpoint. We change only the local invocation rule; both local-enabled configurations retain guidance and refinement. Success and activation rates are in %; latency is per action chunk.

Variant	Local activation at inference	Clean	Rand.	Average	Active (%)	Latency (ms)
Global-only	Off	78.5	40.7	59.6	0.0	166.7
Always-on local	Always on	83.6	46.4	65.0	100.0	267.5
FocusWAM	Predicted	86.2	50.4	68.3	10.3	177.8

Adaptive view prediction. In Table 3, the global-only configuration sets $b_t = 0$, always-on local sets $b_t = 1$, and adaptive inference uses the predicted decision. All three configurations share the same parameters and training history. Always-on prediction improves over global-only by 5.1 percentage points on Clean and 5.7 on Rand., supporting the value of local information. FocusWAM raises success by a further 2.6 and 4.0 points while invoking the local stream for 10.3% of chunks. For this trained policy, selective invocation achieves higher success with substantially fewer local predictions than unconditional invocation. Appendix A.3 separately examines the activation input.

Future prediction quality. Table 4 evaluates the visual prediction objectives associated with the two interaction directions. Global guidance increases local future PSNR from 19.48 to 21.12 dB and SSIM from 0.724 to 0.783. Refinement increases global future PSNR from 21.06 to 22.58 dB and SSIM from 0.764 to 0.817. The first comparison measures whether scene context improves the wrist prediction; the second measures whether incorporating local features improves the global prediction. Both trends accompany higher control success in Table 2.

Table 4: Future prediction quality on RoboTwin 2.0. PSNR and SSIM compare VAE-decoded future frames with their targets. Each pair evaluates one direction of cross-view interaction.

Prediction target	Variant	PSNR (dB) ↑	SSIM ↑
Local future	w/o global guidance	19.48	0.724
Local future	FocusWAM	21.12	0.783
Global future	w/o local-to-global refinement	21.06	0.764
Global future	FocusWAM	22.58	0.817

These measurements assess decoded visual predictions, whereas the action model reads hidden features from a single forward pass at $\tau_f = 1$. Prediction quality and task success therefore characterize complementary uses of the shared video model. The paired comparisons show that global guidance improves local prediction and that local-to-global refinement improves global prediction, with both directions accompanied by higher control success. Together, the results establish the role of cross-view exchange within the proposed policy.

4.4 Real-World Evaluation

Platform and control. We use an AirbotPlay dual-arm robot with one head-mounted Intel RealSense D455 camera and one wrist-mounted 720p webcam. Their RGB images provide the global and local observations, respectively ($M = 1$), and are resized to 256×256 for the model. Following the default execution horizon in the official DiT4DiT real-world deployment code (Ma et al., 2026), we execute the first eight actions of each 16-action prediction before replanning. Execution is synchronous, with commands interpolated to 20 Hz (Appendix B.5). The reported latency measures the model path from observation encoding to action generation on an NVIDIA RTX 4090; Appendix B.5 defines the full synchronous execution cycle.

Tasks and protocol. We train on 50 robot demonstrations per task and evaluate 50 trials per task. Knife placement requires removing a fruit knife from a rack and releasing it on a cutting board. Sausage cutting requires using the knife to separate a sausage into pieces. Seasoning requires grasping a bottle and depositing seasoning onto sausage in a bowl. Stirring requires using a spatula to

redistribute cucumber pieces within a pot. These tasks concern physical outcomes involving placement, material separation, dispensing, and sustained tool–object contact. We report task success and the mean across 200 trials per method, assessing deployment on this platform after training on demonstrations of the evaluated tasks.

Table 5: Real-world success rates (%, ↑) and mean inference latency (↓) on an NVIDIA RTX 4090. Each task uses 50 training demonstrations and 50 evaluation trials.

Method	Knife placement	Sausage cutting	Seasoning	Stirring	Avg.	Latency (ms)
$\pi_{0.5}$	78.0	46.0	66.0	60.0	62.50	200.0
DiT4DiT	86.0	54.0	74.0	72.0	71.50	166.7
Fast-WAM	80.0	52.0	68.0	66.0	66.50	210.0
FocusWAM	92.0	58.0	82.0	74.0	76.50	181.5

Manipulation performance. Table 5 reports 76.50% average success for FocusWAM, compared with 71.50% for DiT4DiT, 66.50% for Fast-WAM, and 62.50% for $\pi_{0.5}$. FocusWAM completes 153 of 200 trials, compared with 143 for DiT4DiT. Its task-wise gains over DiT4DiT are 6, 4, 8, and 2 percentage points for knife placement, cutting, seasoning, and stirring. FocusWAM reaches 92.0% on knife placement, 82.0% on seasoning, and 74.0% on stirring, while cutting reaches 58.0%. Appendix A.8 provides the corresponding success counts.

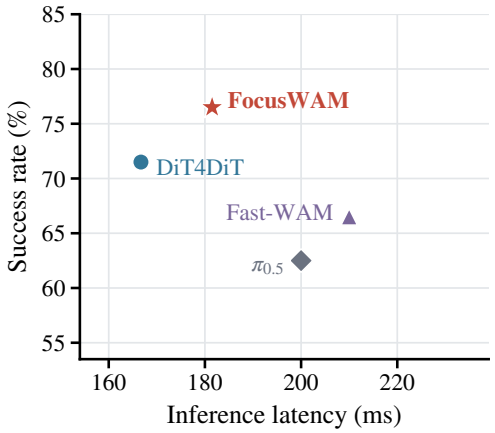

Figure 3: Real-world success and mean inference latency per action chunk on an NVIDIA RTX 4090.

Success–latency trade-off. Figure 3 places these results in the context of inference cost. FocusWAM improves average success over DiT4DiT by 5 percentage points with 14.8 ms of additional latency, from 166.7 to 181.5 ms. It also achieves higher observed success with lower latency than $\pi_{0.5}$ and Fast-WAM, reducing latency by 9.3% and 13.6%, respectively. Based on these point estimates, FocusWAM and DiT4DiT offer the two nondominated operating points: one favors task success and the other lower latency. The RoboTwin comparison in Table 3 evaluates adaptive and fixed invocation rules using the same trained checkpoint.

The four tasks cover both discrete object transfers and sustained tool use. The largest gain over DiT4DiT occurs in seasoning, where success rises from 74.0% to 82.0%. Gains also extend to knife placement, cutting, and stirring, spanning grasping, material separation, and contact-rich motion. Together with the controlled RoboTwin comparisons, these results support allocating local prediction to selected action chunks while using global guidance and local-to-global refinement to connect scene context with interaction details.

5 Conclusion

FocusWAM combines adaptive view prediction with global-to-local guidance and local-to-global refinement. Experiments on RoboCasa GR1, RoboTwin 2.0, LIBERO, and four real-world tasks show that this conditional exchange improves both task success and inference efficiency. On RoboTwin, FocusWAM achieves 68.3% average success with a 1.50× speedup over always-on prediction. Real-world evaluation reaches 76.5% average success across four manipulation tasks while retaining a 181.5 ms mean inference latency per action chunk.

Reproducibility Statement

We have made every effort to ensure that the results presented in this paper are reproducible. Section 3 describes the FocusWAM architecture and training objectives, including adaptive view selection, global-to-local guidance, and local-to-global refinement. Section 4.1 specifies the RoboCasa GR1 and RoboTwin 2.0 evaluation protocols, while Section 4.4 details the real-world tasks, robot platform, and camera setup.

Appendix B provides further details on interaction-label construction, branch sampling, loss aggregation, and future prediction and refinement during training. It also documents model configurations, optimization settings, training hardware, controlled evaluation protocols, and synchronous execution and timing. Appendix A reports task-wise results, additional evaluation on LIBERO, prediction-quality evaluation, and real-world success counts. The public RoboCasa, RoboTwin 2.0, and LIBERO benchmarks further support consistent evaluation. Together, these descriptions are intended to facilitate independent reproduction and verification of our experiments.

AI Use Statement

Large Language Models (LLMs) were used to assist with the writing and polishing of this manuscript. Specifically, we used an LLM to refine the language, improve readability, and enhance clarity across various sections of the paper. This assistance included sentence rephrasing, grammar checking, and improving the overall flow of the text.

The LLM was not involved in ideation, research methodology, or experimental design. All research concepts, ideas, and analyses were developed and conducted by the authors. The LLM's contributions were limited to improving the linguistic quality of the manuscript, with no involvement in the scientific content or data analysis.

The authors take full responsibility for the content of the manuscript, including any text generated or polished by the LLM. We have reviewed the LLM-assisted text to ensure that it complies with ethical guidelines and does not introduce plagiarism or scientific misconduct.

References

Sining Ang, Yuguang Yang, and Yan Wang. Adaptive-wam: Quality-guided early-exit planning from intermediate video-diffusion features. *arXiv preprint arXiv:2608.06008*, 2026.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *Proceedings of the Conference on Robot Learning*, 2025.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.014.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan-ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. RoboTwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

Junjie He, Junfeng Li, Zhide Zhong, Haodong Yan, Ruixin Li, Yangyang Zheng, Jiaguan Zhu, Tianran Zhang, Yuqiao Du, Wen Chen, Shunbo Zhou, and Haoang Li. SG-WAM: Text-grounded and spatial-aware semantic guidance for world-action models. *arXiv preprint arXiv:2608.08839*, 2026. doi: 10.48550/arXiv.2608.08839.

Yucheng Hu, Yanjiang Guo, Pengchao Wang, Xiaoyu Chen, Yen-Jen Wang, Jianke Zhang, Koushil Sreenath, Chaochao Lu, and Jianyu Chen. Video prediction policy: A generalist robot policy with predictive visual representations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 24328–24346, 2025.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.017.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

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*, 2026. doi: 10.48550/arXiv.2608.09381.

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*, volume 36, pp. 44776–44791, 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*, 2026. doi: 10.48550/arXiv.2608.10780.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of household tasks for generalist robots. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.050.

NVIDIA, Arslan Ali, Junjie Bai, Maciej Bala, Yogesh Balaji, Aaron Blakeman, Tiffany Cai, Jiaxin Cao, Tianshi Cao, Elizabeth Cha, Yu-Wei Chao, Prithvijit Chattopadhyay, Mike Chen, Yongxin Chen, Yu Chen, Shuai Cheng, Yin Cui, Jenna Diamond, Yifan Ding, Jiaojiao Fan, Linxi Fan, Liang Feng, Francesco Ferroni, Sanja Fidler, Xiao Fu, Ruiyuan Gao, Yunhao Ge, Jinwei Gu, Aryaman Gupta, Siddharth Gururani, Imad El Hanafi, Ali Hassani, Zekun Hao, Jacob Huffman, Joel Jang, Pooya Jannaty, Jan Kautz, Grace Lam, Xuan Li, Zhaoshuo Li, Maosheng Liao, Chen-Hsuan Lin, Tsung-Yi Lin, Yen-Chen Lin, Huan Ling, Ming-Yu Liu, Xian Liu, Yifan Lu, Alice Luo, Qianli Ma, Hanzi Mao, Kaichun Mo, Seungjun Nah, Yashraj Narang, Abhijeet Panaskar, Lindsey Pavao, Trung Pham, Morteza Ramezanali, Fitsum Reda, Scott Reed, Xuanchi Ren, Haonan Shao, Yue Shen, Stella Shi, Shuran Song, Bartosz Stefaniak, Shangkun Sun, Shitao Tang, Sameena Tasmeen, Lyne Tchapmi, Wei-Cheng Tseng, Jibin Varghese, Andrew Z. Wang, Hao Wang, Haoxiang Wang, Heng Wang, Ting-Chun Wang, Fangyin Wei, Jiashu Xu, Dinghao Yang, Xiaodong Yang, Haotian Ye, Seonghyeon Ye, Xiaohui Zeng, Jing Zhang, Qinsheng Zhang, Kaiwen Zheng, Andrew Zhu, and Yuke Zhu. World Simulation with Video Foundation Models for Physical AI. *arXiv preprint arXiv:2511.00062*, 2025a. doi: 10.48550/arXiv.2511.00062.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu,

Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025b. doi: 10.48550/arXiv.2503.14734.

Yuhao Pan, Haosong Peng, Zhengshen Zhang, Zhengyang Yan, Yalun Dai, Fushuo Huo, Chujie Wang, Tianyu Qi, Xiucheng Wang, Nan Cheng, et al. World-to-wrist: Task-conditioned future wrist modeling for fine-grained robot manipulation. *arXiv preprint arXiv:2608.05369*, 2026.

Chenhao Qiu, Ruixiang Wang, Runyi Zhao, Sixu Lin, Songen Gu, Shufeng Nan, Guiliang Liu, Kui Jia, Yanwei Fu, and Simo Wu. Vid2WAM: Distilling video diffusion priors into world action models. *arXiv preprint arXiv:2608.08558*, 2026. doi: 10.48550/arXiv.2608.08558.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. VideoVLA: Video generators can be generalizable robot manipulators. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Ge Yan, Jinghao Liu, Yuzhi Fan, Lei Cai, Minwen Liao, Jesse Zhang, and Dieter Fox. Flex-π: A Multi-Stream World-Action Model with Compute Flexibility. *arXiv preprint arXiv:2608.10860*, 2026a.

Haodong Yan, Junfeng Li, Junjie He, Zhide Zhong, MingMing Yu, Wenxuan Song, Jiaguan Zhu, Yangyang Zheng, Yuqiao Du, Jiadi You, Yingjie Cai, Xu Yan, Guanyi Zhao, Bingbing Liu, and Haoang Li. Robust-WAM: Bridging generative pretraining and semantic foresight in world-action models. *arXiv preprint arXiv:2608.05903*, 2026b. doi: 10.48550/arXiv.2608.05903.

Lishan Yang, Wenxuan Song, Xi Wang, Pingyue Sheng, Zheng Fang, Ziyang Zhou, Junjie He, Haodong Yan, Jiayi Chen, Nan Sun, et al. 4d-wam: Infusing spatiotemporal awareness into world action models through trajectory fields. *arXiv preprint arXiv:2608.08023*, 2026.

Feng Ye, Yiming Zhao, Yong Yu, Hongxu Zhou, Yong Pan, Yuan Xue, Peng Jia, and Chuanmin Jia. Rethink before you execute: Adaptive execution for world action models. *arXiv preprint arXiv:2608.09492*, 2026a. doi: 10.48550/arXiv.2608.09492.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026b. doi: 10.48550/arXiv.2602.15922.

Shanglin Yuan, Weiheng Zhao, Xin Shi, Haoyi Jiang, Xianda Guo, Liu Liu, Wenyu Liu, Wei Sui, and Xinggang Wang. Dreamwam: Beyond rgb future prediction for world action models. *arXiv preprint arXiv:2608.04996*, 2026a.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination? *arXiv preprint arXiv:2603.16666*, 2026b.

Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. UP-VLA: A unified understanding and prediction model for embodied agent. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74911–74922, 2025a.

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, 2025b.

Yanping Zhao, Hang Yu, Yiwei Wang, Chen Ye, Siyu Tian, Di Zhang, Qingjun Wang, Qian Chen, Junqiao Zhao, Chen Ye, and Guang Chen. GWM-VLA: Geometry-aware latent world modeling for vision-language-action learning. *arXiv preprint arXiv:2608.07619*, 2026. doi: 10.48550/arXiv.2608.07619.

Yuhang Zheng, Songen Gu, Yupeng Zheng, Weize Li, Yujie Zang, Shuai Tian, Xiang Li, Ce Hao, Chen Gao, Si Liu, Haoran Li, Yilun Chen, Shuicheng Yan, and Wenchao Ding. OmniVTA: Visuotactile world modeling for contact-rich robotic manipulation. *arXiv preprint arXiv:2603.19201*, 2026. doi: 10.48550/arXiv.2603.19201.

A Additional Experimental Results

We report task-wise RoboCasa GR1 and RoboTwin 2.0 results, additional evaluation on LIBERO, an activation-input comparison, and gate classification metrics across thresholds. We also detail the prediction-quality evaluation summarized in Table 4. RoboTwin variants train on the clean split; control comparisons report Clean, Rand., and their Average as defined in Section 4.1.

A.1 Task-Wise Results on RoboCasa GR1

Table A1 expands the aggregate results in Table 1(a). FocusWAM achieves 55.0% average success, compared with 50.8% for DiT4DiT. It improves on DiT4DiT in 22 of 24 tasks. Among closing tasks, Can to drawer improves from 74.0% to 82.0%. Among placement tasks, Cutting board to tiered basket improves from 50.0% to 58.0%, and Plate to pan from 68.0% to 74.0%. The gains span both closing and placement operations across different objects and receptacles.

Each task uses the evaluation protocol in Section 4.1: 50 rollouts with a maximum horizon of 720 environment steps. The table reports task-level success rates and their average.

Table A1: Task-wise success rates (%, ↑) on RoboCasa GR1. Baseline results are quoted from Table 2 of Ma et al. (2026); Qwen3DiT and DiT4DiT use no action data outside this benchmark. Avg. is recomputed from the 24 task scores and rounded to one decimal place; (close) denotes a closing operation.

Task	GR00T N1.5	GR00T N1.6	Qwen3DiT	DiT4DiT	FocusWAM
Bottle to cabinet (close)	40.0	36.0	50.0	48.0	48.0
Can to drawer (close)	56.0	28.0	48.0	74.0	82.0
Cup to drawer (close)	50.0	12.0	42.0	52.0	58.0
Milk to microwave (close)	52.0	20.0	38.0	50.0	56.0
Potato to microwave (close)	22.0	28.0	18.0	36.0	38.0
Wine to cabinet (close)	44.0	18.0	28.0	42.0	44.0
Cutting board to basket	46.0	42.0	42.0	52.0	56.0
Cutting board to cardboard box	44.0	40.0	30.0	48.0	52.0
Cutting board to pan	58.0	62.0	50.0	76.0	78.0
Cutting board to pot	48.0	60.0	44.0	62.0	68.0
Cutting board to tiered basket	28.0	48.0	36.0	50.0	58.0
Placemat to basket	32.0	42.0	14.0	50.0	54.0
Placemat to bowl	52.0	34.0	28.0	56.0	62.0
Placemat to plate	42.0	42.0	40.0	32.0	36.0
Placemat to tiered shelf	26.0	24.0	30.0	18.0	22.0
Plate to bowl	38.0	48.0	36.0	56.0	60.0
Plate to cardboard box	40.0	44.0	36.0	58.0	62.0
Plate to pan	56.0	48.0	34.0	68.0	74.0
Plate to plate	50.0	66.0	44.0	58.0	56.0
Tray to cardboard box	36.0	42.0	48.0	38.0	42.0
Tray to plate	54.0	52.0	44.0	56.0	60.0
Tray to pot	36.0	64.0	34.0	54.0	58.0
Tray to tiered basket	34.0	42.0	36.0	46.0	52.0
Tray to tiered shelf	22.0	38.0	18.0	38.0	44.0
Avg.	41.9	40.8	36.2	50.8	55.0

A.2 Additional Benchmark: LIBERO

We evaluate FocusWAM on the Spatial, Object, Goal, and Long suites of LIBERO (Liu et al., 2023), each containing 10 tasks. For FocusWAM, each task is evaluated over 50 episodes, for a total of 2,000 episodes. Suite scores average task success rates, and the overall score averages the four suites. Table A2 additionally includes OpenVLA-OFT (Kim et al., 2025) and Motus (Bi et al., 2026) alongside the baselines introduced in Section 4.1.

Table A2 reports 99.0% average success for FocusWAM, compared with 98.6% for DiT4DiT and 97.6% for Fast-WAM. FocusWAM reaches 99.4% on Spatial and 98.8% on Goal, the highest scores

Table A2: Success rates (%, ↑) on the four LIBERO suites. Avg. is the mean across suites.

Method	Spatial	Object	Goal	Long	Avg.
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1
GR00T-N1.5	96.2	94.0	96.0	90.0	94.1
Motus	96.8	99.8	96.6	97.6	97.7
LingBot-VA	98.5	99.6	97.2	98.5	98.5
DiT4DiT	98.4	99.6	98.6	97.6	98.6
Fast-WAM	98.2	100.0	97.0	95.2	97.6
FocusWAM	99.4	99.6	98.8	98.2	99.0

among the listed methods for these suites. On Long, it achieves 98.2%, exceeding DiT4DiT at 97.6% and Fast-WAM at 95.2%, while LingBot-VA reaches 98.5%. These results complement the main benchmarks by evaluating spatial variation, object variation, goal changes, and longer task sequences.

A.3 Gate Input Comparison on RoboTwin 2.0

We compare current global image features $E_v(I_t^g)$ with global future features h_t^g as inputs to the local-activation head. The current-image variant reuses the frozen visual VAE encoder in FocusWAM and reads the global observation features before Video DiT processing. The future-feature variant uses hidden features from the global prediction pass. Each input is pooled and projected to the same classifier input dimension. Both variants use the same frozen control policy, classifier architecture, training data, interaction labels, and optimization budget; only the input projection and activation head are trained. No additional image encoder is introduced.

Table A3: Activation-input ablation on RoboTwin 2.0 Clean2Rand. Success rates are in %; Average is the mean of Clean and Rand. Active is the local invocation rate, and recall is measured against interaction labels. Both heads use $\eta = 0.5$.

Gate input	Active (%)	Clean	Rand.	Average	Latency (ms)	Recall (%)
Current global image features	36.8	77.6	44.4	61.0	204.3	70.4
Global future features	10.3	86.2	50.4	68.3	177.8	94.2

Table A3 compares two activation interfaces within FocusWAM. Global Video DiT features yield 68.3% average success, compared with 61.0% for VAE observation features, with fewer local invocations (10.3% versus 36.8%) and lower mean latency (177.8 versus 204.3 ms). Recall against the interaction labels is 94.2% versus 70.4%. The Video DiT interface includes task-adapted backbone processing and language conditioning, whereas the VAE interface reads visual encoding alone. The global Video DiT interface provides higher task success with fewer local invocations in this comparison.

A.4 Local Future Prediction Quality

We examine whether global guidance improves local future prediction. Table 4 reports PSNR and SSIM for FocusWAM and the variant without global guidance on RoboTwin 2.0. These metrics compare VAE-decoded video predictions with target images: PSNR measures pixel-level fidelity, and SSIM measures structural similarity. They assess the learned visual dynamics, whereas action inference uses hidden features at $\tau_f = 1$ without decoding. The control ablations in Table 2 evaluate the corresponding modules through task success. The visual metrics characterize future modeling, and the control ablations measure the contribution of the corresponding modules to task success.

Global guidance raises PSNR by 1.64 dB and SSIM by 0.059. In the corresponding control comparison, it raises Clean success from 83.4% to 86.2% and Rand. success from 46.6% to 50.4%,

averaging 68.3% instead of 65.0%. Global context benefits both local visual prediction and action generation in these comparisons.

A.5 Global Future Prediction Quality

We next examine the global prediction after incorporating local information. Table 4 compares FocusWAM with the variant that omits local-to-global refinement. The latter retains local prediction and supplies both streams to the action model, so the control comparison tests refinement beyond the availability of local features.

Refinement increases global PSNR by 1.52 dB and SSIM by 0.053. In the corresponding control comparison, Clean success increases from 84.0% to 86.2% and Rand. success from 46.4% to 50.4%, raising Average from 65.2% to 68.3%. These measurements assess the supervised refinement module as a whole. The gains in visual prediction quality complement the task-success improvements obtained with supervised refinement.

A.6 Task-Wise Results on RoboTwin 2.0

Table A4 presents Clean and Rand. success rates for the 50 RoboTwin 2.0 tasks. The baseline task-level scores are taken from the RoboTwin 2.0 co-training leaderboard. LingBot-VA reports aggregate results but does not provide a task-level breakdown, and is therefore omitted from this table. The FocusWAM task-level scores average to 86.20% on Clean and 50.40% on Rand., matching Table 1(b).

Table A4: Task-level success rates (%) on RoboTwin 2.0 Clean2Rand. C and R denote Clean and Rand. evaluations. The average row follows the displayed benchmark aggregates.

	$\pi_{0.5}$		Fast-WAM		4D-WAM		FocusWAM	
Task	C	R	C	R	C	R	C	R
Adjust Bottle	99	77	99	0	80	36	92	70
Beat Block Hammer	93	23	82	0	92	28	95	35
Blocks Ranking RGB	72	44	81	0	88	10	90	19
Blocks Ranking Size	45	21	52	0	54	8	58	14
Click Alarmclock	65	50	99	60	100	70	98	74
Click Bell	31	35	95	7	98	77	93	79
Dump Bin Bigbin	92	84	94	0	86	57	93	65
Grab Roller	99	97	100	2	100	68	98	89
Handover Block	30	12	66	0	79	25	85	31
Handover Mic	98	6	98	0	85	13	92	24
Hanging Mug	17	14	14	0	70	10	63	12
Lift Pot	98	35	91	0	88	9	94	17
Move Can Pot	60	10	96	0	94	17	93	27
Move Pillbottle Pad	46	32	92	0	89	22	91	32
Move Playingcard Away	84	65	94	0	82	35	89	50
Move Stapler Pad	20	18	33	0	43	12	52	20
Open Laptop	93	68	84	0	85	82	89	81
Open Microwave	86	59	17	0	68	12	79	33
Pick Diverse Bottles	66	42	72	0	77	44	81	49
Pick Dual Bottles	72	57	87	1	92	67	95	71
Place A2B Left	71	49	69	0	71	19	81	36
Place A2B Right	68	39	60	1	62	42	76	45
Place Bread Basket	63	60	87	0	87	77	91	77
Place Bread Skillet	76	47	85	0	82	69	85	70
Place Burger Fries	82	86	97	0	86	72	90	76
Place Can Basket	56	10	77	0	83	51	89	55
Place Cans Plasticbox	31	73	98	0	94	48	94	55
Place Container Plate	92	72	100	0	93	82	97	84
Place Dual Shoes	70	45	67	0	66	19	73	33
Place Empty Cup	96	80	90	0	99	70	98	78
Place Fan	66	26	59	0	48	17	63	27

Continued on next page

Table A4 (continued)

Task	$\pi_{0.5}$ C	$\pi_{0.5}$ R	Fast-WAM C	Fast-WAM R	4D-WAM C	4D-WAM R	FocusWAM C	FocusWAM R
Place Mouse Pad	34	31	55	0	54	9	67	22
Place Object Basket	70	29	82	0	75	40	80	45
Place Object Scale	78	38	83	0	79	24	84	35
Place Object Stand	85	47	94	0	87	42	88	50
Place Phone Stand	65	37	85	0	84	21	88	31
Place Shoe	82	50	87	0	83	52	89	57
Press Stapler	79	63	90	3	76	61	84	67
Put Bottles Dustbin	78	50	55	0	68	19	85	33
Put Object Cabinet	39	22	42	0	69	47	76	51
Rotate QRcode	90	21	81	0	79	67	87	72
Scan Object	49	40	68	0	73	53	81	57
Shake Bottle Horizontally	99	100	100	9	98	70	96	85
Shake Bottle	100	100	100	10	98	72	98	85
Stack Blocks Three	77	25	89	0	95	82	97	83
Stack Blocks Two	91	43	99	0	100	34	98	52
Stack Bowls Three	81	52	80	0	82	27	88	42
Stack Bowls Two	94	70	94	0	94	81	96	84
Stamp Seal	53	21	30	0	74	10	82	21
Turn Switch	52	25	39	3	86	11	89	20
Average	70.70	46.00	77.80	1.90	81.50	41.80	86.20	50.40

A.7 Gate Threshold Analysis

Table A5 reports interaction classification on RoboTwin 2.0 across activation thresholds. Precision, recall, and F1 measure agreement with interaction labels; Active reports the fraction of chunks selected for local prediction. These metrics characterize the interaction proxy. Closed-loop success and latency are evaluated at $\eta = 0.5$ in Table 3.

Table A5: Gate threshold analysis on RoboTwin 2.0. All values except η are percentages. Active denotes the local invocation rate; precision, recall, and F1 are computed with respect to interaction labels.

η	Active ↓	Precision ↑	Recall ↑	F1 ↑
0.2	11.95	83.18	99.40	90.57
0.3	11.44	86.36	98.80	92.16
0.4	10.88	89.06	96.90	92.82
0.5	10.30	91.46	94.20	92.81
0.6	9.30	94.62	88.00	91.19
0.7	7.89	95.06	75.00	83.85
0.8	6.11	98.20	60.00	74.49

Increasing η reduces local invocation and raises precision at the cost of interaction recall. At $\eta = 0.5$, the invocation rate is 10.30%, with 91.46% precision, 94.20% recall, and 92.81% F1. The highest reported F1 is 92.82% at $\eta = 0.4$, only 0.01 points higher, while the invocation rate increases to 10.88%. Lowering the threshold to 0.3 increases recall to 98.80% but reduces precision to 86.36% and raises invocation to 11.44%. Raising it to 0.7 lowers invocation to 7.89%, while recall falls to 75.00%. The selected threshold therefore retains high interaction recall with fewer local invocations than the lower-threshold settings, while achieving nearly the same classification F1 as $\eta = 0.4$.

A.8 Success Counts

Real-world trials. Table A6 gives the success counts underlying Table 5. Each method is evaluated in 50 trials per task, for 200 trials across the four tasks.

Table A6: Real-world success counts. Each task uses 50 evaluation trials.

Method	Knife placement	Sausage cutting	Seasoning	Stirring
$\pi_{0.5}$	39/50	23/50	33/50	30/50
DiT4DiT	43/50	27/50	37/50	36/50
Fast-WAM	40/50	26/50	34/50	33/50
FocusWAM	46/50	29/50	41/50	37/50

B Interaction Supervision and Training Details

B.1 Interaction Annotation

Annotation target. We annotate intervals of fine-grained manipulation, including grasp and release transitions, local alignment, insertion, and sustained tool–object interaction. The state-action and visual grounding in World-to-Wrist (Pan et al., 2026) motivates the annotation procedure. The verified temporal intervals define an interaction-based proxy target for binary activation supervision.

Trajectory preprocessing. For each arm, annotation uses measured end-effector position $p(u)$ in meters, orientation $R(u) \in \mathrm{SO}(3)$, and gripper aperture $g(u)$ at timestamp u. Joint-state datasets are converted to end-effector poses using forward kinematics and the robot model. Aperture is normalized to $[0, 1]$ using the calibrated fully closed and fully open positions, rather than the extrema of each demonstration. All poses use a fixed robot-base frame. For annotation only, trajectories are resampled at 20 Hz by linear interpolation of position and aperture and spherical interpolation of orientation. A three-sample median filter is applied to each position coordinate and to aperture, with endpoint replication. Orientation distances use the shortest rotation angle. This preprocessing does not resample the observations or actions used to train the policy.

Candidate intervals. At each annotation timestamp u_k, we consider a centered 0.4 s window, clipped at episode boundaries. A gripper event is proposed when the aperture range within the window is at least 0.10. For motion candidates, we measure translational and angular path lengths within the window, together with the maximum position and orientation deviations from the central pose. A window qualifies as localized motion when the path length is at least 2 mm or 2°, while every pose remains within 4 cm and 30° of the central pose. Thus, a candidate requires observable movement within a bounded region, rather than low velocity alone. A motion candidate must persist for at least four consecutive annotation samples; gripper events have no minimum-duration requirement. Each consecutive run of qualifying samples defines one candidate interval. The settings are summarized in Table A7.

Table A7: Settings for interaction annotation and branch sampling. Temporal annotation parameters are expressed in seconds.

Setting	Value
Annotation sampling rate	20 Hz
Median-filter width for position and aperture	3 samples
Centered candidate window	0.4 s
Minimum normalized aperture range	0.10
Minimum translational or angular path length	2 mm or 2°
Maximum position and orientation deviation	4 cm and 30°
Minimum motion-candidate duration	4 samples (0.2 s)
Candidate padding on each side	0.2 s
Maximum gap for joining intervals	0.2 s
Local training probability for positive chunks	0.90
Local training probability for negative chunks	0.10
Inference activation threshold η	0.50

Visual verification and temporal boundaries. Candidate intervals are extended by 0.2 s on each side, clipped to the episode, and joined when they overlap or their intervening gap is at most 0.2 s. An

annotator reviews the synchronized global and wrist video with the task instruction and trajectory traces. The start is the first frame showing local alignment or the onset of the relevant gripper–object or tool–object interaction; the end is the final frame before the interaction ceases or broad transport resumes. Padding provides review context and does not automatically become part of the final interval. Free-space transport, idle pauses, and sensor jitter are excluded. Cutting, stirring, dispensing, and contact-guided motion remain positive throughout the interaction, including periods with unchanged gripper aperture. The complete trajectory is also reviewed to add interactions missed by the motion rules. Ambiguous boundaries are resolved by a second annotator before labels are finalized.

After verification, overlapping intervals are united. A gap of at most 0.2 s is filled only when the same interaction continues on both sides and the intervening video shows a brief pause, rather than transport or a different operation. For bimanual trajectories, intervals are first annotated per arm and then combined by temporal union, since the policy activates the local stream as a whole. Final labels are stored as timestamped intervals with the active arm and interaction type. The annotation configuration and the finalized interval files together define the supervision. The motion rules generate candidates; visual verification determines the semantic boundaries.

From intervals to chunk labels. Let $\mathcal{S} = \bigcup_j [s_j, e_j)$ be the union of verified intervals in an episode. If action a_t begins at timestamp u_t, an H-action chunk covers $[u_t, u_{t+H})$. Its target is

$$y_t = \mathbb{I}\big[\mathrm{duration}\big([u_t, u_{t+H}) \cap \mathcal{S}\big) > 0\big]\,. \tag{7}$$

Labels are computed at the original action timestamps with $H = 16$. No additional temporal dilation is applied after the chunk-overlap operation: a chunk that will reach an interaction is already positive before that interaction begins. Only starts with a complete action chunk and the required future targets are eligible for training. Episode-level train/validation splits are fixed before chunk extraction, and the annotation settings are shared across tasks. Future demonstration frames support offline annotation and auxiliary prediction targets; they are not observation inputs at inference.

The positive-label fraction is computed over eligible training chunks after this conversion, separately for each dataset. It differs from the fraction of interaction frames and from the activation rate of a trained policy. The 10.3% rate in Table 3 is the fraction of evaluation chunks that activate local prediction on RoboTwin 2.0. Interaction labels provide a proxy for local activation: the classifier estimates overlap between the upcoming action chunk and an interaction interval. Recall in Table A3 is measured against this annotation target. All reported adaptive control results use $\eta = 0.5$.

B.2 Branch Selection and Loss Aggregation

Label-guided branch sampling. For adaptive FocusWAM training, minibatches sample uniformly from eligible chunk starts without class balancing. The global stream runs for every example. For each sampled chunk, an independent Bernoulli variable m_t determines whether to execute the local stream and refinement:

$$m_t \sim \mathrm{Bernoulli}(\rho_{y_t}), \qquad \rho_1 = 0.90, \quad \rho_0 = 0.10. \tag{8}$$

These probabilities are fixed throughout training. Branch selection uses the offline label and a random draw, rather than the predicted gate decision or a curriculum. Most positive examples train local prediction and refinement. Positive examples with $m_t = 0$ also train the global-only action path, while negative examples with $m_t = 1$ expose the local path to non-interaction states. The gate retains target y_t regardless of the sampled branch; a random activation is never relabeled as an interaction.

When $m_t = 1$, the action model receives $[\widetilde{F}_t^g; F_t^w]$, and both the local video objective and the refinement objective are evaluated. When $m_t = 0$, wrist encoding, local video passes, and refinement are skipped, and actions are conditioned on F_t^g. The global video objective and gate supervision are evaluated in both cases. At inference, m_t is replaced by the learned decision $b_t = \mathbb{I}[q_p > 0.5]$; neither annotation nor random branch sampling is used. Training therefore uses labels to choose branch availability, while inference uses the predicted gate. The contents of the action-conditioning features are computed from current observations, instruction, and noise in both settings.

Training and inference branch distributions. For training positive-label fraction π_{tr}, the expected local activation rate is $0.9\pi_{\mathrm{tr}} + 0.1(1 - \pi_{\mathrm{tr}})$. At inference, let r be interaction recall, f the false-positive rate, and π_{eval} the positive-label fraction over visited chunks. The activation rate is $r\pi_{\mathrm{eval}} + f(1 - \pi_{\mathrm{eval}})$. Thus, activation frequency can change with both classification errors and the states visited by the policy. Sampling both branches within each label class trains both action-conditioning paths. Evaluation uses the learned gate and measures the resulting closed-loop policy, including its activation decisions.

Loss normalization. For a minibatch $\mathcal{B}$ of size N, let $\mathcal{B}_w = \{t \in \mathcal{B} : m_t = 1\}$ and $N_w = |\mathcal{B}_w|$. Denote per-example losses by $\ell_{\mathrm{act},t}$, $\ell_{g,t}$, $\ell_{w,t}$, $\ell_{r,t}$, and $\ell_{p,t}$. Action losses are averaged over valid action steps and dimensions; video and refinement losses are averaged over valid latent elements within each view and then over views. The batch objectives are

$$\mathcal{L}_k = \frac{1}{N} \sum_{t \in \mathcal{B}} \ell_{k,t}, \qquad k \in \{\mathrm{act}, g, p\}, \tag{9}$$

$$\mathcal{L}_k = \frac{1}{N_w} \sum_{t \in \mathcal{B}_w} \ell_{k,t}, \qquad k \in \{w, r\}, \quad N_w > 0. \tag{10}$$

If $N_w = 0$, both local objectives are zero for that update. Counts and loss sums are aggregated across training devices before normalization. The gate uses unweighted binary cross-entropy,

$$\ell_{p,t} = -y_t \log q_{p,t} - (1 - y_t) \log(1 - q_{p,t}), \tag{11}$$

with $q_{p,t}$ denoting the probability in Equation 2 for example t. The total loss and weights follow Equation 6 and Appendix B.3. Local and refinement objectives average over the sampled active subset without inverse-probability weighting; they intentionally emphasize annotated interactions. For a positive-label fraction π, the expected training activation rate is $0.9\pi + 0.1(1 - \pi)$, not the inference activation rate. The discrete sampling variable is not differentiated.

For a training chunk x drawn from distribution p_{tr}, the active subset follows

$$p_{\mathrm{tr}}(x \mid m = 1) = \frac{\rho_{y(x)} p_{\mathrm{tr}}(x)}{\mathbb{E}_{p_{\mathrm{tr}}}[\rho_y]}. \tag{12}$$

The subset-normalized objectives therefore optimize local prediction and refinement on this interaction-weighted distribution. The action loss trains both conditioning paths under the sampled branch mixture.

B.3 Future Prediction and Refinement Paths

Observation-conditioned action path. The features F_t^g and F_t^w in Equations 1–5 are computed from current observations, the instruction, and independent Gaussian noise at future latent positions. We use a flow-time convention in which $\tau = 0$ denotes clean data and $\tau = 1$ denotes Gaussian noise, so that $\tau_f = 1$ for this path. Global features are computed first and provide the intermediate features G_t^ℓ for local prediction. The local stream does not read features from a pass conditioned on noisy ground-truth future latents. The same observation-conditioned features supply action training and inference; ground-truth future observations are used only in the auxiliary video objectives.

Following the inference algorithm of DiT4DiT, each policy query samples fresh Gaussian noise at future latent positions. Within the query, the activation head, action-conditioning projection, and global-to-local guidance reuse one global forward pass; the resulting visual conditions remain fixed across the four Action DiT sampling steps. A new query resamples the future noise. Thus, $\tau_f = 1$ specifies a fixed noise level, while activation scores remain functions of both the observation and the sampled noise.

Latent video flow matching. For each stream $v \in \{g, w\}$, let $O_{t,+}^v$ denote the ground-truth future frames over the shared prediction horizon. The frozen VAE encoder produces the clean target $z_t^v = E_v(O_{t,+}^v)$. For wrist views, encoding is performed per camera before the resulting tokens are concatenated. For each training example, we sample $\tau \sim \mathcal{U}(0, 1)$ and independent Gaussian tensors ξ_t^g and ξ_t^w, and form

$$z_{t,\tau}^v = (1 - \tau) z_t^v + \tau \xi_t^v, \qquad u_t^v = \xi_t^v - z_t^v. \tag{13}$$

The target $u_t^v = \partial z_{t,\tau}^v / \partial \tau$ is the velocity along the data-to-noise interpolation. The Video DiT predicts this field from $z_{t,\tau}^v$, the noise level, and the observation and language conditions. With $\widehat{u}_t^v$ denoting its output, the per-example video loss is

$$\ell_{v,t} = \frac{1}{M_v} \sum_{j=1}^{M_v} \frac{1}{n_{v,j}} \left\| \widehat{u}_t^{v,j} - u_t^{v,j} \right\|_2^2, \qquad M_g = 1, \quad M_w = M, \tag{14}$$

where $n_{v,j}$ counts valid latent elements for camera j. This averaging agrees with the loss aggregation in Appendix B.2. The future frames define the latent targets; the predicted velocity is supervised directly in this latent space. Both VAE encoder and decoder remain frozen.

Noise pairing and conditioning. The global and local video losses use the same sampled noise level and temporally aligned targets, but independent noise realizations. These noise tensors are also independent of those used in the observation-conditioned action path. The local video pass receives the global guidance cached from the observation-conditioned global pass, as does the local action-conditioning pass. Thus, changing the local video input during flow training does not change the source of its global condition.

Video decoding and action conditioning. For video generation, the learned velocity field is integrated from $\tau = 1$ to $\tau = 0$ to obtain a predicted clean future latent, which the frozen VAE decoder maps to RGB frames. PSNR and SSIM compare these decoded predictions with the target frames during evaluation; they are not training objectives. The action path uses the hidden features extracted at $\tau_f = 1$ without running this video sampling or decoding procedure. Its Action DiT performs action sampling conditioned on the available visual features and proprioception. Thus, learning future visual content and generating actions use the same Video DiT with different evaluation paths.

Refinement target. We distinguish the global video-training features $F_{t,\tau}^g$ from the observation-conditioned features F_t^g:

$$F_{t,\tau}^g = P\big(\mathcal{H}\big[V_\Theta\big(z_{t,\tau}^g, \tau \mid E_v(I_t^g), c\big)\big]\big), \tag{15}$$

$$\widetilde{F}_{t,\tau}^g = F_{t,\tau}^g + R_\rho(F_{t,\tau}^g, F_t^w, \tau), \tag{16}$$

$$\ell_{r,t} = \frac{1}{n_g} \left\| D_\psi(\widetilde{F}_{t,\tau}^g, \tau) - u_t^g \right\|_2^2. \tag{17}$$

Here, D_ψ is the auxiliary global flow-prediction head and n_g is the number of valid global latent elements. The batch loss $\mathcal{L}_r$ averages $\ell_{r,t}$ over examples with $m_t = 1$ as specified in Equation 10, with noise and flow time sampled on each update. The refined and original global predictions use the same noisy global input and flow target. The local condition is F_t^w from the fixed-noise observation-conditioned path, not the hidden representation of $z_{t,\tau}^w$. Consequently, refinement pairs global features at τ with local features at τ_f during video training, and pairs both streams at τ_f during action conditioning. The noise-level embedding in R_ρ identifies the global noise state in both cases. The same projection and refinement parameters are shared across these paths, and D_ψ is not used for action inference.

Gradient flow. During joint training, on examples with $m_t = 1$, $\mathcal{L}_r$ updates D_ψ, R_ρ, P, and both video streams, including the global guidance used by local prediction. The action loss also propagates through the available conditioning features. Neither path applies stop-gradient to global or local hidden features; caching global features preserves their computation graph during training. The two video streams share Θ, so their gradient contributions accumulate in the same backbone parameters. The VAE parameters are fixed throughout training. The activation head and its input features receive supervision from $\mathcal{L}_p$; action and video losses do not differentiate through branch selection. The auxiliary video passes are used only during training and do not add inference passes.

Implementation settings. We initialize the shared Video DiT from Cosmos-Predict2.5-2B (NVIDIA et al., 2025a) and retain its pretrained VAE as a frozen encoder and decoder. Following DiT4DiT (Ma et al., 2026), we extract 2048-dimensional hidden features at the 18th backbone layer. The shared projection maps them to $d = 1024$, and global-to-local cross-attention is inserted at layers $\mathcal{L}_{gw} = \{8, 16, 24\}$, using one-based layer indices. Each camera provides its current RGB

frame at 256×256 resolution. The local stream concatenates encoded tokens from the available wrist cameras; real-world deployment uses one wrist camera ($M = 1$). Each active video stream uses one forward pass at $\tau_f = 1$, and the resulting conditions are reused across four Action DiT sampling steps to generate $H = 16$ actions. The loss weights are $(\lambda_g, \lambda_w, \lambda_r, \lambda_p) = (1, 1, 0.5, 0.1)$.

Temporal sampling. Following the released DiT4DiT configuration (Ma et al., 2026) for a 16-action horizon, each video training sequence contains the current frame and eight future frames at offsets $\{2, 4, \ldots, 16\}$ in the original action sequence. All camera views use the same timestamps. These are RGB-frame offsets before VAE temporal compression; their physical spacing follows the recorded action timestamps. The 20 Hz interpolation used for deployment does not change this training-time sampling rule.

Optimization. Following the training budget in DiT4DiT (Ma et al., 2026), we train FocusWAM for 100,000 updates. We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and gradient-norm clipping at 1.0. The per-device batch size is eight with one gradient accumulation step; four NVIDIA A100 GPUs yield an effective batch size of 32. We use 5,000 linear warmup updates followed by cosine learning-rate decay to 5×10^{-7}. Peak learning rates are 1×10^{-5} for the trainable video backbone and 5×10^{-5} for the action model and additional modules, with weight decay 0.0. The inference ablations below reuse the resulting checkpoint with all parameters fixed.

B.4 Controlled Evaluation Protocols

Shared trained checkpoint. Tables 2 and 3 use the same fully trained FocusWAM checkpoint. Training includes both directions of cross-view interaction and all objectives in Equation 6, with label-guided branch sampling as specified in Equation 8. All model parameters remain fixed during the ablations. Thus, every row inherits the same learned representations, action model, activation head, and training history.

Module ablations. Table 2 changes only the enabled cross-view connections at inference. To remove global guidance, we bypass the cross-attention residual in Equation 4 at each designated layer, so that $\widetilde{W}_t^\ell = W_t^\ell$. To remove local-to-global refinement, we bypass R_ρ in Equation 5 and pass $[F_t^g; F_t^w]$ to the action model whenever the local branch is active. The variant without either component applies both interventions. All rows retain the trained activation head and threshold, both prediction streams, and the original action model. These interventions assess how the trained policy uses each direction of interaction; the training objectives and learned weights are common to all rows.

Invocation-rule comparison. Table 3 changes only the inference-time branch decision. Global-only forces $b_t = 0$ and conditions actions on F_t^g. Always-on local forces $b_t = 1$, retaining global guidance, local prediction, and refinement for every action chunk. FocusWAM uses $b_t = \mathbb{I}[q_p > 0.5]$. All three configurations use the same trained checkpoint, camera availability, preprocessing, 16-action horizon, and four-step action sampling. Training-time branch sampling and loss aggregation remain those of the shared checkpoint. The comparison measures the effect of deploying this policy with fixed or adaptive local invocation.

B.5 Synchronous execution and timing

Policy inference and action execution are synchronous, without overlapping the generation of a new chunk with execution of the preceding chunk. Each prediction contains $H = 16$ actions. The Action DiT generates this entire sequence using $N_a = 4$ numerical integration steps, each of which updates all action positions. For real-world deployment, we set the execution horizon to $K = 8$, following the default in the official DiT4DiT deployment code (Ma et al., 2026). The robot executes the first eight predicted actions, acquires an updated observation, and generates a new chunk; the remaining eight predictions are replaced by the new plan. Thus, H specifies the prediction horizon, N_a specifies the sampling computation, and K specifies the number of predicted actions executed before replanning. Predicted actions are interpolated into commands sent at 20 Hz.

The mean model latencies are 177.8 ms on RoboTwin 2.0 with an NVIDIA A100 and 181.5 ms in real-world deployment with an NVIDIA RTX 4090. Their reciprocals are approximately 5.62

and 5.51 chunks per second, respectively, characterizing model-only throughput. For synchronous execution, the interval between successive policy updates also includes action execution and system overhead:

$$T_{\text{update},t} = T_{\text{infer},t} + T_{\text{exec},t} + T_{\text{overhead},t}. \tag{18}$$

Here, $T_{\text{exec},t}$ is the duration of the commands executed before replanning, and $T_{\text{overhead},t}$ includes observation acquisition and communication outside the timed model path. The 20 Hz command rate therefore differs from the policy update rate. The action horizon specifies the number of predicted actions; their execution duration additionally depends on interpolation and the number executed before replanning.

For adaptive inference, mean latency averages the two execution paths. With invocation rate $p = \Pr(b_t = 1)$, it satisfies

$$\mathbb{E}[T_{\text{infer}}] = (1-p)\,\mathbb{E}[T_{\text{infer}} \mid b_t = 0] + p\,\mathbb{E}[T_{\text{infer}} \mid b_t = 1]. \tag{19}$$

The 177.8 ms RoboTwin result uses the observed invocation rate of 10.3%. Conditional latencies refer to the active and inactive queries of this adaptive policy, whereas the global-only and always-on entries in Table 3 summarize their respective evaluation runs. Full policy update intervals additionally include the execution and overhead terms above.