POD-RoPE: Rotary Position Encoding over Physical Observation Distributions for Embodied AI

Anonymous authors
Paper under double-blind review

Abstract

Embodied perception requires positional representations that capture both physical geometry and the spatial and temporal extent of observations. We introduce POD-RoPE (Physical Observation Distribution RoPE), which represents token observation footprints as distributions over camera centers, viewing directions, and sampling times. Under independent sampling, expected rotations encode relative relationships between distributions; when each distribution collapses to a single address, the construction recovers pointwise RoPE. The same rotation statistics support Q/K matching and coherence-dependent V/O transformations, allowing observation geometry to guide both information selection and content aggregation. Per-token pre-aggregation avoids increasing sequence length or processing pairs of observation nodes. We evaluate POD-RoPE across four robot policy architectures, simulation and real-world manipulation, and complementary reconstruction, depth-estimation, and novel-view-synthesis tasks. On two real-robot tasks, POD-RoPE raises mean success from 32.5% to 52.5%, while controlled reconstruction ablations support the contributions of distribution encoding and V/O transformations. In H20 model-block benchmarks, POD-RoPE achieves median forward-plus-backward speedups of $1.91\times$ over P-RoPE and $5.34\times$ over RayRoPE, with a median latency overhead of 17.4% over Standard RoPE. These results support physical observation distributions as an effective and computationally efficient basis for geometry-aware attention.

1 Introduction

Perception and interaction in the physical world require embodied systems to establish physically meaningful spatial and temporal relationships between observations. Standard RoPE (Su et al., 2024), however, typically defines relative positions using sequence indices or spatiotemporal grid coordinates: identical pixel offsets can correspond to different viewing angles, and identical frame-index offsets can represent different elapsed times. Positional encoding therefore needs to extend beyond discrete index differences to represent observation geometry and actual spatial and temporal scales.

Geometric positional encodings such as P-RoPE and RayRoPE incorporate camera and ray information into relative relationship modeling (Li et al., 2025; Wu et al., 2026), but incur substantially higher latency than Standard RoPE in our model-block benchmarks. Meanwhile, an image token aggregates information over a pixel region, and a temporally compressed video token combines observations from multiple times. A single center address cannot fully describe these observation footprints. Our central view is that positional encoding should represent not only *where and when a token is observed, but also the spatial and temporal extent of its observations*. We investigate how to encode these physical observation distributions and their relative relationships at low computational overhead.

We propose POD-RoPE, which maps camera centers, viewing directions, and sampling times into a shared physical address space and represents each token's observation footprint as an address distribution. Averaging address-dependent rotations yields a token-level positional operator; under independent sampling, the pairwise Q/K operator equals the expected rotation over the relative-address distribution (Eq. (6)). When each distribution concentrates at a single address, the construction recovers pointwise RoPE at the corresponding addresses. The full method reuses these statistics for V/O trans-

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.

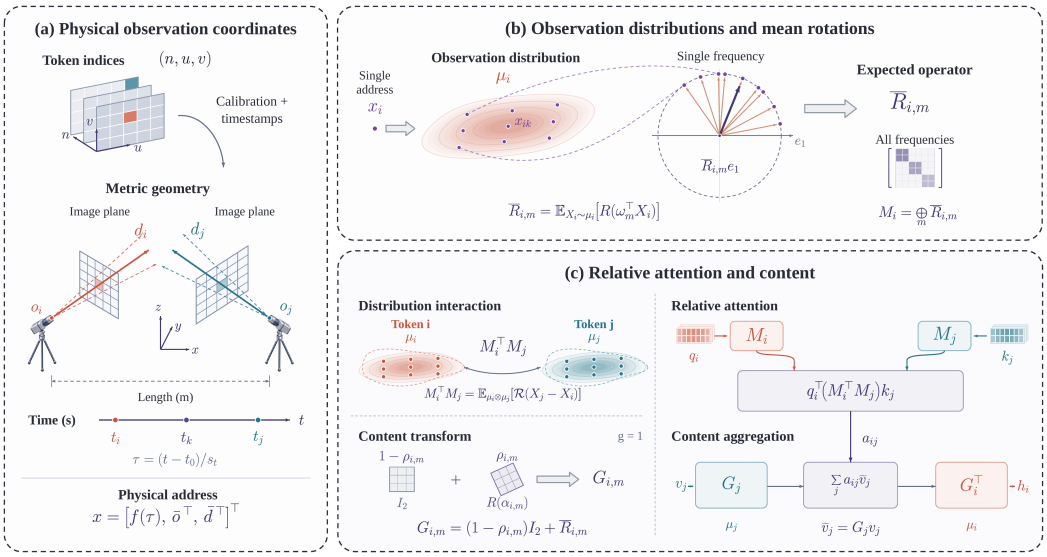


Figure 1: POD-RoPE**: physical observation distributions for attention and content aggregation.** (a) Calibration and timestamps map token indices to physical addresses. (b) Averaging address-dependent rotations under μ_i gives M_i; vectors illustrate one frequency acting on e_1. (c) For independent address samples, $M_i^\top M_j$ equals the expected relative rotation and guides Q/K matching. The full integration also transforms values and aggregated outputs using $G_{i,m} = (1-\rho_{i,m})I_2 + \overline{R}_{i,m}$ ($g = 1$), where $\rho_{i,m}$ measures rotation coherence. Distributions and rotations are schematic; rays indicate viewing directions.

formations, allowing geometry to guide both matching and aggregation. Per-token pre-aggregation avoids extra tokens and pairwise processing of observation nodes, limiting computational overhead.

We evaluate POD-RoPE across four robot policy architectures, three visual-geometry tasks, and two real-robot tasks on Sharpa North. Across all 50 RoboTwin 2.0 tasks, full integration into Fast-WAM raises mean success from 91.83% to 92.88%; on Sharpa North, it raises mean success from 32.5% to 52.5%. Visual-geometry experiments show task-dependent gains, and controlled ablations isolate the two design components. In H20 microbenchmarks, the full method achieves median forward-plus-backward speedups of $1.91\times$ over P-RoPE and $5.34\times$ over RayRoPE, with 17.4% median overhead over Standard RoPE. Our contributions are:

- **Physical observation distribution encoding.** We represent token footprints as distributions over camera centers, viewing directions, and sampling times. Expected rotations encode relative-address interactions and recover pointwise RoPE for singleton distributions.
- **Efficient matching and aggregation.** Per-token rotation statistics are shared by Q/K matching and coherence-dependent V/O transformations, avoiding additional tokens and pairwise observation-node processing.
- **Cross-domain validation.** Tests across four robot policy architectures and three visual-geometry tasks show improved aggregate manipulation success and task-dependent geometry gains. Ablations support both components; model-block benchmarks show lower latency than P-RoPE and RayRoPE.

2 Related Work

Rotary positional encoding and geometry-aware attention. RoPE applies position-dependent rotations to queries and keys so that positional interactions in attention depend on their relative displacement (Su et al., 2024). For multi-view vision, GTA applies group transformations associated with camera geometry to attention matching and value-feature aggregation (Miyato et al., 2024). P-

RoPE (denoted PRoPE in the original paper) further unifies intrinsic and extrinsic camera parameters through projective transformations, incorporating complete view-frustum relationships into relative positional representations (Li et al., 2025). RayRoPE constructs projective ray coordinates in the query-camera frame and predicts depth and its uncertainty. It uses expected RoPE to encode position distributions along rays and applies the transformations to Q/K/V/O (Wu et al., 2026). POD-RoPE addresses the observation footprints induced by tokenization. It jointly encodes camera centers, unit viewing directions, and actual time in a shared reference frame, averaging over the address distributions covered by pixels and source frames. Rotation statistics can be pre-aggregated per token and reused across attention interactions. The full method further uses rotation coherence to modulate V/O content transformations with an identity branch.

Observation footprints and spatiotemporal positional representations. Extending positional encoding from points to regions allows representations to retain observation scale. Mip-NeRF approximates conical frustums with Gaussian distributions and averages sinusoidal positional features to obtain scale-aware integrated positional encoding (Barron et al., 2021). RayRoPE applies expected rotations to uncertainty intervals of predicted depth (Wu et al., 2026). Meanwhile, DiT performs diffusion modeling on image-latent patches (Peebles & Xie, 2023), and Wan compresses spatial and temporal dimensions through a causal video autoencoder (Team Wan et al., 2025), so one visual token corresponds to multiple pixels and potentially multiple source frames. We map these nominal footprints into physical observation distributions, using camera calibration and timestamps to determine their addresses and the phase and magnitude of average rotations to retain the observation composition. This applies regional integration to relative matching and content aggregation between compressed visual tokens, with one construction shared by static multi-view and video observations.

Vision-action policies and world-action models. ACT predicts action chunks (Zhao et al., 2023), RDT uses a multimodal diffusion Transformer (Liu et al., 2025), and π_0 combines a pretrained vision-language model with flow-matching action generation (Black et al., 2025). Fast-WAM jointly models video and actions during training but omits explicit future-video generation at inference (Yuan et al., 2026). We study observation geometry and footprints as positional inputs across these policy architectures; Section 4.1 specifies their integrations.

Embodied data and visual geometry evaluation. We evaluate manipulation on RoboTwin 2.0 (Mu et al., 2025; Chen et al., 2025) and Sharpa North, and reconstruction using RoboTwin, AgiBot World (Bu et al., 2025), and collected Sharpa North observations. Complementary geometry tasks use UniMatch (Xu et al., 2023) with the DeMoN depth protocol (Ummenhofer et al., 2017), and LVSM (Jin et al., 2025) for novel-view synthesis on RealEstate10K (Zhou et al., 2018).

3 Method

POD-RoPE incorporates camera geometry and actual sampling time into positional encoding through physical observation addresses, observation distributions, and their expected rotations. We next describe the address and distribution constructions, relative Q/K relationships, and efficient computation and V/O content aggregation through shared rotation statistics (Figure 1).

3.1 From Discrete Indices to Physical Observation Addresses

Discrete indices describe the arrangement of tokens in a tensor but do not directly express camera baselines, viewing directions, or actual time intervals. We use these observed quantities as positional inputs to RoPE. For pixel p of camera c at time t, let $\widetilde{p}$ denote homogeneous pixel coordinates, $K_c(t)$ the intrinsics, and $(Q_c(t), b_c(t))$ the extrinsics mapping camera coordinates to a shared reference frame. For undistorted pinhole images with matching intrinsics, the camera center and unit viewing direction are

$$o(t,c) = b_c(t),$$
$$d(t,p,c) = Q_c(t)\frac{K_c(t)^{-1}\widetilde{p}}{\|K_c(t)^{-1}\widetilde{p}\|_2}. \tag{1}$$

Normalizing actual time, camera centers, and directions by specified scales gives the physical observation address

$$x = g(t, p, c) = \left[f(\tau(t)),\, \bar{o}(t, c)^\top,\, \bar{d}(t, p, c)^\top \right]^\top. \tag{2}$$

where $\tau = (t - t_0)/s_t$, $\bar{o} = o/s_o$, and $\bar{d} = s_d d$. The parameters s_t and s_o specify time and length scales, respectively, while s_d adjusts the directional component. The temporal mapping f controls how phase changes with time; our experiments use $f(\tau) = 2\tanh(\tau/2)$. Static observations omit the time component. This address describes where, in which direction, and when an observation is made, without assuming the unknown surface depth along the ray. Robot experiments use camera-to-robot-root extrinsics. Appendix A.4 details the calibration chain, intrinsic updates, and actual-time mapping.

3.2 From a Single Address to an Observation Distribution

A token typically aggregates a pixel region, and a video token may also integrate multiple times. Even with physical addresses, encoding only the center therefore omits the observation footprint: tokens with the same central viewing direction or time can cover different fields of view or temporal spans. We represent the physical addresses covered by token i and their normalized weights as a probability distribution μ_i, with $X_i \sim \mu_i$. It describes the composition of observations within the token, rather than uncertainty about an unknown scene position. Appendix A.1 gives the construction.

To incorporate these observations into positional encoding, we first define the rotation for a single address and then average it over the observation distribution. For head dimension $d_h = 2M$, choose frequency vectors $\omega_m \in \mathbb{R}^D$ and define

$$\mathcal{R}(x) = \bigoplus_{m=1}^{M} R(\omega_m^\top x), \tag{3}$$

where $R(\theta)$ is a 2D rotation matrix and $\bigoplus$ places the rotation blocks along the diagonal. The token's positional operator is

$$M_i = \mathbb{E}_{X_i \sim \mu_i}[\mathcal{R}(X_i)]. \tag{4}$$

This construction allows all observations covered by a token to determine its positional encoding jointly. When the distribution concentrates at one address x_i, $M_i = \mathcal{R}(x_i)$, recovering pointwise RoPE at that physical address.

3.3 Preserving Relative Relationships between Distributions

After averaging rotations, the key question is whether RoPE's relative-position property is preserved. Pointwise rotations satisfy

$$\mathcal{R}(x)^\top \mathcal{R}(y) = \mathcal{R}(y - x). \tag{5}$$

Thus, independently encoding two addresses yields the rotation of their relative address through pairwise interaction. Distribution encoding preserves this structure:

$$M_i^\top M_j = \mathbb{E}[\mathcal{R}(X_j - X_i)]. \tag{6}$$

Here, $X_i \sim \mu_i$ and $X_j \sim \mu_j$ are sampled independently, representing all pairwise combinations of the two observation sets under their respective weights. The equality follows directly from the relative-position property of pointwise rotations and linearity of expectation; Appendix A.1 provides the proof.

The interaction between two positional operators is therefore determined by the distribution of relative observation addresses, extending relative relationships between points to those between observation sets. Relative addresses contain the mapped time difference $f(\tau_j) - f(\tau_i)$, camera-center displacement, and viewing-direction difference. Let q_i, k_j denote queries and keys after projection and any model-required normalization, but before the positional operator. The attention logit is

$$\ell_{ij} = \frac{(M_i q_i)^\top (M_j k_j)}{\sqrt{d_h}}. \tag{7}$$

Equation (6) thus directly constrains pairwise positional interactions in attention. Each model applies its required Q/K normalization before the positional operator; Appendix E.1 specifies the order.

3.4 Efficient Computation and Distribution-Guided Content Aggregation

Computing expected rotations. For each token and frequency, the mean cosine and sine suffice to construct the positional operator:

$$c_{i,m} = \mathbb{E}[\cos(\omega_m^\top X_i)], \qquad s_{i,m} = \mathbb{E}[\sin(\omega_m^\top X_i)],$$
$$\overline{R}_{i,m} = \begin{bmatrix} c_{i,m} & -s_{i,m} \\ s_{i,m} & c_{i,m} \end{bmatrix}, \qquad M_i = \bigoplus_m \overline{R}_{i,m}. \tag{8}$$

These statistics are reused across all attention interactions without processing pairs of observation nodes. Encoding retains Standard RoPE's computation over 2D feature blocks. Appendix A.5 provides numerical quadrature and complexity details.

Constructing content transformations from rotation coherence. Average rotations generally combine rotation and contraction. For $\rho_{i,m} > 0$, we have $\overline{R}_{i,m} = \rho_{i,m} R(\alpha_{i,m})$, where $\rho_{i,m} = \sqrt{c_{i,m}^2 + s_{i,m}^2} \leq 1$ measures rotation coherence and $\alpha_{i,m}$ is the mean direction. Directly transforming content with M_i would suppress value features when rotations cancel, so we introduce an identity branch:

$$G_{i,m} = (1 - g\rho_{i,m}) I_2 + g\overline{R}_{i,m}, \qquad G_i = \bigoplus_m G_{i,m}, \qquad g \in [0, 1]. \tag{9}$$

This mixes the identity and the rotation in the mean direction with weights $1 - g\rho_{i,m}$ and $g\rho_{i,m}$. Greater coherence assigns more weight to the rotation branch. At $\rho_{i,m} = 0$, the expression directly gives $G_{i,m} = I_2$, preserving content without defining a mean direction. Appendices A.2 and A.3 analyze coherence under fixed sampling templates and the transformation properties.

Distribution-guided content aggregation. Let $a_{ij} = \text{softmax}_j(\ell_{ij})$. We transform values before weighted aggregation and apply an output transformation afterward:

$$h_i = G_i^\top \sum_j a_{ij} G_j v_j. \tag{10}$$

Here, a_{ij} weights each token's contribution, while $G_i^\top G_j$ transforms its content features. Setting $g = 0$ yields Q/K-only integration; the full method fixes $g = 1$ without adding learnable gating parameters. Appendix Table 12 lists model-specific integrations.

4 Experiments

Robot experiments evaluate the complete policy integrations; Table 6 tests the contributions of distribution encoding and V/O transformations. Visual geometry and model-block benchmarks assess quality and efficiency. Appendices E and E.7 provide complete training and evaluation protocols.

4.1 End-to-End Embodied Manipulation

Simulation evaluation setup. We use two training regimes on RoboTwin 2.0 (Chen et al., 2025). Multitask training uses Fast-WAM (Yuan et al., 2026) as the baseline and jointly trains a shared policy on 2,500 Clean demonstrations and 25,000 randomized-scene demonstrations from 50 tasks. Single-task training learns a separate policy for each task; we adapt π_0, RDT, and ACT on 15 tasks with 50 Clean demonstrations per task. Both regimes evaluate 100 episodes per task under each of the Clean and Randomized conditions and report task-averaged success rates. Randomized conditions vary clutter, lighting, textures, and table height. Both comparisons follow the training recipe and evaluation settings of their corresponding baselines. All policies use Q/K+V/O integration. Appendices C and E.3 provide detailed configurations.

Joint multitask training. Table 1 shows gains under both Clean and Randomized conditions, raising mean success across the two conditions from 91.83% to 92.88%. Integrating POD-RoPE into Fast-WAM improves task-averaged success by 1.24 and 0.86 percentage points under Clean and Randomized conditions, respectively. The table lists the first five tasks in their original order; Appendix Table 9 reports all 50, including task-level regressions.

Table 1: Fast-WAM success rates (%) on 50 RoboTwin 2.0 tasks. Baseline (Yuan et al., 2026) is Fast-WAM; Ours integrates POD-RoPE into Fast-WAM. Both use identical training and evaluation settings. Light blue shading identifies our method; the best average results are bold.

	Baseline		Ours	
Task	Clean	Rand.	Clean	Rand.
adjust_bottle	100	100	100	100
beat_block_hammer	99	97	100	98
blocks_ranking_rgb	100	100	100	100
blocks_ranking_size	94	98	95	94
click_alarmclock	100	100	100	100
⋮	⋮	⋮	⋮	⋮
Average (50 tasks)	91.88	91.78	**93.12**	**92.64**
Δ (pp)	–	–	+1.24	+0.86

Table 2: Success rates (%) after single-task fine-tuning on 15 RoboTwin 2.0 tasks across different model architectures. Each model is compared under its corresponding baseline's training and evaluation configuration. Baseline (Chen et al., 2025) is the original policy; Ours integrates full POD-RoPE. Rand. denotes Randomized conditions; bold marks the best result per task and condition across the six configurations.

Task	π_0				RDT				ACT			
	Baseline		Ours		Baseline		Ours		Baseline		Ours	
	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.	Clean	Rand.
adjust bottle	90	56	**100**	**100**	81	75	97	88	97	23	99	26
beat block hammer	43	21	51	33	77	**37**	**89**	33	56	3	77	0
blocks ranking rgb	19	5	**26**	**22**	3	0	9	1	1	0	4	0
click bell	44	3	**96**	**24**	80	9	83	0	58	3	92	8
handover block	45	8	53	15	45	14	**67**	**16**	42	0	28	0
hanging mug	11	3	15	10	**23**	**16**	16	6	7	0	7	0
lift pot	84	36	**95**	**47**	72	9	88	22	88	0	90	0
move can pot	58	21	49	**31**	25	12	29	18	22	4	**60**	5
open laptop	**85**	46	84	40	59	32	72	**50**	56	0	68	0
open microwave	80	50	**100**	**63**	37	20	90	29	86	0	85	0
pick dual bottles	**57**	12	49	**22**	42	13	55	11	31	0	46	0
place object basket	16	2	**72**	**57**	33	17	45	11	15	0	20	0
press stapler	62	29	**80**	**36**	41	24	43	26	31	6	49	7
stack blocks three	17	0	**31**	**14**	2	0	7	1	0	0	20	0
stack bowls two	**91**	41	82	**55**	76	30	83	34	82	0	85	0
Mean	53.47	22.20	**65.53**	**37.93**	46.40	20.53	58.20	23.07	44.80	2.60	55.33	3.07
Δ (pp)	–	–	+12.07	+15.73	–	–	+11.80	+2.53	–	–	+10.53	+0.47

Single-task training across architectures. After integrating POD-RoPE, all three single-task policies achieve higher average success rates (Table 2). π_0 benefits under both conditions, whereas RDT and ACT improve primarily under Clean conditions; ACT's Randomized success rate remains 3.07%. The gains therefore vary substantially across architectures and evaluation conditions.

Real-robot manipulation on Sharpa North. We further evaluate bottle opening and straw insertion on the Sharpa North bimanual robot to evaluate the complete POD-RoPE integration in real manipulation. Both methods train jointly on the two tasks, use observations from the head's left-eye camera and both wrist cameras, and are evaluated for 20 trials per task at the 20,000-step checkpoint. The real-robot version enables both Q/K distribution encoding and V/O content transformations in the video and action branches. Appendix E.7 gives the training settings.

Table 3: Real-robot task results on Sharpa North. Both methods use the same architecture; the baseline uses standard RoPE. Both use the 20,000-step checkpoint and are evaluated for 20 trials per task; success rates are in parentheses. Δ denotes the success-rate difference in percentage points.

Task	Fast-WAM	Fast-WAM + POD-RoPE	Δ (pp)
Bottle opening	7/20 (35%)	12/20 (60%)	+25.0
Straw insertion	6/20 (30%)	9/20 (45%)	+15.0
Average (2 tasks)	32.5%	**52.5%**	+20.0

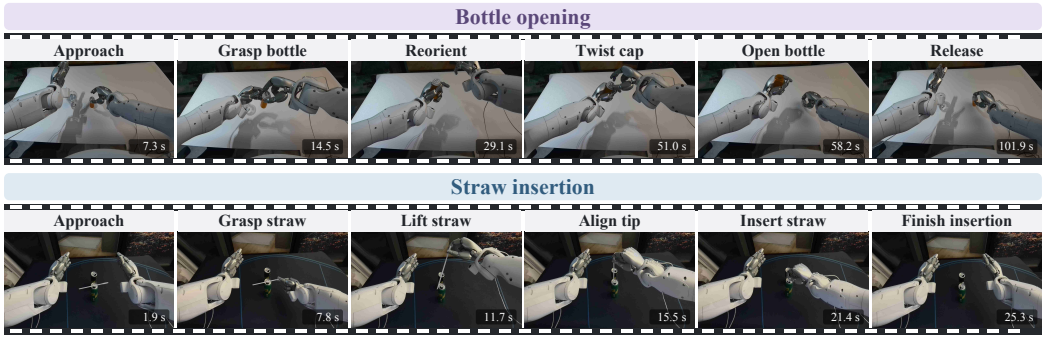


Figure 2: Bottle-opening and straw-insertion examples on Sharpa North. Each row shows six moments from one successful training demonstration, ordered from left to right; times are relative to the start of that recording. The images illustrate the task process. Table 3 reports the real-robot evaluation results of the two policies.

Table 4: Cross-view reconstruction and $h16$ conditional temporal extrapolation with different positional encodings on three datasets. At $h16$, the target occurs 16 time steps after the reference observations, with each step spanning four original synchronized sampling intervals: approximately 2.15, 4.27, and 2.13 seconds on Sharpa North, RoboTwin, and AGIBot, respectively. Bold marks the best result within each dataset and evaluation setting.

		Cross-view Reconstruction				$h16$ Conditional Reconstruction			
Dataset	Method	MSE↓	PSNR↑	SSIM↑	LPIPS↓	MSE↓	PSNR↑	SSIM↑	LPIPS↓
Sharpa North	Standard RoPE	0.1931	16.264	0.5682	0.5668	0.3078	15.204	0.5364	0.5937
Sharpa North	P-RoPE	0.1129	18.825	0.6476	0.2728	0.2114	17.160	0.5931	0.3678
Sharpa North	POD-RoPE	**0.1022**	**19.376**	**0.6779**	**0.2371**	**0.1853**	**17.808**	**0.6279**	**0.3164**
RoboTwin	Standard RoPE	0.1378	22.064	0.7474	0.3921	0.4716	17.733	0.6917	0.5085
RoboTwin	P-RoPE	0.1335	22.459	0.7517	0.3451	0.3961	18.104	0.6964	0.4720
RoboTwin	POD-RoPE	**0.1130**	**23.302**	**0.7643**	**0.3235**	**0.3631**	**18.571**	**0.7036**	**0.4552**
AGIBot	Standard RoPE	0.2841	18.164	0.5768	0.5249	0.3377	18.285	0.5869	0.5279
AGIBot	P-RoPE	0.1862	20.622	0.6354	**0.3308**	0.2477	20.223	0.6314	**0.3533**
AGIBot	POD-RoPE	**0.1806**	**20.998**	**0.6392**	0.3319	**0.2387**	**20.709**	**0.6363**	0.3564

Both real-robot tasks improve, raising mean success from 32.5% to 52.5% (Table 3). These tasks test bimanual coordination, rotation, and small-object alignment. Figure 2 illustrates their demonstration sequences.

Table 5: Depth estimation with different positional encodings on UniMatch/DeMoN. All methods use the same backbone, training, and evaluation settings; bold marks the best results.

Dataset	Method	AbsRel↓	SqRel↓	RMSE↓	RMSE-log↓	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$
RGBD	Standard RoPE	0.1753	0.2523	0.9085	0.2716	0.7273	0.8741	0.9407
	P-RoPE	0.1231	0.2129	0.6839	0.2115	0.8523	**0.9264**	0.9463
	RayRoPE	**0.1221**	0.2303	**0.6668**	0.2071	**0.8563**	0.9244	0.9429
	POD-RoPE (Q/K-only)	0.1307	**0.1982**	0.7041	0.2051	0.8133	0.9194	**0.9491**
	POD-RoPE	0.1243	0.2094	0.6753	**0.2033**	0.8403	0.9227	0.9454
Scenes11	Standard RoPE	0.1185	0.2060	0.9395	0.2104	0.8660	0.9422	0.9700
	P-RoPE	0.0681	0.0939	0.6149	0.1332	0.9455	0.9816	0.9908
	RayRoPE	0.0597	0.0810	0.5390	0.1216	**0.9603**	**0.9863**	0.9918
	POD-RoPE (Q/K-only)	0.0696	0.0954	0.6002	0.1337	0.9490	0.9829	0.9899
	POD-RoPE	**0.0594**	**0.0739**	**0.5306**	**0.1215**	**0.9603**	0.9850	**0.9923**

4.2 Further Evaluation on Visual Geometry Tasks

Synchronized cross-view reconstruction. We reconstruct a third view from two simultaneous camera observations on Sharpa North, RoboTwin, and AGIBot, using shared splits and trajectory/task-balanced evaluation. Table 4 compares Standard RoPE, P-RoPE, and a fixed POD-RoPE configuration. Relative to P-RoPE, POD-RoPE improves all four metrics on Sharpa North and RoboTwin, and improves PSNR but slightly worsens LPIPS on AGIBot. Spatial reconstruction gains are thus dataset-dependent. Appendix Figure 4 shows examples from identical reference observations.

Conditional temporal extrapolation. We also fix the current reference observations and supply the future target's camera geometry and time. Under matched settings, POD-RoPE improves future-target PSNR over P-RoPE on all three datasets (Table 4), supporting physical spatiotemporal encoding for conditional temporal reconstruction.

Appendix E.2 provides complete curves and analysis across temporal horizons, and Appendix D reports experiments with input changes.

Depth estimation. We compare five positional encodings on UniMatch/DeMoN under identical backbone, training, and evaluation settings (Table 5). Full POD-RoPE improves 12 of 14 metrics over POD-RoPE (Q/K-only) and 10 over P-RoPE. It leads on five Scenes11 metrics and ties RayRoPE on δ_1, whereas RayRoPE performs better on four RGBD metrics. Section 4.4 separately evaluates model-block cost, and Appendix E gives the setup.

Novel-view synthesis. Under matched 80k training on RE10K/LVSM, full POD-RoPE improves all three quality metrics over POD-RoPE (Q/K-only), including a 0.398 dB PSNR gain. Its PSNR exceeds Plücker Raymap and GTA, while RayRoPE remains best on all three metrics and leads POD-RoPE by 0.832 dB. Figure 3(b) shows the quality–latency trade-off; Appendix Table 8 reports all metrics.

4.3 Ablation Studies

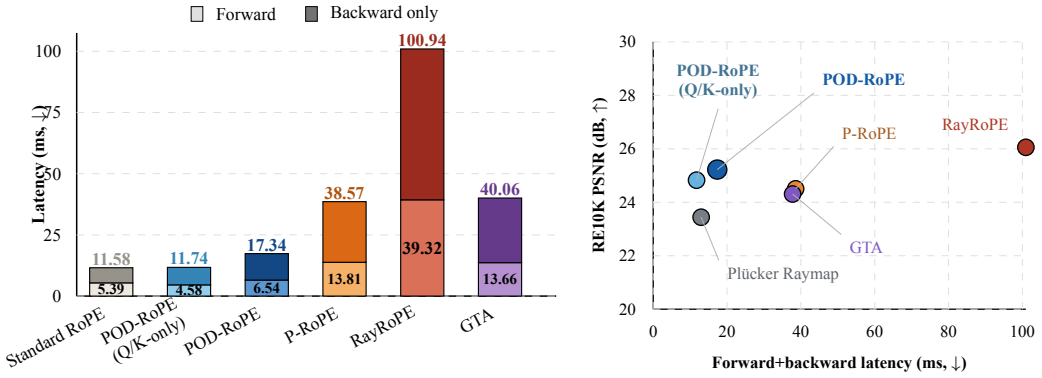
Physical point Q/K raises PSNR from 18.138 to 21.832 dB over Standard RoPE (Table 6), supporting physical addresses with their frequency construction.

With coordinates and frequencies fixed, distribution Q/K improves PSNR/SSIM/LPIPS by $+0.534/+0.0259/-0.0373$ over point Q/K. With V/O enabled, distributions improve all three metrics over points, including 0.744 dB in PSNR. Both comparisons support encoding token footprints beyond single addresses.

V/O improves all metrics with both encodings, raising PSNR by 0.340 dB for points and 0.550 dB for distributions. Direct-average V/O instead underperforms distribution Q/K-only by 0.373 dB and worsens SSIM/LPIPS. While $\overline{R} = \rho R(\alpha)$ contracts features at low coherence, adding $(1-\rho)I$ makes

Table 6: Component ablation on Sharpa North head-to-wrist reconstruction under matched settings. Direct-average V/O keeps distribution Q/K fixed. Bold marks the best results.

Configuration	PSNR↑	SSIM↑	LPIPS↓
Standard RoPE	18.138	0.6576	0.6014
Physical Point (Q/K-only)	21.832	0.7206	0.3194
Physical Point (Q/K+V/O)	22.172	0.7331	0.2997
POD-RoPE (Q/K-only)	22.366	0.7465	0.2821
Distribution + direct-average V/O	21.993	0.7255	0.3341
POD-RoPE	**22.916**	**0.7524**	**0.2655**



(a) Model-block latency

(b) RE10K quality–latency trade-off

Figure 3: Computation cost and reconstruction quality on an H20. (a) Six-layer reconstruction-block latency at 256×320, B1, T1, and 960 tokens. Light and dark segments show forward and backward-only time; totals give forward+backward time, using medians over 30 measurements. (b) RE10K PSNR versus forward+backward latency of the six-layer LVSM block at 256×320, B1, three views, and 960 tokens. RayRoPE takes 100.942 ms and is shown at the 100-ms boundary. The panels use separate model-block benchmarks, not end-to-end timings; their absolute latencies are not pooled. Full protocols and values are in Appendix B and Table 8.

the operator approach identity as coherence vanishes. The full method gains 0.923 dB over direct averaging, supporting this identity branch (Appendix E.6).

4.4 Computational Efficiency

In matched model-block microbenchmarks, full POD-RoPE takes $1.281\times/1.174\times$ the Standard RoPE latency for forward/forward-plus-backward passes; POD-RoPE (Q/K-only) takes $1.073\times/1.034\times$. Full POD-RoPE achieves speedups of $1.81\times/1.91\times$ over P-RoPE across seven configurations and $5.69\times/5.34\times$ over RayRoPE across five supported configurations, using median per-configuration ratios. Figure 3 shows model-block cost and the separate RE10K quality–latency trade-off; Appendix B provides complete timings.

5 Conclusion

POD-RoPE represents token footprints as distributions of physical observation addresses. Expected rotations preserve relative Q/K interactions, while coherence-dependent V/O transformations reuse these statistics to guide content aggregation. The same construction handles static and temporal observations without increasing sequence length. Complete policy integrations improve aggregate success rates in closed-loop experiments, including Sharpa North manipulation; controlled reconstruction ablations support the contributions of distribution encoding and V/O. Visual geometry evaluations reveal task-dependent gains and quality–cost trade-offs, with low additional model-block latency relative to Standard RoPE.

AI Use Statement

Generative AI tools assisted result analysis, writing and translation, methodological and experimental discussions, code implementation, and figure preparation. AI-assisted analyses were subsequently checked by the authors. The authors led the research motivation, methodology, and scientific decisions, and take responsibility for the experiments, reported results, and final manuscript.

References

Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-NeRF: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5855–5864, 2021. URL https://openaccess.thecvf.com/content/ICCV2021/html/Barron_Mip-NeRF_A_Multiscale_Representation_for_Anti-Aliasing_Neural_Radiance_Fields_ICCV_2021_paper.html.

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 *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.010.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Xindong He, Xu Huang, et al. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3549–3556, 2025. doi: 10.1109/IROS60139.2025.11247088.

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. *arXiv preprint arXiv:2506.18088*, 2025. URL https://arxiv.org/abs/2506.18088v2.

Haian Jin, Hanwen Jiang, Hao Tan, Kai Zhang, Sai Bi, Tianyuan Zhang, Fujun Luan, Noah Snavely, and Zexiang Xu. LVSM: A large view synthesis model with minimal 3D inductive bias. In *International Conference on Learning Representations*, 2025.

Ruilong Li, Brent Yi, Junchen Liu, Hang Gao, Yi Ma, and Angjoo Kanazawa. Cameras as relative positional encoding. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-0540.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1B: A diffusion foundation model for bimanual manipulation. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/49f80e4d2471ad4f2edf4f5f1ab62339-Abstract-Conference.html.

Takeru Miyato, Bernhard Jaeger, Max Welling, and Andreas Geiger. GTA: A geometry-aware attention mechanism for multi-view transformers. In *International Conference on Learning Representations*, 2024.

Yao Mu, Tianxing Chen, Zanxin Chen, Shijia Peng, Zhiqian Lan, Zeyu Gao, Zhixuan Liang, Qiaojun Yu, Yude Zou, Mingkun Xu, Lunkai Lin, Zhiqiang Xie, Mingyu Ding, and Ping Luo. RoboTwin: Dual-arm robot benchmark with generative digital twins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 27649–27660, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Mu_RoboTwin_Dual-Arm_Robot_Benchmark_with_Generative_Digital_Twins_CVPR_2025_paper.html.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4195–4205, 2023.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/j.neucom.2023.127063.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025. URL https://arxiv.org/abs/2503.20314.

Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5622–5631, 2017. doi: 10.1109/CVPR.2017.596.

Yu Wu, Minsik Jeon, Jen-Hao Rick Chang, Oncel Tuzel, and Shubham Tulsiani. RayRoPE: Projective ray positional encoding for multi-view attention. In *Computer Vision – ECCV 2026*, volume 17024 of *Lecture Notes in Computer Science*, pp. 192–210. Springer Nature Switzerland, 2026. doi: 10.1007/978-3-032-37556-8_11.

Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, Fisher Yu, Dacheng Tao, and Andreas Geiger. Unifying flow, stereo and depth estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13941–13958, 2023. doi: 10.1109/TPAMI.2023.3298645.

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*, 2026. URL https://arxiv.org/abs/2603.16666v1.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.016.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics*, 37(4):65:1–65:12, 2018. doi: 10.1145/3197517.3201323.

A Method Derivations and Implementation Details

A.1 Observation Distributions and the Relative-Position Property

Observation distributions. The observation distribution μ_i consists of the physical addresses covered by a token and their normalized weights. For $X_i \sim \mu_i$, $\mathbb{E}[F(X_i)]$ denotes averaging with observation weights; it is computed using finitely many nodes in practice.

Proof of the relative-position property. Two-dimensional rotations satisfy $R(\alpha)^\top R(\beta) = R(\beta - \alpha)$, so Eq. (3) satisfies $\mathcal{R}(x)^\top \mathcal{R}(y) = \mathcal{R}(y - x)$. Sampling $X_i \sim \mu_i$ and $X_j \sim \mu_j$ independently according to the observation weights of the two tokens gives

$$\begin{aligned} M_i^\top M_j &= \mathbb{E}[\mathcal{R}(X_i)]^\top \mathbb{E}[\mathcal{R}(X_j)] \\ &= \mathbb{E}[\mathcal{R}(X_i)^\top \mathcal{R}(X_j)] \\ &= \mathbb{E}[\mathcal{R}(X_j - X_i)]. \end{aligned} \tag{11}$$

The second step uses independent sampling, and the third uses the relative-position property of pointwise rotations. This proves the Q/K relative-position identity in the main text. For finite nodes, the pairwise weights are $w_{ik}w_{jl}$. Self-interaction of a token also uses two independent samples.

Two immediate consequences. When all observations of a token concentrate at address x_i, $M_i = \mathcal{R}(x_i)$, recovering pointwise RoPE with the same addresses and frequencies. If a common offset a is added to every address, $(X_j + a) - (X_i + a) = X_j - X_i$, so the pairwise Q/K positional kernel remains unchanged.

A.2 Observation Footprints and Rotation Magnitude

An alternative expression for average rotations. The mean cosine and sine can be combined into a complex number:

$$\Phi_i(\omega_m) = c_{i,m} + \mathrm{i}s_{i,m} = \mathbb{E}[e^{\mathrm{i}\omega_m^\top X_i}]. \tag{12}$$

This is the characteristic function of the observation distribution under the positive-exponent convention. It jointly expresses the direction and magnitude of the average rotation at the selected frequency.

Identical centers with different footprints. Consider a uniform distribution over one-dimensional physical addresses, $X \sim \mathrm{Unif}[c - a, c + a]$, where c is the center and $a > 0$ is the half-width of the footprint. Its average rotation is

$$\Phi(\omega) = \frac{1}{2a}\int_{c-a}^{c+a} e^{\mathrm{i}\omega x}\, dx = e^{\mathrm{i}\omega c}\frac{\sin(\omega a)}{\omega a}. \tag{13}$$

Center encoding retains only $e^{\mathrm{i}\omega c}$, while distribution encoding also retains the footprint factor. For $\omega a = \pi/6$ and $\pi/2$, the two distributions have the same center and mean direction but magnitudes of approximately 0.955 and 0.637, respectively. Distribution encoding can thus distinguish footprints that the center phase cannot express.

Meaning of the magnitude. Each pointwise rotation corresponds to a unit-length vector, whose mean satisfies

$$\rho_{i,m} = |\Phi_i(\omega_m)| \le \mathbb{E}[|e^{\mathrm{i}\omega_m^\top X_i}|] = 1. \tag{14}$$

The magnitude equals one if and only if the observation rotations have the same direction at that frequency, allowing phases to differ by integer multiples of 2π. For $\rho_{i,m} > 0$,

$$\overline{R}_{i,m} = \rho_{i,m} R(\arg \Phi_i(\omega_m)). \tag{15}$$

The magnitude therefore describes rotation coherence at that frequency. Both widths in the example above lie within $0 < \omega a < \pi$; for more general footprints and frequencies, the magnitude should be computed from the actual phase distribution.

Why a fixed sampling template can still yield varying magnitudes. Let the fixed pixel nodes be $p_i + \delta_k$ and the address mapping be g_i, with $x_{ik} = g_i(p_i + \delta_k)$ and $x_i = g_i(p_i)$. Since the center phase has unit modulus,

$$\rho_{i,m} = \left|\sum_k w_k e^{\mathrm{i}\omega_m^\top (x_{ik} - x_i)}\right|. \tag{16}$$

The magnitude is determined by physical address offsets. Using the mapping's local Jacobian J_i, $x_{ik} - x_i \approx J_i\delta_k$, so a fixed pixel template can still produce different magnitudes as intrinsics, viewing location, or camera orientation change. If the physical offsets are also identical, the magnitude is fixed. The camera-center term shared by the same camera at the same frame does not independently change the magnitude. The V/O ablation compares direct average rotations with and without the coherence-dependent identity branch; it does not separately compare retaining magnitude against phase-only transport.

A.3 Properties of Distribution-Guided Content Aggregation

Norm bound on the transformation. Equation (9) forms a convex combination of the identity and a rotation. Using $\|\overline{R}_{i,m}\|_2 = \rho_{i,m}$, for $g, \rho_{i,m} \in [0, 1]$ we have

$$\|G_{i,m}\|_2 \le (1 - g\rho_{i,m}) + g\|\overline{R}_{i,m}\|_2 = 1. \tag{17}$$

Thus $\|G_i\|_2 \le 1$, so the content transformation itself cannot amplify the feature norm. When $\rho_{i,m} = 0$, the block returns to the identity.

Relationship to Q/K relativity. Omitting the frequency subscript and writing $\beta_i = 1 - g\rho_i$ gives

$$G_i^\top G_j = \beta_i\beta_j I + g\beta_i\overline{R}_j + g\beta_j\overline{R}_i^\top + g^2\overline{R}_i^\top\overline{R}_j. \tag{18}$$

The last term has the relative structure of Q/K, while the two linear terms also contain the individual observation addresses. Invariance to a common address translation therefore applies to the Q/K positional kernel; distribution-guided content aggregation generally also uses absolute address information.

A.4 Physical Addresses and Preprocessing Conventions

Coordinates and scales. In Eq. (2), time is measured in seconds and camera centers retain the input coordinate units (meters for metric poses and reference-baseline-normalized units for RE10K). The scales $s_t, s_o > 0$ have the corresponding units, and $s_d > 0$ adjusts the direction scale. Static tasks omit the time component of the seven-dimensional address. Robot experiments use camera-to-robot-root extrinsics. When the RGB undistortion, crop/resize, and intrinsic-update chain has not been fully verified, d denotes a nominal ray direction.

Relative addresses. The address difference entering the positional operator is

$$x_j - x_i = \big[f(\tau_j) - f(\tau_i),\ (o_j - o_i)/s_o,\ s_d(d_j - d_i)\big]. \tag{19}$$

The three terms represent the mapped time difference, camera baseline or motion displacement, and viewing-direction difference. With $f(\tau) = \tau$, the first term recovers the actual time interval scaled by s_t. If two unit viewing directions form an angle θ, then $\|d_j - d_i\|_2 = 2\sin(\theta/2)$. Together, these quantities describe geometric and temporal relationships between observations.

Image preprocessing. Pinhole images should first be undistorted, with intrinsics updated consistently after cropping and resizing. If homogeneous pixel coordinates satisfy $\widetilde{p}' = A_{\text{img}}\widetilde{p}$, choosing $K' = A_{\text{img}}K$ yields $K'^{-1}\widetilde{p}' = K^{-1}\widetilde{p}$, preserving the back-projected viewing direction of corresponding pixels. Other camera models use their corresponding calibrated back-projection. When the resolution or tokenizer changes, observation distributions are reconstructed from the new pixel footprints.

All addresses containing time use $f(\tau) = 2\tanh(\tau/2)$, which is approximately linear near small offsets and limits phase growth at long offsets. The relative-position property applies to the mapped addresses.

A.5 Numerical Implementation and Complexity

Weighted averaging over observation nodes. We select K_i deterministic nodes within a token's pixel and temporal footprint, map them to addresses x_{ik}, and assign normalized weights $w_{ik} \geq 0$ with $\sum_k w_{ik} = 1$. The statistics at each frequency are computed directly as

$$\begin{aligned}\widehat{c}_{i,m} &= \sum_{k=1}^{K_i} w_{ik}\cos(\omega_m^\top x_{ik}),\\ \widehat{s}_{i,m} &= \sum_{k=1}^{K_i} w_{ik}\sin(\omega_m^\top x_{ik}).\end{aligned} \tag{20}$$

Substituting this pair of statistics into Eq. (8) gives the numerical positional operator, which is reused across all attention interactions involving that token.

Spatiotemporal quadrature nodes. Visual tokens use 3×3 spatial nodes on $\{1/6, 3/6, 5/6\}^2$, with weights

$$w_{ab} \propto \exp\left(-\frac{\|p_{ab} - (1/2, 1/2)\|_2^2}{2\sigma^2}\right), \qquad \sigma = 0.25 \tag{21}$$

normalized to form the observation weights. For temporal quadrature, we assign each latent token a nominal bin of at most four source frames according to the tokenizer's temporal compression stride. This bin specifies our observation nodes rather than the full receptive field of the causal VAE. Combined with nine spatial nodes per frame, this gives at most 36 nodes per token. Table 12 lists the frequency bands, scales, and V/O settings for each task.

Computation and storage. For N tokens, $M = d_h/2$ frequencies, and a fixed address dimension, computing the statistics costs $\mathcal{O}(M \sum_i K_i)$ and requires $\mathcal{O}(NM)$ storage. Q/K encoding and V/O transformations each cost $\mathcal{O}(Nd_h)$, while the dominant attention term remains $\mathcal{O}(N^2 d_h)$. Section 4.4 reports measured latency.

B Complete Numerical Results

B.1 Per-Case Latency

Table 7 reports raw median latencies for seven cases. Effective token counts already include the batch dimension; each entry lists forward time above forward+backward time. RayRoPE fails on T3 and T5 because the reference implementation does not accept the corresponding multiframe shapes, so these entries remain missing. Each case uses 10 warm-up runs and 30 timed measurements. Complete CSV files, P10–P90 ranges, and throughput data are provided with the experiment artifacts.

Table 7: Median per-case model-block latency in milliseconds. Each cell lists forward time above forward+backward time; bold marks the minimum in each case and pass.

Case	Image size	B	T	Tokens	Standard RoPE	P-RoPE	RayRoPE	POD-RoPE (Q/K-only)	POD-RoPE
R0	128×160	1	1	240	4.907 **10.552**	13.395 39.429	41.789 102.995	**4.717** 12.028	7.234 16.496
R1	256×320	1	1	960	5.390 **11.581**	13.812 38.569	39.319 100.942	**4.577** 11.737	6.537 17.343
R2	336×448	1	1	1764	**6.037** **16.795**	14.019 38.312	41.616 105.185	6.624 17.461	7.736 19.718
B2	256×320	2	1	1920	**5.454** **15.653**	13.340 37.914	41.385 106.076	6.087 16.423	7.270 19.861
B4	256×320	4	1	3840	**9.099** **27.395**	16.545 47.976	48.219 107.367	9.938 28.333	11.840 32.045
T3	256×320	1	3	2880	**9.683** **28.712**	15.705 45.628	– –	10.394 29.611	11.894 32.890
T5	256×320	1	5	4800	**16.942** **55.506**	24.996 75.113	– –	17.874 56.632	20.176 62.116

B.2 Complete RE10K Metrics

Figure 3(b) in the main text uses forward-plus-backward latency from the matched RE10K/LVSM timing protocol as its speed axis. Table 8 supplies PSNR, SSIM, and LPIPS under the same 80k training protocol. These timings are reported separately from the reconstruction-block microbenchmarks above.

Table 8: LVSM novel-view synthesis on RE10K with 80k steps and $n = 6474$. Bold denotes the best result; pale blue identifies POD-RoPE variants.

Method	PSNR↑	SSIM↑	LPIPS↓
Plücker Raymap	23.442	0.7490	0.1390
GTA	24.308	0.7790	0.1260
P-RoPE	24.503	0.7838	0.1118
RayRoPE	**26.055**	**0.8312**	**0.0842**
POD-RoPE (Q/K-only)	24.825	0.7970	0.1020
POD-RoPE	25.223	0.8061	0.0994

C Policy Results on All RoboTwin 2.0 Tasks

Table 9 reports all 50 tasks under the same training budget and evaluation settings. Our method uses the 30k checkpoint; implementation details are given in Appendix E.4. Both groups evaluate 100 episodes per task and condition, with equal weighting across tasks.

Table 9: Fast-WAM success rates (%) on all 50 RoboTwin 2.0 tasks, using the methods and settings of Table 1.

Task	Baseline		Ours	
	Clean	Rand.	Clean	Rand.
adjust_bottle	100	100	100	100
beat_block_hammer	99	97	100	98
blocks_ranking_rgb	100	100	100	100
blocks_ranking_size	94	98	95	94
click_alarmclock	100	100	100	100
click_bell	100	100	100	100
dump_bin_bigbin	97	96	96	96
grab_roller	100	100	100	100
handover_block	95	81	92	82
handover_mic	99	100	99	99
hanging_mug	58	62	55	63
lift_pot	100	100	100	99
move_can_pot	90	88	94	94
move_pillbottle_pad	100	99	100	99
move_playingcard_away	100	100	100	100
move_stapler_pad	77	64	80	81
open_laptop	98	100	96	99
open_microwave	62	45	70	59
pick_diverse_bottles	80	85	83	85
pick_dual_bottles	100	96	100	97
place_a2b_left	95	93	99	98
place_a2b_right	93	99	99	98
place_bread_basket	91	93	95	95
place_bread_skillet	90	93	93	93
place_burger_fries	96	99	97	97
place_can_basket	71	69	68	70
place_cans_plasticbox	99	96	95	98
place_container_plate	96	100	99	100
place_dual_shoes	94	88	93	88
place_empty_cup	100	100	100	100
place_fan	96	96	96	97
place_mouse_pad	83	89	90	90
place_object_basket	89	88	90	86
place_object_scale	90	97	97	95
place_object_stand	90	94	92	95
place_phone_stand	97	99	95	98
place_shoe	96	99	96	97
press_stapler	90	97	97	94
put_bottles_dustbin	95	90	90	90
put_object_cabinet	94	89	92	92
rotate_qrcode	93	89	91	88
scan_object	89	92	93	93
shake_bottle	100	100	100	100
shake_bottle_horizontally	100	100	100	100
stack_blocks_three	95	97	96	96

Continued on next page

Task	Baseline		Ours	
	Clean	Rand.	Clean	Rand.
stack_blocks_two	100	100	99	100
stack_bowls_three	80	81	84	82
stack_bowls_two	92	98	97	97
stamp_seal	90	94	88	91
turn_switch	61	59	75	69
Average (50 tasks)	91.88	91.78	**93.12**	**92.64**
Δ (pp)	–	–	+1.24	+0.86

D Reconstruction under Input Changes

Shared protocol. We reuse the nine 2500-step checkpoints from Sharpa North, RoboTwin, and AGIBot, changing image processing or historical-frame sampling with fixed model weights and identical samples across methods. Every condition processes actual images, updates intrinsics K and actual times, and re-encodes them with the frozen Wan2.1 VAE, yielding 50,700 predictions. Evaluation includes latent MSE and PSNR, SSIM, and LPIPS against both context-consistent VAE-decoded targets and original RGB. This experiment changes the reference inputs, unlike conditional temporal extrapolation, which fixes the references and moves the future target.

Table 10: Input-change conditions. Each transformation updates effective intrinsics and actual times consistently.

Condition	Processing
base	Re-encode under this suite's baseline; target times and scoring differ from normal reconstruction and temporal $h0$.
res336	Resize from 256×320 to 336×448, changing pixel count and aspect ratio.
crop70; fov_mixed_a/b	Center-crop to 70% and resize, or apply camera-specific crop ratios; use only the available image area.
native_fov70/90/130	AGIBot only: rectify raw fisheye images to virtual pinhole views with 70/90/130-degree FOV.
stride5/8; irregular	Sample history at stride 5/8 or irregular intervals; use sufficiently long clips.
res336_stride5/8	Combine res336 with the corresponding history sampling stride.

Table 11 reports the PSNR difference POD-RoPE $-$P-RoPE under the same condition, rather than changes relative to each method's own base condition. Metrics are first averaged across tasks within each trajectory and then across trajectories; RoboTwin aggregates three target directions. The 95% intervals use trajectory-level paired bootstrap with 2000 resamples and reflect uncertainty across evaluation trajectories at fixed checkpoints. Sharpa North uses 50 trajectories. RoboTwin uses 166 sufficiently long trajectories for sampling conditions and all 271 for the remaining image conditions. AGIBot uses 20 trajectories. Each condition is compared with base on the same subset of trajectories, anchors, and target tasks.

Results. Under res336, PSNR, SSIM, and LPIPS improve over P-RoPE on all three datasets. Scoring against original RGB still gives PSNR differences of $+4.160$, $+3.321$, and $+2.473$ dB on Sharpa North, RoboTwin, and AGIBot, respectively. PSNR-difference intervals for cropping and mixed FOV are all above zero, but LPIPS is slightly worse on RoboTwin and does not improve under AGIBot's native 70-degree FOV. When only the sampling interval changes, POD-RoPE has lower PSNR than P-RoPE under stride5/8 and irregular sampling on both Sharpa North and RoboTwin. When combined with size changes, the PSNR differences become positive. These results support advantages under the tested image sizes and effective intrinsics; they do not establish improved adaptation to sampling changes alone or directly test generalization beyond the training camera-position envelope.

Table 11: PSNR differences for POD-RoPE −P-RoPE under all transfer conditions, with trajectory-level paired 95% confidence intervals. RoboTwin aggregates three target directions. Each transfer condition is paired with base on the same eligible subset. Dashes indicate undefined conditions.

Condition	Sharpa North	RoboTwin	AGIBot
base	+0.751 [+0.663, +0.837]	-0.594 [-0.687, -0.505]	+0.406 [+0.219, +0.567]
res336	+4.174 [+3.911, +4.453]	+3.328 [+3.101, +3.560]	+2.456 [+2.058, +2.803]
crop70	+1.898 [+1.775, +2.025]	+0.568 [+0.418, +0.725]	+1.442 [+1.258, +1.625]
fov_mixed_a	+2.507 [+2.392, +2.624]	+0.319 [+0.191, +0.450]	+1.300 [+1.078, +1.534]
fov_mixed_b	+1.388 [+1.274, +1.503]	+0.584 [+0.448, +0.725]	+1.580 [+1.365, +1.771]
stride5	-0.671 [-0.846, -0.509]	-0.582 [-0.700, -0.476]	+0.393 [+0.220, +0.551]
stride8	-0.694 [-0.871, -0.529]	-0.590 [-0.705, -0.486]	+0.386 [+0.211, +0.544]
irregular	-0.617 [-0.785, -0.450]	-0.572 [-0.689, -0.467]	+0.400 [+0.227, +0.556]
res336_stride5	+3.718 [+3.444, +4.003]	+2.750 [+2.495, +3.015]	+2.419 [+2.026, +2.749]
res336_stride8	+3.359 [+3.097, +3.636]	+2.368 [+2.142, +2.604]	+2.380 [+1.986, +2.708]
native_fov70	—	—	+0.779 [+0.550, +1.039]
native_fov90	—	—	+1.142 [+0.973, +1.316]
native_fov130	—	—	+1.347 [+1.093, +1.542]

Table 12: Experimental integration configurations for POD-RoPE. Q/K denotes distribution-based positional encoding and V/O denotes content transformations. L2 and RMSNorm precede the positional operator; input LN precedes projection.

Model / variant	Address	Bands	Scale	Norm.	V/O
Reconstruction	Space-time	$\mathcal{B}$	S	L2	Yes
π_0	Space-time	$\mathcal{B}$	S	None	Yes
RDT	Space-time	$\mathcal{B}$	S	Input LN	Yes
ACT	Space-time	$\mathcal{B}$	S	None	Yes
Fast-WAM (sim., vision)	Space-time	$\mathcal{B}$	S	Input LN	Yes
Fast-WAM (sim., action)	Sequence	A	A	Input LN	Yes
Fast-WAM (real, vision)	Space-time	$\mathcal{B}$	S	Input LN	Yes
Fast-WAM (real, action)	Sequence	A	A	Input LN	Yes
UniMatch (Q/K-only)	Spatial	L	U	L2	No
UniMatch (Q/K+V/O)	Spatial	L	U	L2	Yes
LVSM (Q/K-only)	Spatial	H	R	RMSNorm	No
LVSM (Q/K+V/O)	Spatial	H	R	RMSNorm	Yes

E Experimental Settings and Supplementary Analysis

E.1 Method Integration and Training Budgets

Policy integration and parameter overhead. POD-RoPE integrates into existing attention layers or through lightweight residual attention modules, depending on the policy architecture. For π_0 and ACT, positional and content transformations operate within the existing visual attention, requiring no additional adapters or trainable parameters. Fast-WAM in simulation and RDT use residual attention modules of width 192, adding 3.154M and 0.887M trainable parameters, respectively, equivalent to approximately 0.052% and 0.072% of their respective baselines' trainable parameter counts, excluding frozen encoders. The reported evaluations show higher average task success for all four policy architectures, supporting the practical use of POD-RoPE with zero or small parameter increments. The robot experiments evaluate the complete integrations, while the controlled ablations in Table 6 assess the contributions of distribution encoding and V/O transformations.

In Table 12, spatiotemporal addresses are $[f(\tau), \bar{o}, \bar{d}]$ and spatial addresses omit time, with $f(\tau) = 2\tanh(\tau/2)$. S uses $s_t = 0.125\,\mathrm{s}$, $s_o = 0.7504372976$, and $s_d = 103.01593647$; R uses the same numerical spatial scales, with camera centers expressed in reference-baseline-normalized units; S retains the units of its input camera centers. U directly uses $\bar{o} = 4.2127334\,o$ and $\bar{d} = 245.97024\,d$,

retaining the DeMoN input units for camera-center coordinates. S/R divide camera centers by s_o, while U multiplies them by the coefficient above.

The six-band set is $\mathcal{B} = \{0.0625, 0.125, 0.25, 0.5, 1, 2\}$, assigned cyclically to feature pairs. Let $P = d_h/2$ and $r = 0, \ldots, P-1$. L uses frequency magnitudes $10{,}000^{-r/(P-1)}$; H further multiplies them by $1 + h/H$, where $h = 0, \ldots, H-1$ is the head index. Fast-WAM's action branch A uses $10{,}000^{-r/P}$ and maps the nth action and its nine subinterval nodes δ_k to $2\tanh((n+\delta_k)/64)$, corresponding to a reference length of 32. This encodes action-sequence positions. Each visual distribution uses nine spatial nodes per source frame with $\sigma = 0.25$. All configurations with V/O enabled fix $g = 1$.

Complete frequency vectors. The address coordinates are ordered as time, camera-center xyz, and unit-ray xyz; static addresses omit time. For address dimension D, coordinate index $a = 1, \ldots, D$, and zero-based head and feature-pair indices h, r, we construct

$$\begin{aligned} u_{h,r,a} &= \sin\big((hP + r + 1)a\sqrt{2}\big) + \cos\big((hP + r + 1)a\sqrt{3}\big), \\ \omega_{h,r} &= b_{h,r}\, u_{h,r}/\|u_{h,r}\|_2, \end{aligned} \tag{22}$$

where $b_{h,r}$ is the magnitude specified by the frequency configuration above; for $\mathcal{B}$, $b_{h,r} = \mathcal{B}_{r \bmod 6}$ with zero-based indexing. The π_0 integration shares directions across heads, replacing $hP + r$ with r.

Temporal support and weights. In reconstruction, t_0 is the minimum timestamp over all reference and target support nodes in the current sample, and the temporal coordinate is $2\tanh((t-t_0)/(2s_t))$, with fixed $s_t = 0.125$ s. The first latent temporal plane uses raw frame 0; plane $l \geq 1$ uses raw frames $4l-3, \ldots, 4l$ with equal temporal weights. Each frame contributes Gaussian-9 spatial nodes. Joint weights are the products of temporal and spatial weights, normalized over the token's support.

Spatial scale calibration. On RE10K, $s_o = 0.7504372976$ is the median baseline length of selected training-view pairs, and $s_d = 103.01593647$ is the reciprocal of the median within-token Gaussian-weighted angular RMS of the rays. Angles are measured relative to the normalized weighted mean ray, using the training preprocessing and spatial nodes. RE10K translations are normalized by the reference-view baseline, so s_o is expressed in normalized baseline units. Other settings with the same numerical scales in Table 12 reuse these constants.

Table 13: Training budgets for visual experiments. Backbone entries list layers / width / heads / FFN width. The ablation and main reconstruction experiments use different backbones.

Experiment	Backbone	Global batch	Updates
Reconstruction	24 / 1152 / 8 / 4608	2,048	2,500
Component ablation	6 / 1152 / 8 / 1024	512	2,500
UniMatch	6 blocks / 144 ch. / 1 head	16	100,000
RE10K/LVSM	6 / 1152 / 8 / 1024	8	80,000

Attention operation order. Each model first projects Q/K, then applies the positional operator M, and finally computes scaled dot products. L2 normalization in Table 12 is followed by multiplication by $\sqrt{d_h}$; both L2 and RMSNorm precede M, while input LN precedes projection. ACT and π_0 do not separately normalize Q/K in the integration. The full method aggregates content as $G_j v_j$ and applies $G_i^\top$. Table 12 summarizes addresses, frequency bands, and scales for all integrations.

E.2 Embodied Reconstruction and Conditional Temporal Extrapolation

Models and data. On each of the three datasets, we separately train Standard RoPE (axial 3D), P-RoPE (Li et al., 2025), and a fixed POD-RoPE configuration (Gaussian-9, bounded temporal mapping, and Q/K+V/O). Table 13 gives the training budgets. Inputs are 256×320, Wan2.1 VAE is frozen, and the model receives a single latent plane. Training uses BF16, AdamW, a learning rate of 2×10^{-4}, and shared splits. Standard RoPE and POD-RoPE each have 382,409,856 parameters; P-RoPE has 382,416,768.

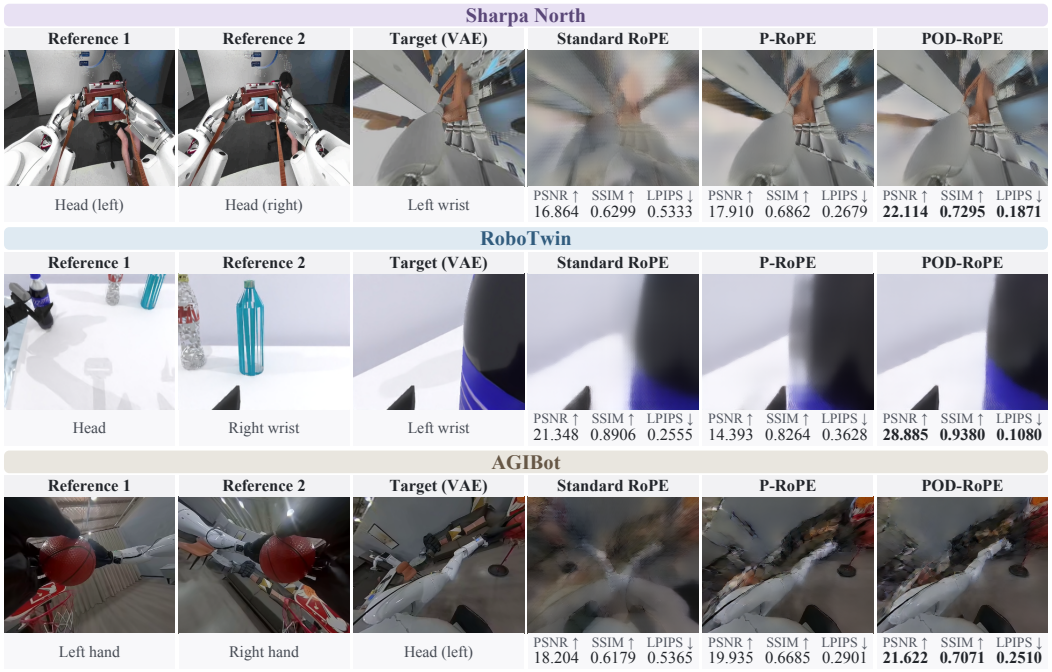


Figure 4: Synchronized cross-view reconstruction examples. Rows show Sharpa North, RoboTwin, and AGIBot from top to bottom. Each model reconstructs a third camera view from two simultaneous reference observations. Columns show Reference 1, Reference 2, the VAE-decoded target, and reconstructions from Standard RoPE, P-RoPE, and POD-RoPE. Metrics below the images are PSNR (dB), SSIM, and LPIPS, respectively; higher is better for the first two and lower is better for the last. Bold marks the best value for each example. Appendix E.2 specifies the scoring windows.

Table 14: Data and tasks for embodied reconstruction. Sample counts give the ordinary reconstruction / $h16$ evaluation totals per method. Splits are made by trajectory. Models are trained independently on each dataset, and all three methods share the split within a dataset.

Dataset	Train / val. trajectories	Eval. samples (normal / $h16$)	Target task
Sharpa North	281 / 50	200 / 200	Head–wrist cross-view
RoboTwin	2,212 / 271	1,800 / 1,728	Head and both wrists as targets
AGIBot	180 / 20	80 / 80	Head–hand cross-view

The datasets are, in order, our collected Sharpa North head-stereo and dual-wrist camera data, RoboTwin, and the public AgiBot World dataset (AGIBot) (Bu et al., 2025). Each task reconstructs a third view from two references; Table 14 gives camera roles and dataset sizes. Reconstructing one adjacent head camera from the other is not a primary Sharpa North task. RoboTwin alternates among head, left-wrist, and right-wrist targets and aggregates all three directions. AGIBot likewise does not use reconstruction between adjacent head cameras as a primary task. Clean/randomized records from the same source are grouped into one trajectory. Validation data are used for training monitoring and constitute held-out validation rather than an independent blind test.

Scoring and temporal protocol. Metrics are averaged across tasks within each trajectory and then across trajectories. We compute 95% intervals with trajectory-level paired bootstrap using 2000 resamples. PSNR, SSIM, and LPIPS use context-consistent VAE-decoded targets as references; latent MSE is reported separately. Conditional temporal extrapolation fixes the two reference observations, camera roles, and target trajectory, and uses actual future targets at $h \in \{0, 1, 2, 4, 8, 16\}$. One step spans four original synchronized sampling positions. Mean delays at $h16$ are 2.151, 4.267, and 2.133 seconds on Sharpa North, RoboTwin, and AGIBot, respectively; Table 14 gives sample counts. The predictor receives target camera geometry and time but does not access the future target latent. Target history is used only for shared contextual decoding and scoring. This protocol therefore evaluates conditional reconstruction with known target geometry. Images show the last frame; metrics cover one or four frames according to the latent position.

Results across temporal horizons. Figure 5 provides all temporal horizons. On RoboTwin, our method is 0.581 dB below P-RoPE at $h0$ in this suite; difference intervals at $h4/h8$ cross zero, while all four metrics lead at $h16$. LPIPS does not improve on AGIBot. Temporal changes should be compared against $h0$ within this suite.

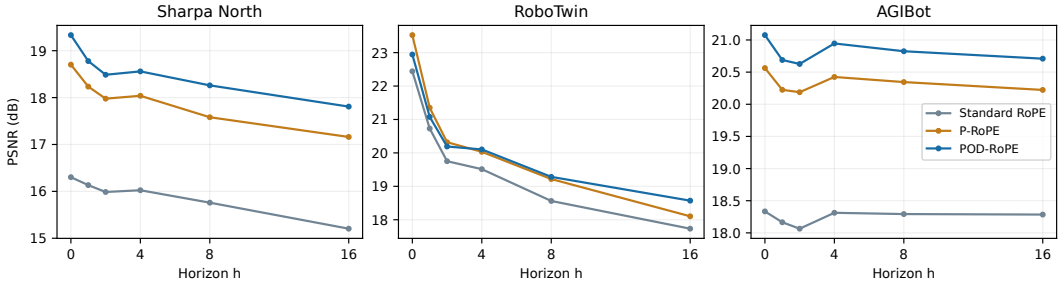


Figure 5: Temporal extrapolation to actual targets with fixed references, showing Sharpa North, RoboTwin, and AGIBot from left to right. All points use the same step-2500 checkpoint. Evaluated horizons are $h = 0, 1, 2, 4, 8, 16$; connecting lines are visual guides only. The same h corresponds to different actual delays across datasets. Curves show trajectory-balanced means; RoboTwin aggregates three target directions, as in the main table. Numerical results at $h16$ are reported in Table 4.

E.3 Single-Task Evaluation across Policy Architectures

On 15 RoboTwin 2.0 tasks, we follow each model's corresponding baseline training recipe and evaluation settings, independently training each model for each task. Each task uses 50 Clean demonstrations. ACT uses 40 training and 10 validation demonstrations; Table 15 gives the remaining recipes. Clean/Randomized correspond to Easy/Hard. Table 2 in the main text reports complete results.

Evaluation. For each task and environment, 100 scenes are selected using the native expert feasibility check. We use native qpos execution and task-success criteria. Clean uses seen instructions and Randomized uses unseen instructions. Success rates are averaged equally across the 15 tasks.

E.4 Fast-WAM Simulation Evaluation across All Tasks

Joint training on 50 tasks uses 27,500 trajectories, AdamW, a learning rate of 10^{-4}, and a 30k-step budget. Data splits, image decoding, per-sample language supervision, and action execution follow

Table 15: Single-task policy training and checkpoint selection. Both evaluation conditions use the same checkpoint.

Model	Global batch	Budget	Training / checkpoint	Execution
π_0	32	30,000 updates	4 GPUs, FSDP; final EMA	50 actions per call
RDT	64	10,000 updates	4 GPUs; final checkpoint	Native qpos
ACT	8	6,000 epochs	1 GPU; best validation	Chunk 50; temporal aggregation

the baseline pipeline. Each task runs 100 episodes under each of Clean/Randomized conditions, totaling 10,000 episodes. Results are first averaged equally over tasks, then equally over the two conditions. Both groups use identical training steps and evaluation episode counts; Appendix C gives per-task results.

E.5 Visual Geometry and Efficiency Evaluation

UniMatch/DeMoN. Five positional mechanisms are trained from scratch in UniMatch's (Xu et al., 2023) six Transformer blocks with single-head cross-image attention, using the combined DeMoN (Ummenhofer et al., 2017) protocol, 448×576 crops, eight GPUs, and a learning rate of 4×10^{-4}. Table 13 gives the budgets. Q/K-only uses six-dimensional addresses comprising camera origins and rays in world coordinates, Gaussian-9 with $\sigma = 0.25$, and scales calibrated on the training set. Q/K are normalized to $\sqrt{144}$ before the positional operator. Both variants use the same L frequency configuration; the full method additionally enables V/O with $g = 1$. RGBD and Scenes11 are evaluated with `validate_demon` at the same 100k checkpoint. We implement P-RoPE and RayRoPE (Li et al., 2025; Wu et al., 2026) from their published formulations and compare them under the same protocol.

RE10K. On RE10K (Zhou et al., 2018), we use the small LVSM (Jin et al., 2025) configuration with 256×256 inputs, Q/K normalization, CamRay, FP16, and eight-GPU training. Both POD-RoPE (Q/K-only) and full POD-RoPE use Gaussian-9, $\sigma = 0.25$, the same H frequency configuration, and calibrated spatial scales. The full method enables V/O with $g = 1$, without predicting or sampling scene depth. We report PSNR, SSIM, and LPIPS on a fixed set of 6,474 test samples.

Efficiency microbenchmarks. We benchmark the same six-layer reconstruction block on one H20 in BF16 using random calibrated rays and latents. Table 7 gives seven configurations, each with 10 warm-up runs and 30 measurements. Relative latency is the median of per-configuration ratios. RayRoPE is restricted to the five executable single-slice configurations. Figure 3(a) fixes 256×320, B1, T1 (960 tokens) and includes GTA under the same protocol. The quality–efficiency plot in Figure 3(b) uses a separate matched RE10K/LVSM timing protocol. Absolute latencies from the two settings are not mixed; neither includes the full data-loading, decoding, optimizer, or simulation pipeline.

E.6 Component Ablation

Using the Sharpa North four-camera cache and existing split, we fix the two head-camera inputs and reconstruct the left or right wrist view at the same time. The six groups share sampling and parameter initialization; Table 13 gives the backbone, with 46,168,704 parameters. Inputs are 256×320, Wan VAE is frozen, and the latent grid and patch size are $1 \times 32 \times 40$ and $1 \times 2 \times 2$. Training uses two GPUs with batch 256 per GPU, no gradient accumulation, BF16, AdamW, learning rate 2×10^{-4}, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.05, gradient clipping at 1.0, and 125 warm-up steps.

The five physical encodings share the frequency bands $\mathcal{B}$, scale S, tanh temporal mapping, and Q/K normalization in Table 12. Each source frame uses nine Gaussian nodes ($\sigma = 0.25$), with temporal nodes following the tokenizer footprint. Both point configurations first compute the weighted center of the same spatiotemporal addresses, then normalize the direction and restore its scale; the

time component is mapped before averaging. Distribution Q/K retains the expected rotation over all nodes with $g = 0$, and the full method changes only g to 1. The direct-average variant keeps the same distribution Q/K and replaces the V/O operators with $\overline{R}$ and $\overline{R}^{\top}$, respectively, omitting the identity branch. Standard RoPE uses a different axial 3D frequency layout, so component effects are compared within the five physical configurations by varying point versus distribution encoding and enabling V/O. All groups evaluate the same 200 validation samples at the 2500-step checkpoint, score against VAE-decoded targets, and take arithmetic means over samples, unlike the trajectory/task-balanced aggregation in the main table.

E.7 Real-Robot Evaluation on Sharpa North

Data and observations. Both methods train jointly on bottle opening and straw insertion, with task text distinguishing the operations. The two tasks have 343/1,351 training trajectories and 244,097/197,660 training windows, respectively, and 35/185 validation trajectories with 25,141/32,787 validation windows. Splits are made by collection batch, and training mixes valid windows. Preprocessing excludes 304 straw-insertion trajectories with abnormal mobile-base behavior (37,803 windows). Means and standard deviations are estimated only from deduplicated data rows in the retained training windows, with a standard-deviation floor of 0.001.

Images from the head's left eye and both wrists are first remapped to 256×320 and then assembled into a 384×320 input: the head image is on top, with two 128×160 wrist images side by side below. The head uses batch-specific intrinsics and distortion parameters; the wrists use an approximation based on a historical fisheye model. Both methods share this preprocessing. Camera links are mapped to robot root through URDF forward kinematics; this geometry is not an independently and precisely calibrated world coordinate system. Head timestamps define 33-frame windows. Video uses nine frames at offsets $0, 4, \ldots, 32$, while actions cover 32 steps. The 67-dimensional state/action comprises 7 dimensions per arm, 22 per hand, 7 for other joints, and 2 for mobile-base velocity. The policy reads only current observation geometry.

Models and training. Both methods share the pretrained Fast-WAM backbone, data, and identical width-192, six-head modules in the video and action branches. The baseline uses standard RoPE; POD-RoPE uses Gaussian-9 ($\sigma = 0.25$) for Q/K and V/O with $g = 1$. Visual tokens use robot-root addresses, actions encode sequence positions, and the original positional mechanism is retained.

Both groups use 8 nodes with 8 H20 GPUs per node, batch 16 per GPU, global batch 1024, no gradient accumulation, BF16, and ZeRO-1, with gradient checkpointing disabled. AdamW uses a learning rate of 10^{-4}, $\beta = (0.9, 0.95)$, weight decay 0.01, and gradient clipping at 1. A 100k-step cosine schedule includes 1500 warm-up steps; both real-robot evaluations use the 20k checkpoint.

Evaluation and examples. For bottle opening, the robot grasps the bottle and unscrews the cap; success requires the cap to be fully unscrewed and separated from the bottle opening. For straw insertion, the robot grasps the straw, aligns it with the target container opening, and inserts it; success requires the straw to remain inside the container after release. Each method is evaluated for 20 trials per task. Only trials that complete all task steps count as successes; incomplete trials count as failures. Success rates divide the number of successes by 20, and the mean weights tasks equally; Table 3 reports the results. Figure 2 uses original left-eye recordings from successful training demonstrations, with temporally selected frames, a consistent crop, and stage annotations to explain the tasks. The images are neither generated nor content-retouched.