ON THE MOVE: LATENCY-AWARE EMBODIED VISUAL TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Embodied visual tracking (EVT) is inherently latency-sensitive. While a robot computes its next action, the target keeps moving, so observations grow stale before actions take effect. Existing benchmarks, however, advance the environment only after the policy returns its action, charging nothing for decision time. We introduce RT-EVT, a latency-aware tracking benchmark that evaluates policies under configurable action latency. Accounting for deployment latency substantially changes tracking performance. For example, LightNav-0's tracking success rate drops from a SOTA 89.8% to 22.9% under 200 ms latency, even worse than a 5× smaller model. We propose TrackRT, a compact 0.9B vision-language navigation model designed for real-time tracking. We further introduce latency-aware training, implemented through action-effect-state supervision. The policy is trained against expert trajectories from the states in which its actions take effect. On the Jetson Orin NX, TrackRT achieves a mean inference latency of 145.0 ms versus 1201.0 ms for LightNav-0, with corresponding RT-EVT STT success rates of 65.3% and 0.5% respectively. Controlled ablations across different latency settings further confirm the benefit of latency-aware training.

1 INTRODUCTION

Embodied visual tracking (EVT) requires a robot to follow a language-designated person through a dynamic scene (Wang et al., 2025c). This task combines three capabilities. The robot must navigate around obstacles and passers-by (Krantz et al., 2020), remember target identity through appearance and order-of-appearance cues, and sustain following as people move or become occluded. Recent vision-language navigation models address these demands with temporal observations and pretrained multimodal representations (Zhang et al., 2025; 2026; Wang et al., 2026a). Several recent models and benchmarks directly evaluate EVT or include moving-target tracking as part of general-purpose navigation (Wang et al., 2025c; Liu et al., 2025; Lee et al., 2025; Zhang et al., 2025; 2026; Wang et al., 2026a;b).

EVT-Bench (Wang et al., 2025c), the standard benchmark for EVT, does not account for inference latency: the environment waits for the policy to return an action before advancing. In real deployment, however, the target continues moving during inference, so the action may take effect in a scene different from the one observed by the policy. We find that accounting for this latency substantially degrades tracking performance and can even reverse model rankings.

We therefore introduce RT-EVT, a latency-aware benchmark for embodied visual tracking. RT-EVT explicitly models the delay between observation and action execution and supports configurable and hardware-measured latency. Under RT-EVT, increasing latency from 0 to 200 ms reduces STT success from 89.8% to 22.9% for LightNav-0 and from 64.8% to 9.0% for OmTrackVLA. These results show that tracking performance and inference latency must be evaluated together.

To address this setting, we develop TrackRT, a compact 0.9B vision-language navigation model for latency-aware embodied tracking. It combines dual-tower encoding for richer semantics, a 0.5B VLN backbone, and single-pass trajectory prediction to reduce inference cost while preserving tracking history. We further introduce latency-aware training, implemented through action-effect-state supervision: request-time observations are paired with expert trajectories from the states in which the actions actually take effect. This aligns trajectory supervision with the scene after the inference delay.

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.

Experiments demonstrate the importance of both inference efficiency and latency-aware training. On the Jetson Orin NX, TrackRT reduces inference latency from 1201.0 ms to 145.0 ms and increases STT success from 0.5% to 65.3% under the corresponding RT-EVT delays. When all models are instead evaluated at the same 200 ms latency, TrackRT achieves 55.0/34.0/41.1% success on STT/DT/AT, outperforming all evaluated open-weight baselines and reversing the zero-latency ranking. Controlled ablations across latency settings further confirm the benefit of latency-aware training.

Our contributions are:

- We introduce RT-EVT, the first latency-aware benchmark for embodied visual tracking, with configurable action-effect latency, consistent target identity, and 4,215 evaluation episodes across three tracking tasks.
- We propose TrackRT, a compact vision-language tracking model with efficient temporal representations, direct trajectory regression, and latency-aware training.
- We conduct extensive experiments under common and onboard latency, revealing substantial performance degradation and ranking reversals. We will release the complete training and evaluation pipeline upon acceptance.

2 Related Work

2.1 Navigation and embodied tracking models

VLN-CE extends language navigation to continuous 3D environments with low-level action execution (Krantz et al., 2020). Recent generalist models further unify multiple navigation capabilities, including route following, object search, and moving-target tracking. Uni-NaVid learns navigation tasks from video input (Zhang et al., 2025), while NavFoM, VLingNav, Qwen-RobotNav, and the ABot series integrate multiple capabilities within a single model (Zhang et al., 2026; Wang et al., 2026b; Team, 2026; Chu et al., 2026; Gong et al., 2026). LightNav-0 explores compact generalist policies (Wang et al., 2026a), and SimpleNav provides a unified training and evaluation framework (SimpleNav Contributors, 2026). For embodied tracking, TrackVLA combines target recognition with trajectory generation and introduces EVT-Bench (Wang et al., 2025c). Subsequent trackers improve identity maintenance and dynamic interaction: TrackVLA++ introduces target-identity memory (Liu et al., 2025), OmTrackVLA provides an open model and implementation (Lee et al., 2025), and CoMaTrack strengthens target–distractor interactions through competitive multi-agent training (Gao et al., 2026). Unlike fixed-destination navigation, embodied tracking requires continuous target identification and adaptation to target motion. Recent work has also begun to report deployment latency. For example, Qwen-RobotNav reports average end-to-end latencies of 196 ms on a remote server and 204 ms on a Jetson Thor with FP8 optimization (Team, 2026). However, these tracking scores are still measured with step-synchronized evaluation, where inference time does not advance the environment.

2.2 Dynamic environments and non-blocking navigation

HA-VLN introduces dynamic pedestrians and social-interaction constraints into language navigation (Dong et al., 2026). DynaNav evaluates navigation with moving crowds, physical control, and inference latency, while TIC-VLA uses stale semantic features together with latency information and current observations for fast control (Huang et al., 2026). SPARK-VLN further compares environments that pause during inference with those that continue to advance (Hu et al., 2026). Other systems mitigate inference latency through asynchronous execution. LiveVLN overlaps action execution with observation processing (Wang et al., 2026c), while AsyncVLA uses an edge adapter to correct actions from a workstation-scale model with the latest observation (Hirose et al., 2026). Despite considering dynamic environments or inference latency, these methods navigate toward a fixed goal that does not move during inference. RT-EVT instead considers a moving target whose position and visibility can change while the policy computes.

2.3 LATENCY-AWARE METHODS

RTC handles inference latency by conditioning on actions already committed during the wait and generating the remaining action sequence (Black et al., 2025a), while TT-RTC incorporates this conditioning into training with latency-dependent action prefixes (Black et al., 2025b). VLASH aligns robot states and action labels with future execution times while retaining earlier visual observations (Tang et al., 2026), and TIDAL similarly pairs earlier visual semantics with later robot states and actions (Sun et al., 2026). For navigation, TIC-VLA injects latency into semantic inference and combines stale features with current observations, robot states, and explicit latency information (Huang et al., 2026). Existing methods compensate for latency using committed actions, future states, or fresh observations. In contrast, our latency-aware training directly supervises request-time observations with trajectories from action-effect states.

3 RT-EVT: MEASURING TRACKING UNDER DEPLOYMENT LATENCY

3.1 LANGUAGE-CONDITIONED CONTINUOUS TRACKING

Given a target description ℓ and the RGB observations $I_{\le t}$ acquired by the robot up to time t, the policy predicts a local trajectory

$$\widehat{\mathbf{W}}_t = \pi_\theta(\ell, I_{\le t}), \qquad \widehat{\mathbf{W}}_t \in \mathbb{R}^{K\times 3}. \tag{1}$$

Each waypoint contains a planar position and heading (x, y, ψ) in the robot's local frame; we use $K = 8$. A controller executes local motion along the predicted trajectory, after which the policy replans from new observations. The task requires the robot to keep facing and following the language-designated person while maintaining an appropriate distance.

RT-EVT follows EVT-Bench's three task families—Single-Target Tracking (STT), Distracted Tracking (DT) and Ambiguity Tracking (AT)—while preserving their task semantics (Wang et al., 2025c). Task definitions and split statistics are provided in Appendix B.1.

3.2 REPAIRING TARGET IDENTITY AND EVALUATION CONSISTENCY

Reliable latency comparisons require the instruction and the instantiated target to agree in every episode. EVT-Bench switches person assets only at first initialization, allowing later episodes to retain a previous person's appearance (Wang et al., 2025b). This breaks the instruction–person correspondence, particularly in DT, and makes evaluation depend on episode ordering and worker sharding. RT-EVT configures persons per episode before environment construction and verifies loaded assets and semantic IDs against episode metadata; inconsistent instances abort with a logged reason. In an episode-level audit of the three task families, 4,017 of 4,215 original episodes (95.3%) had a target-avatar mismatch, and RT-EVT restored consistency in all of them (Appendix A.3).

3.3 LATENCY-AWARE EVALUATION PROTOCOL

RT-EVT injects latency between observation acquisition and action effect. An observation acquired at logical time t_k produces an action $\mathbf{u}_k$. The action is executed immediately when the latency expires, at t_k^{act}; until then, the environment evolves while the robot retains its previous velocity command. We parameterize the delay by inference latency δ_{inf} and additional transport overhead δ_{com} (Rupam Mahmood et al., 2018; Kronauer et al., 2021). The total delay is rounded upward to a logical scheduling grid of $\Delta = 10\,\mathrm{ms}$:

$$\delta = \delta_{\mathrm{inf}} + \delta_{\mathrm{com}}, \qquad \bar{\delta} = \Delta \left\lceil \frac{\delta}{\Delta} \right\rceil, \qquad t_k^{\mathrm{act}} = t_k + \bar{\delta}. \tag{2}$$

The protocol supports zero-latency evaluation, common-delay comparisons, and deployment-aligned comparisons using the target device's measured inference latency. We set $\delta_{\mathrm{com}} = 0$ in the reported experiments, excluding communication overhead. Appendix A.1 provides the scheduling timeline and scheduler implementation.

3.4 Core Observation: Ranking Reversal under Deployment Latency

We conduct a controlled experiment to examine how inference latency affects the relative performance of embodied visual trackers. We evaluate the same set of models on STT under zero-latency ("Ideal") and deployment-aware ("Reality") settings, where actions are delayed according to the measured inference latency on an RTX 4090 and Orin NX. Ours denotes TrackRT, while Ours+ further optimizes the execution runtime for onboard inference. Detailed latency measurements are reported in Table 1. Figure 1 shows a clear ranking reversal under deployment latency. LightNav-0 achieves the highest SR under zero-latency evaluation, followed by TrackRT and OmTrackVLA. After incorporating measured inference latency into the control loop, TrackRT achieves the highest SR on both devices, whereas LightNav-0 drops from first to last.

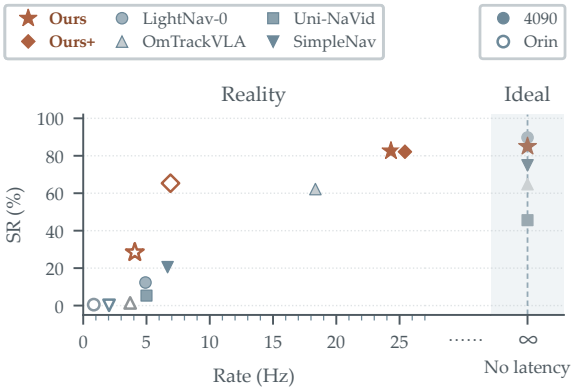

Figure 1: STT success rate (SR) under zero-latency and deployment-aware evaluation.

Thus, deployment latency does more than degrade tracking performance. It also changes the relative ranking of the models, leading to conclusions that differ from those drawn under zero-latency evaluation. These results motivate deployment-aware evaluation and trackers that remain robust to delayed actions. We address the latter through the compact architecture and latency-aware training introduced in the next section.

4 TrackRT: A Compact Model with Latency-Aware Training

Tracking under latency requires both efficient decisions and supervision that accounts for the wait before actions take effect. Figure 2 presents these complementary parts of TrackRT. The architecture compresses visual history and directly predicts local trajectories to reduce decision cost. Four stages of base training establish visual-language alignment and trajectory prediction; a fifth stage then trains on expert trajectories from action-effect states, accounting for robot and target motion during the wait.

4.1 Bounding the Cost of Each Decision

Figure 2a shows the architecture of TrackRT. The model is built around MiniCPM4-0.5B as the VLN backbone and uses DINOv3 ViT-S/16 and SigLIP So400m as dual visual encoders, for 0.9B parameters in total. DINOv3 provides spatial features, while SigLIP provides vision-language semantics (Siméoni et al., 2025; Zhai et al., 2023). Their features are concatenated and projected into the language backbone. To reduce the cost of temporal context, we pool each history frame into 4 visual tokens while retaining 64 tokens for the current frame. We refer to this token allocation as asymmetric coarse-to-fine temporal tokenization (ACTT). A window of 31 history frames and the current frame therefore contains only 188 visual tokens. History features are cached, so each decision encodes only the latest frame.

Single-pass trajectory prediction. Following the line of compressing task and observation information through dedicated queries (Octo Model Team et al., 2024), we append one learnable control query $\mathbf{q}$ at the end of the sequence. With input context $\mathbf{X}_t$ and the control query's final hidden state $\mathbf{h}_t$, the predicted trajectory is

$$\mathbf{h}_t = \mathrm{Backbone}([\mathbf{X}_t; \mathbf{q}])_{\mathbf{q}}, \qquad \widehat{\mathbf{W}}_t = \mathrm{reshape}_{8\times 3}(\tanh(g_\phi(\mathbf{h}_t)))\,\mathrm{diag}(2,2,1), \tag{3}$$

where the trajectory head g_ϕ is a funnel MLP. Each prediction contains eight future waypoints spaced 100 ms apart (10 Hz), independently of RT-EVT's 10 ms action-effect scheduling grid (Section 3.3).

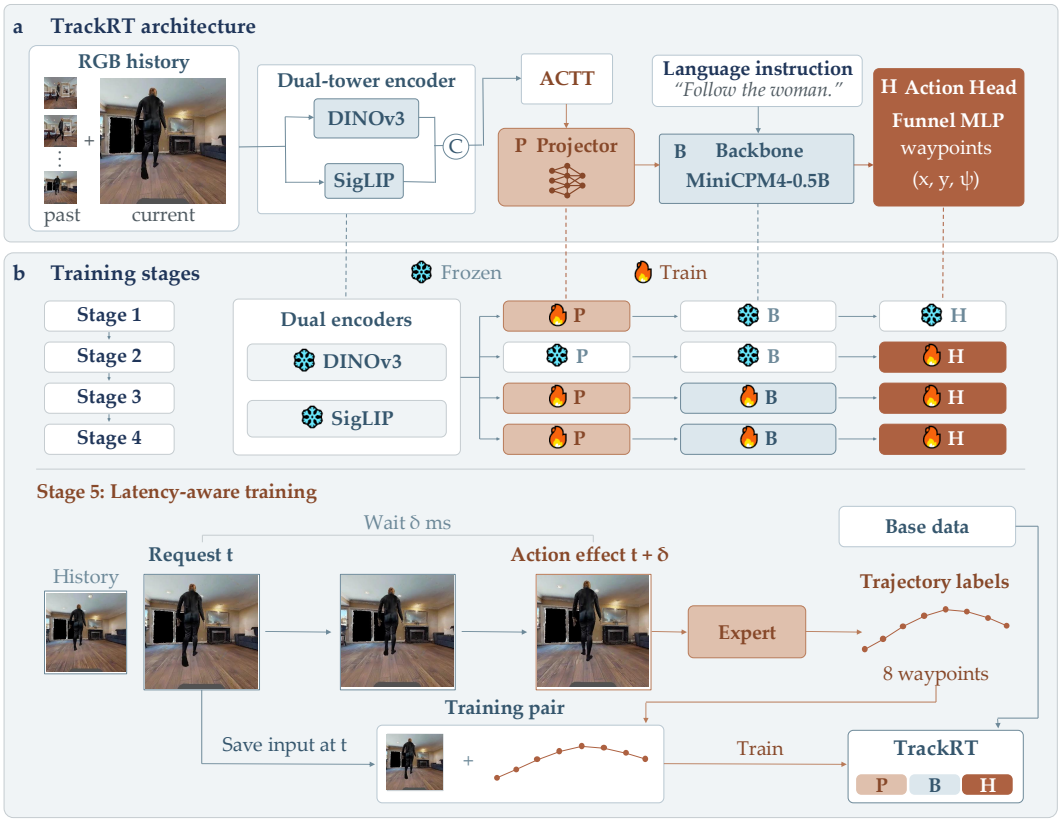

Figure 2: TrackRT architecture and training. (a) Dual-tower visual features and language tokens are combined in the MiniCPM4-0.5B backbone, followed by a funnel MLP that predicts eight future waypoints. (b) Four base-training stages are followed by latency-aware training, which pairs request-time observations with expert trajectories labelled at the action-effect state.

A complete action requires only one backbone forward pass and one regression. In contrast, autoregressive methods require repeated backbone passes, while diffusion- and flow-based heads require multiple denoising steps (Black et al., 2026). In our ablation, a 12-step DiT head takes 21.6 ms, compared with only 0.2 ms for our single-pass MLP, with SR gains limited to AT (Section 5.4). After execution, the robot re-observes and replans, so the benefit of the history representation is tested in continuous feedback.

4.2 Latency-Aware Training via Action-Effect-State Supervision

Base training. We first train TrackRT for embodied tracking using dynamic trajectories collected in Habitat (Savva et al., 2019; Szot et al., 2021) with HM3D (Ramakrishnan et al., 2021) and MP3D (Chang et al., 2017) scenes. As shown in Figure 2b, Stages 1–2 train the projector (P) and trajectory head (H), while Stages 3–4 jointly fine-tune the projector, backbone (B), and trajectory head. The visual encoders remain frozen throughout. Data construction and base training are detailed in Appendices B.1–B.4.

Action-effect-state supervision. Base training associates each observation with a trajectory for its capture-time state. Under inference latency, however, the robot and target continue moving before the predicted action takes effect, creating a mismatch between observation and execution. To address this mismatch, we sample latencies uniformly from $\{0, 50, 100, 150, 200\}$ ms during interactive rollouts. At request time t, we save the visual-language input $\mathbf{X}_t = (\ell, I_{\leq t})$ and let the environment advance during the sampled delay. The expert then generates trajectory labels $\mathbf{W}^*(s^{\delta}_{t+\bar{\delta}})$ from the resulting action-effect state. Pairing these action-effect-state labels with the request-time input trains the policy to predict trajectories appropriate for when its actions actually execute.

Joint training. With $\mathcal{D}_\delta$ denoting the relabeled interaction data, we optimize

$$\mathcal{L}_{\text{delay}} = \mathbb{E}_{\delta\sim p_{\text{train}}}\, \mathbb{E}_{(\mathbf{X}_t, s^{\delta}_{t+\delta})\sim\mathcal{D}_\delta}\left[\mathcal{L}_{\text{traj}}\Big(\pi_\theta(\mathbf{X}_t), \mathbf{W}^*(s^{\delta}_{t+\bar{\delta}})\Big)\right], \qquad \mathcal{L}_{\text{train}} = \mathcal{L}_{\text{base}} + \lambda\mathcal{L}_{\text{delay}}. \tag{4}$$

We mix the relabeled data with base supervision and jointly update P, B, and H while keeping the visual encoders frozen. Delayed rollouts are re-collected as the policy improves to cover states visited by the updated policy. The resulting model learns from request-time observations while being supervised for the states reached after inference delay.

Training protocol. The relabeled data comprise roughly 20% of the base supervision volume and are collected in two rounds. The complete stage-wise schedules and shared optimization settings are listed in Appendix B.5; sample construction and matched training controls are detailed in Appendix B.6.

5 Experiments

5.1 Implementation and Evaluation setup

We evaluate all policies on RT-EVT using a unified action scheduling protocol. Policy-specific adapters preserve each model's native action semantics. All comparisons use identical person configurations, reference episode lengths, and a fixed split of 1,405 episodes per task family.

Baselines. We primarily compare TrackRT with two open-weight VLN baselines: OmTrackVLA and LightNav-0. OmTrackVLA uses a compact 0.6B backbone, comparable in scale to the 0.5B backbone of TrackRT (Lee et al., 2025). LightNav-0 achieves the state-of-the-art STT and DT success rates on the original EVT-Bench among publicly available models and is evaluated with vLLM (Wang et al., 2026a). SimpleNav and Uni-NaVid report lower STT/DT success rates than LightNav-0 despite using equal or larger backbones (SimpleNav Contributors, 2026; Zhang et al., 2025). We therefore exclude them from the primary comparison, while reporting supplementary Uni-NaVid and SimpleNav results in Tables 17 and 18. Other models listed in Table 9 have not publicly released the tracking checkpoints. Their published EVT-Bench results are retained for context in that table. Appendix C.1 details baseline selection and checkpoint availability.

Tasks, Metrics and Latency settings. We include three common tracking tasks: Single-Target Tracking (STT), Distracted Tracking (DT), and Ambiguity Tracking (AT). Following EVT-Bench, we report success rate (SR), tracking rate (TR), and collision rate (CR) using its original metric definitions and aggregation rules (Wang et al., 2025c;a). We evaluate action latency over a common range of 0–270 ms. This range covers all measured RTX 4090 configurations and the Orin NX MAXN configurations of TrackRT and OmTrackVLA. Appendix A.2 provides the full definitions.

5.2 Tracking performance on RT-EVT

Tracking performance deteriorated with increasing action latency across the three models in Figure 3, panels 1(a)–1(c). The sharp SR drop between 100 and 110 ms, circled in panel 1(a), is consistent with the 10 Hz discretization of predicted trajectories (Section 4.1). At delays just beyond 100 ms, the robot can travel farther than the distance represented by the first predicted waypoint without reaching the second. Its actual progress then falls between two predicted trajectory points, creating a mismatch between the discrete waypoint predictions and motion execution.

TrackRT retained the highest SR at every tested delay from 110 to 270 ms, and its SR declined more slowly than either baseline. It also achieved the highest TR throughout the 0–270 ms sweep. At 200 ms, its TR was 66.9%, compared with 26.6% for LightNav-0 and 27.6% for OmTrackVLA; its CR was 13.4%, versus 16.9% and 46.2%, respectively. Across the sweep, TrackRT's CR remained below OmTrackVLA's and close to LightNav-0's.

We also compared policies at a common 200 ms latency (Figure 3, panel 2). Holding the delay constant tests whether TrackRT's advantage persists when faster inference provides no timing benefit. Device-specific comparisons then use the measured P50 values in Table 1.

At 200 ms, TrackRT achieved SR of 55.0%, 34.0%, and 41.1% on STT, DT, and AT. These exceeded the strongest baseline, LightNav-0, by 32.1, 16.9, and 31.3 percentage points, respectively. The

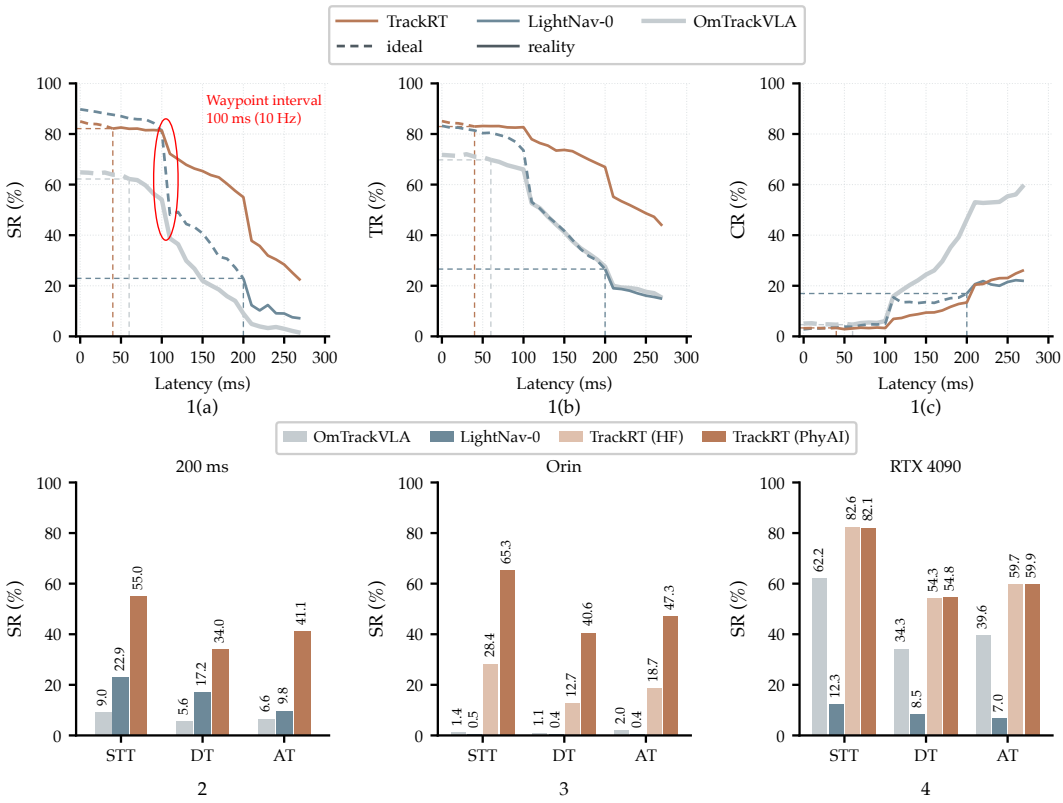

Figure 3: Tracking performance on RT-EVT. (1) Performance across 0–270 ms latency. (2) Performance at a common 200 ms latency. (3) Performance on Orin. (4) Performance on RTX 4090.

Table 1: Online inference latency of open-weight trackers (ms). MAXN denotes Orin NX 16 GB mode 0 (MAXN_SUPER) with locked clocks; its measurements use 100 requests after 10 warmups. Runtime and timing details are in Appendix C.2.

Model / configuration	**RTX 4090**			**Orin NX MAXN**		
	Mean	**P50**	**P95**	**Mean**	**P50**	**P95**
Uni-NaVid	201.47	200.90	225.05	—	—	—
SimpleNav (BATS cache)	142.88	142.35	151.31	469.52	484.86	521.38
OmTrackVLA	54.72	54.50	55.34	269.58	269.59	270.14
LightNav-0 (Transformers)	217.43	216.59	217.68	1,731.19	1,742.83	1,752.14
LightNav-0 (vLLM)	203.76	203.27	205.71	1,200.96	1,200.53	1,221.59
TrackRT (Transformers)	41.78	41.12	**44.05**	245.37	245.33	246.09
TrackRT (PhyAI)	**39.47**	**39.33**	53.55	**145.00**	**144.98**	**145.32**

advantage under equal delay shows that lower inference cost alone does not explain TrackRT's performance. Accounting for latency also reversed the ranking within RT-EVT: LightNav-0 led all three tasks at zero latency, whereas TrackRT led at 200 ms. The complete SR/TR/CR sweeps across all three tasks are provided in Appendix D.1.

5.3 Latency Performance

We used P50 measurements from the Orin NX 16 GB fitted to our Unitree Go2 EDU and from an RTX 4090 to define device-specific RT-EVT delays. These conditions follow the scheduling rule in Section 3.3 and exclude communication delay. Table 1 reports the measurements; timing boundaries and runtime configurations are detailed in Appendix C.2.

Table 2: Core architecture ablations. Each alternative changes one factor of the reference: dual encoders, 31 history frames, ACTT and a funnel MLP. Scores are SR (%) on RT-EVT at zero latency; Table 19 provides the complete metrics. A100 model-path and head-only timings are reported separately; — denotes unmeasured timing.

Variant	SR↑ (%)			Visual tokens	P50 (ms)↓	
	STT	DT	AT		Model path	Head only
Reference	84.98	55.87	60.50	188	84.098	0.212
Visual encoders						
DINOv3 only	72.7	42.8	51.7	188	33.215	—
SigLIP only	75.9	54.1	55.2	188	77.910	—
History window						
No history	76.7	45.2	41.8	64	—	—
Temporal token allocation						
All fine	69.9	40.6	47.9	2,048	116.893	—
All coarse	73.5	52.9	62.1	128	83.530	—
Action head						
DiT (v, 12 steps)	82.1	54.4	60.8	188	—	20.464
DiT (x, 12 steps)	80.6	55.3	63.3	188	—	21.636

Orin NX. With MAXN_SUPER and locked clocks, TrackRT with PhyAI and OmTrackVLA reached P50 latencies of 144.98 and 269.59 ms. These map to 150 and 270 ms, respectively, for the three-task comparison in Figure 3, panel 3. At these injected delays, TrackRT achieved SR of 65.3/40.6/47.3% on STT/DT/AT, compared with 1.4/1.1/2.0% for OmTrackVLA. LightNav-0's vLLM model-output P50 was 1,200.53 ms; the corresponding 1,210 ms RT-EVT condition, which includes action decoding, reached 0.5/0.4/0.4%.

RTX 4090. The RTX 4090 measurements map to 40 ms for TrackRT with PhyAI, 50 ms for its HF backend, 60 ms for OmTrackVLA, and 210 ms for LightNav-0 with vLLM (Figure 3, panel 4). At their respective delays, TrackRT with PhyAI achieved SR of 82.1/54.8/59.9% on STT/DT/AT, compared with 62.2/34.3/39.6% for OmTrackVLA and 12.3/8.5/7.0% for LightNav-0. Its HF backend at 50 ms reached 82.6/54.3/59.7%, showing similar task scores at a modestly higher delay. TrackRT therefore led all three task families when the comparison incorporated each model's measured inference cost.

Onboard inference optimization. On the same device, PhyAI reduced TrackRT's mean latency from 245.37 to 145.00 ms, giving a 1.69× speedup over Transformers. P50 fell from 245.33 to 144.98 ms, shifting the RT-EVT condition from 250 to 150 ms. Accordingly, TrackRT SR increased from 28.4/12.7/18.7% with HF at 250 ms to 65.3/40.6/47.3% with PhyAI at 150 ms on STT/DT/AT. This comparison combines backend speed with the resulting action timing. Appendix C.2 provides stage timings and paired waypoint-output checks.

We also validated TrackRT on a Unitree Go2 in a controlled indoor corridor (Appendix C.3). With the target, instruction and motion setup fixed, reducing end-to-end latency from approximately 245 ms with HF to 145 ms with PhyAI increased trial success from 15/30 (50.0%) to 22/30 (73.3%) across straight following, 90° turns and person crossing, supporting the latency–success relationship observed in RT-EVT.

5.4 Architecture Ablations: History versus Computation Cost

Table 2 summarizes four architecture choices under zero-latency RT-EVT evaluation. Measurement settings are given in Appendix C.4. Dual encoding performed best among the visual-encoder variants on all three tasks.

History and temporal token allocation showed similar accuracy–efficiency trade-offs. Using 31 history frames improved performance over no history on all three tasks, with the largest gain on AT (18.7 percentage points). However, the full sweep in Table 19 shows that this benefit was not monotonic, as three history frames performed worse than no history. Compared with all-fine encoding,

Table 3: Training ablation under injected latency. A uses standard training without delayed rollouts. B and C use the same delayed rollouts, but obtain supervision from the request state and action-effect state, respectively. Each cell reports SR/TR/CR (%).

Task	Mode	Injected Delay (ms)				
		0	**50**	**100**	**150**	**200**
STT	A: base	**86.3/86.3/4.1**	**84.3/85.1/3.9**	**82.8/83.2/5.3**	55.8/66.8/18.0	39.9/56.0/25.5
	B: request	72.9/79.8/5.1	72.0/79.8/5.0	62.2/74.9/6.0	30.6/49.6/20.4	18.0/36.0/39.2
	C: effect	85.0/85.1/3.2	82.6/83.2/2.8	81.4/82.7/3.3	**65.3/73.7/9.4**	**55.0/66.9/13.4**
DT	A: base	**60.5/70.0/16.1**	**59.4/69.4/16.2**	**56.8/66.7/19.0**	34.5/49.2/33.8	24.0/39.8/42.5
	B: request	42.5/61.1/8.0	41.9/60.5/7.3	38.6/58.7/10.5	22.7/40.4/23.8	13.6/29.0/40.6
	C: effect	55.9/67.4/12.9	54.3/65.3/13.1	52.8/64.0/13.6	**40.6/54.7/22.1**	**34.0/49.0/26.5**
AT	A: base	**64.2/75.3/10.3**	**63.7/75.1/10.2**	**61.9/72.6/13.1**	39.8/55.5/30.5	28.7/47.0/39.5
	B: request	47.7/68.5/5.1	48.0/69.3/5.0	45.9/66.7/7.3	27.3/49.0/19.6	15.2/34.4/40.8
	C: effect	60.5/73.6/8.6	59.7/72.3/8.6	60.1/72.2/9.5	**47.3/63.1/18.7**	**41.1/58.3/23.8**

ACTT used 90.8% fewer visual tokens and reduced model-path P50 by 32.8 ms while improving performance across tasks. All-coarse encoding performed better on AT. Thus, ACTT balanced task success and computation.

The action-head ablation further highlights the trade-off between performance and generation cost. Both DiT variants improved AT performance over the funnel MLP but required roughly 100$\times$ its head-only latency. The MLP performed best on STT and DT among the three heads, supporting its use when generation cost matters. Complete history, token-budget, and temporal/stream-marker ablations are reported in Appendix D.4.

5.5 Effect of Latency-Aware Training

Table 3 compares three training settings under injected delays from 0 to 200 ms. We compare standard training without delayed rollouts (A), delayed training using labels from the request state (B), and delayed training using labels from the action-effect state (C). All variants use the same architecture and training budget.

Base supervision (A) achieves the highest zero-latency SR but degrades sharply at 150–200 ms as its labels become stale. Request-state supervision (B) performs worse than A at every delay, reaching only 18.0%, 13.6%, and 15.2% SR at 200 ms on STT, DT, and AT, respectively. In contrast, action-effect-state supervision (C) incurs only a modest zero-latency SR cost over A (1.3, 4.6, and 3.7 percentage points), while outperforming A at both 150 and 200 ms on all three tasks. At 200 ms, C improves SR over A by 15.1, 10.0, and 12.4 percentage points on STT, DT, and AT, respectively. The matched B–C comparison indicates that action-effect-state labels better align request-time visual history with the trajectory required after the delay, improving delayed tracking while largely preserving zero-latency performance.

6 Conclusion

In this work, we introduce RT-EVT to quantify the impact of inference latency on embodied visual tracking, a representative dynamic task in vision-language navigation. We also develop TrackRT, a compact 0.9B model that combines efficient temporal representations with single-pass trajectory prediction. Its latency-aware training pairs request-time observations with expert trajectories from action-effect states. At a common 200 ms delay on RT-EVT, TrackRT achieves higher SR than all evaluated baselines across STT, DT and AT, reversing the zero-latency ranking. It also outperforms OmTrackVLA and LightNav-0 under simulated delays derived from RTX 4090 and Jetson Orin NX measurements. Controlled ablations further support the benefit of action-effect-state supervision for delayed tracking. RT-EVT and TrackRT expose the latency-induced mismatch between benchmark evaluation and deployment conditions, providing a framework to quantify and address it.

AI USE STATEMENT

We used generative AI tools to edit the manuscript for readability, identify relevant literature, format references, and debug software code used for analysis and figure generation. The tools were not used to generate synthetic datasets, formulate mathematical claims, design the methodology or experiments, implement the proposed method, or interpret the experimental results. The authors reviewed and verified all AI-assisted text, references, code changes, and outputs, and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Upon acceptance, we will release the complete training and evaluation pipeline, including the RT-EVT evaluation code, model interfaces, training configurations, deployment timing scripts, per-episode results and figure-generation entry points. The release will include the checkpoints used for the reported results. Scene splits, person configurations and data statistics are given in Appendices B.1–B.2; the scheduler implementation and identity-repair validation are given in Appendices A.1 and A.3, respectively. Evaluation hardware, backend, power mode and deterministic evaluation settings are fixed and recorded for each run.

REFERENCES

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 33383–33407. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1122. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/300ccb2187dedd4edcc07f7e76d8e553-Paper-Conference.pdf.

Kevin Black, Allen Z. Ren, Michael Equi, and Sergey Levine. Training-time action conditioning for efficient real-time chunking, 2025b. URL https://arxiv.org/abs/2512.05964.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pp. 667–676, 2017. doi: 10.1109/3DV.2017.00081.

Zedong Chu, Shichao Xie, Xiaolong Wu, Yanfen Shen, Minghua Luo, Zhengbo Wang, Fei Liu, Xiaoxu Leng, Junjun Hu, Mingyang Yin, Jia Lu, Yingnan Guo, Kai Yang, Jiawei Han, Xu Chen, Yanqing Zhu, Yuxiang Zhao, Xin Liu, Yirong Yang, Ye He, Jiahang Wang, Yang Cai, Tianlin Zhang, Li Gao, Liu Liu, Mingchao Sun, Fan Jiang, Chiyu Wang, Zhicheng Liu, Hongyu Pan, Honglin Han, Zhining Gu, Kuan Yang, Jianfang Zhang, Di Jing, Zihao Guan, Wei Guo, Guoqing Liu, Di Yang, Xiangpo Yang, Menglin Yang, Hongguang Xing, Weiguo Li, and Mu Xu. Abot-n0: Technical report on the vla foundation model for versatile embodied navigation, 2026. URL https://arxiv.org/abs/2602.11598.

Yifei Dong, Fengyi Wu, Qi He, Lingdong Kong, Heng Li, Minghan Li, Zebang Cheng, Yuxuan Zhou, Jingdong Sun, Qi Dai, Alexander G Hauptmann, and Zhi-Qi Cheng. Ha-vln 2.0: An open benchmark and leaderboard for human-aware navigation in discrete and continuous environments with dynamic multi-human interactions, 2026. URL https://arxiv.org/abs/2503.14229.

Li Gao, Liu Liu, Mingyang Lv, and Yang Cai. Comatrack: Competitive multi-agent game-theoretic tracking with vision-language-action models. In Paolo Favaro, Zuzana Kukelova, Atsuto Maki, Anna Rohrbach, Konrad Schindler, and Federico Tombari (eds.), *Computer Vision – ECCV 2026*, pp. 249–268, Cham, 2026. Springer Nature Switzerland. ISBN 978-3-032-37574-2.

Ruiyan Gong, Yingnan Guo, Junjun Hu, Jintao Kong, Xiaoxu Leng, Tianlun Li, Weize Li, Fei Liu, Zhicheng Liu, Jia Lu, Minghua Luo, Chenlin Ming, Yanfen Shen, Jiyue Tao, Zhengbo Wang, Mingyang Yin, Minqi Gu, Zihao Guan, Wei Guo, Guoqing Liu, Huachong Pang, Menglin Yang, Zeqian Ye, Xiaoxiao Geng, Zhining Gu, Honglin Han, Di Jing, Hongyu Pan, Mingchao Sun, Kuan Yang, Jianfang Zhang, Yanghong Chen, Ye He, Wei Mei, Jiahao Shi, Xiangpo Yang, Yanqing Zhu, Yang Cai, Jingjing Ma, Shihui Su, Zixiao Tang, Linbo Zheng, Zedong Chu, Xiaolong Wu, Wenbin Tang, and Mu Xu. Abot-n1: Toward a general visual language navigation foundation model, 2026. URL https://arxiv.org/abs/2607.10383.

Noriaki Hirose, Catherine Glossop, Dhruv Shah, and Sergey Levine. Asyncvla: An asynchronous vla for fast and robust navigation on the edge, 2026. URL https://arxiv.org/abs/2602.13476.

Tianshuai Hu, Yangyi Zhong, Zeying Gong, Lingdong Kong, Xiaodong Mei, Guoyang Zhao, Xiaolu Liu, Song Wang, Rong Li, and Junwei Liang. Token-wise latent streaming from slow reasoners to fast planners for dynamic vision language navigation, 2026. URL https://arxiv.org/abs/2607.16806.

Zhiyu Huang, Yun Zhang, Johnson Liu, Rui Song, Chen Tang, and Jiaqi Ma. Tic-vla: A think-in-control vision-language-action model for robot navigation in dynamic environments. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2026.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, pp. 104–120, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58603-4. doi: 10.1007/978-3-030-58604-1_7. URL https://doi.org/10.1007/978-3-030-58604-1_7.

Tobias Kronauer, Joshwa Pohlmann, Maximilian Matthé, Till Smejkal, and Gerhard Fettweis. Latency analysis of ros2 multi-node systems. In *2021 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pp. 1–7, 2021. doi: 10.1109/MFI52462.2021.9591166.

Kyusong Lee, Heting Ying, and Tiancheng Zhao. Omtrackvla: Open-source visual language action model for visual navigation and following. https://github.com/om-ai-lab/OmTrackVLA, 2025.

Light Origins Team. Serving LightNav-0 on NVIDIA Jetson AGX Thor. https://github.com/lightorigins/LightNav-0/blob/main/docs/JETSON_THOR.md, 2026. Official deployment documentation; accessed 2026-09-19.

Jiahang Liu, Yunpeng Qi, Jiazhao Zhang, Minghan Li, Shaoan Wang, Kui Wu, Hanjing Ye, Hong Zhang, Zhibo Chen, Fangwei Zhong, Zhizheng Zhang, and He Wang. Trackvla++: Unleashing reasoning and memory capabilities in vla models for embodied visual tracking, 2025. URL https://arxiv.org/abs/2510.07134.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/34173cb38f07f89ddbebc2ac9128303f-Paper-round2.pdf.

A. Rupam Mahmood, Dmytro Korenkevych, Brent J. Komer, and James Bergstra. Setting up a reinforcement learning task with a real-world robot. In *2018 IEEE/RSJ International Conference*

on Intelligent Robots and Systems (IROS), pp. 4635–4640. IEEE Press, 2018. doi: 10.1109/IROS.2018.8593894. URL https://doi.org/10.1109/IROS.2018.8593894.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9338–9346, 2019. doi: 10.1109/ICCV.2019.00943.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL https://arxiv.org/abs/2506.01844.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3, 2025. URL https://arxiv.org/abs/2508.10104.

SimpleNav Contributors. Simplenav: Make navigation VLA simple, 2026. URL https://github.com/OpenBMB/SimpleNav.

Yuteng Sun, Haoran Wang, Ruofei Bai, Zhengguo Li, Jun Li, Meng Yee Michael Chuah, and Wei Yun Yau. Tidal: Temporally interleaved diffusion and action loop for high-frequency vla control, 2026. URL https://arxiv.org/abs/2601.14945.

Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimír Vondruš, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra. Habitat 2.0: Training home assistants to rearrange their habitat. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 251–266. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/021bbc7ee20b71134d53e20206bd6feb-Paper.pdf.

Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. Vlash: Real-time vlas via future-state-aware asynchronous inference, 2026. URL https://arxiv.org/abs/2512.01031.

Qwen Team. Qwen-robotnav technical report: A scalable navigation model designed for an agentic navigation system, 2026. URL https://arxiv.org/abs/2606.18112.

Shaoan Wang, Jiazhao Zhang, Minghan Li, Jiahang Liu, Anqi Li, Kui Wu, Fangwei Zhong, Junzhi Yu, Zhizheng Zhang, and He Wang. TrackVLA/EVT-Bench official aggregation: TR aggregation with reference-episode-length compensation in analyze_results.py. https://github.com/ShaoanWang/TrackVLA, 2025a.

Shaoan Wang, Jiazhao Zhang, Minghan Li, Jiahang Liu, Anqi Li, Kui Wu, Fangwei Zhong, Junzhi Yu, Zhizheng Zhang, and He Wang. TrackVLA/EVT-Bench official task implementation: person switching and first-initialization condition in track_task.py. https://github.com/ShaoanWang/TrackVLA, 2025b. Checked 2026-09-10.

Shaoan Wang, Jiazhao Zhang, Minghan Li, Jiahang Liu, Anqi Li, Kui Wu, Fangwei Zhong, Junzhi Yu, Zhizheng Zhang, and He Wang. Trackvla: Embodied visual tracking in the wild. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 4139–4164. PMLR, 27–30 Sep 2025c. URL https://proceedings.mlr.press/v305/wang25f.html.

Shaoan Wang, Aocheng Luo, Fei Huang, Jingyi Xu, Xiaoyang Wang, Yueyu Wang, Qianli Ma, Fan Yang, Ran Mei, Jia Wei, Jiangpeng Hu, Xuhao Liu, Hongming Chen, Yuanbin Shao, Yiyang Lin,

Ziliang Li, Liang Pan, Xinhang Liu, Yuntao Ma, and Tingxiang Fan. Lightnav-0: Eliciting vlm spatial intelligence for generalist embodied navigation, 2026a. URL https://arxiv.org/abs/2608.30935.

Shaoan Wang, Yuanfei Luo, Xingyu Chen, Aocheng Luo, Dongyue Li, Chang Liu, Sheng Chen, Yangang Zhang, and Junzhi Yu. Vlingnav: Embodied navigation with adaptive reasoning and visual-assisted linguistic memory, 2026b. URL https://arxiv.org/abs/2601.08665.

Xiangchen Wang, Weiye Zhu, Teng Wang, Tiantian Geng, Zekai Zhang, Zhiyuan Qi, Jinyu Yang, and Feng Zheng. Livevln: Reducing stop-and-go loops in vision-language navigation. *IEEE Robotics and Automation Letters*, pp. 1–8, 2026c. doi: 10.1109/LRA.2026.3734886.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11975–11986, October 2023.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *Robotics: Science and Systems*, 2025.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, Yuxin Fan, Wenjun Li, Zhibo Chen, Fei Gao, Qi Wu, Zhizheng Zhang, and He Wang. Embodied navigation foundation model. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 127293–127322, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/ceb45c48e29e7f49b0b47edb98e43691-Paper-Conference.pdf.

A RT-EVT Implementation and Validation

A.1 Time scales and scheduler implementation

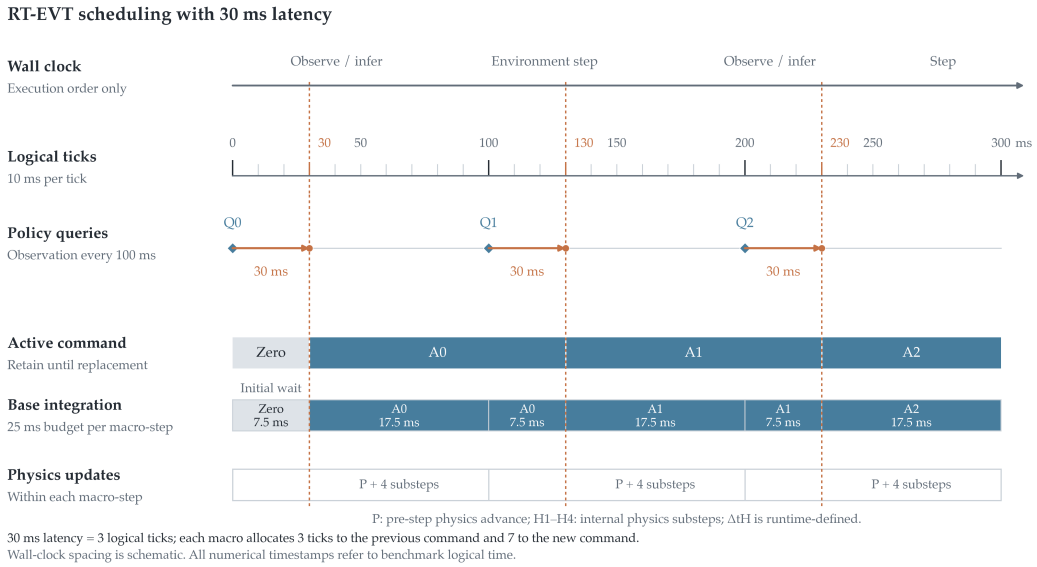

Figure 4: RT-EVT scheduling with a 30 ms action latency. Policy queries at 0, 100, and 200 ms produce commands that take effect at 30, 130, and 230 ms; each command remains active until its replacement arrives. The wall-clock row indicates execution order only; numerical timestamps refer to logical time. Lower rows show the implementation's command allocation and physics updates.

The formal entry point `evaluate_rt_evt_split` calls `_evaluate_rt_evt_unified_episode`, which uses `step_macro` to implement the schedule in Figure 4. Each macro-step executes one original `environment.step`. Logical ticks determine command switching within that step, while observations and scene states are updated through the original environment interface.

The injected latency is the sum of `inference_delay_ms` and `transport_delay_ms`; the measured duration of `policy.act` is stored independently. When using a total measurement that covers communication, the communication portion within its timing scope is counted once.

A.2 EVT-Bench metric definitions and aggregation

We use EVT-Bench's original success and collision criteria and tracking-rate aggregation (Wang et al., 2025c;a).

Let a task split contain N episodes, with episode e executing T_e environment steps. Let $d_{e,t}$ be the Euclidean distance between the robot and the target person's base, and $v_{e,t} \in \{0, 1\}$ the geometric facing test (the dot product between the robot's heading and the robot-to-target direction exceeds 0.5). The per-step effective-tracking indicator is

$$f_{e,t} = \mathbb{I}[d_{e,t} \le 3\,\mathrm{m}]\, v_{e,t}. \tag{5}$$

Success is defined by how the episode terminates. With H the episode step cap (currently 300) and z_e the environment stop flag,

$$S_e = \begin{cases} z_e\, \mathbb{I}[1\,\mathrm{m} \le d_{e,T_e} \le 3\,\mathrm{m}]\, f_{e,T_e}, & T_e < H, \\ f_{e,T_e}, & T_e = H. \end{cases} \tag{6}$$

The collision flag is set to 1 when the robot–target distance first drops below 0.5 m and persists to episode end: $C_e = \mathbb{I}[\min_{1 \le t \le T_e} d_{e,t} < 0.5\,\mathrm{m}]$.

The official EVT-Bench aggregation script compensates early termination with reference episode lengths: with T_e^{ref} available, $L_e = \max(T_e, T_e^{\mathrm{ref}})$; otherwise $L_e = T_e$. Using these reference lengths, we report the three metrics as percentages:

$$\mathrm{SR} = \frac{100}{N}\sum_{e=1}^{N} S_e, \qquad \mathrm{TR} = 100\,\frac{\sum_{e=1}^{N}\sum_{t=1}^{T_e} f_{e,t}}{\sum_{e=1}^{N} L_e}, \qquad \mathrm{CR} = \frac{100}{N}\sum_{e=1}^{N} C_e. \tag{7}$$

RT-EVT model comparisons fix one set of reference lengths and matching rules while also storing actual execution lengths.

A.3 Person-identity repair validation

The identity audit used a nominal 30-split configuration. We used the default configurations of both EVT-Bench and RT-EVT throughout. Because the 4,215 episodes do not divide evenly, execution produced 31 splits after remainder handling. We checked every episode in each task family and classified the original implementation by whether the first episode was aligned, a subsequent episode retained the same avatar and remained aligned, or the target avatar disagreed with the episode metadata.

Table 4: Episode-level target-avatar consistency audit. The fixed evaluation set contains 1,405 episodes per task family. Under `seed=100`, `shuffle=True`, and a nominal 30-split configuration (31 executed splits after remainder handling), each episode was classified by the original implementation. The final column contains episodes whose target avatar disagreed with episode metadata; RT-EVT restored metadata-consistent targets for all such episodes.

Task	First episode correct	Subsequent same avatar, still correct	Avatar mismatch
STT	31	42	1,332
DT	31	35	1,339
AT	31	28	1,346
Total	**93**	**105**	**4,017**

Across the three task families, 4,017 of 4,215 episodes (95.3%) had a target-avatar mismatch in the original implementation. RT-EVT restored the metadata-consistent target avatar in all of these episodes; the remaining 198 episodes were already aligned under the two correct-alignment categories in Table 4.

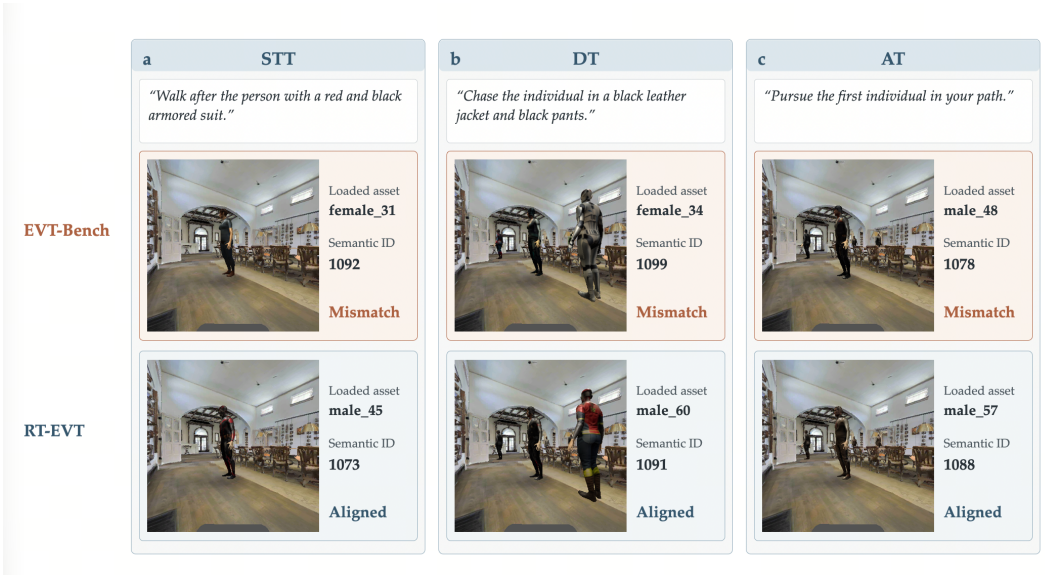

Figure 5: Per-episode person-identity repair in RT-EVT. Panels a–c show STT, DT, and AT examples, respectively, for episode 164 in scene `2n8kARJN3HM`. Each column pairs the instruction with first-frame views from EVT-Bench (top) and RT-EVT (bottom). Labels identify the loaded person asset and semantic ID. In these examples, the original implementation loads an identity inconsistent with the episode metadata; RT-EVT restores the specified asset and semantic ID. These are qualitative identity-alignment examples; aggregate validation is specified in Table 4.

A.4 Delayed-decision formalization

For the RT-EVT scheduler, let s_n contain the full dynamic state of robot, persons, and controller; c_n the currently effective command; q_n the pending command and its remaining wait; and φ_n the scheduling phase and timeout counters. The augmented state is

$$z_n = (s_n, c_n, q_n, \varphi_n), \qquad z_{n+1} \sim \widetilde{P}(\cdot \mid z_n, u_n). \tag{8}$$

Under the assumption that environment dynamics and latency sampling depend on the current augmented state, this representation describes a Markov transition of the full system: persons evolve at environment updates, and pending actions are determined by queue and timing state. The RGB-language policy receives only partial observations, and a finite visual history is not automatically a Markov state, so the policy in fact faces a partially observed decision problem with action delays. This modeling separates "labeling actions from the request state" from "labeling actions from the action-effect state," motivating the expert relabeling of Section 4.2. Future states are used only for training supervision; deployment inputs still end at the request time, without per-latency optimality guarantees.

B Data Construction and Training Details

B.1 Scene and identity splits

RT-EVT retains EVT-Bench's three task definitions. STT measures basic following under simple instructions. DT tests language-based target identification among crossing distractors using fine-grained appearance descriptions. AT tests temporal identity maintenance when visually identical distractors are identified by order of appearance.

Manifest statistics below are per task family. STT, DT and AT share an equally sized base geometric episode set, on which task-specific persons and instructions are configured.

Table 5: Base task splits. Each task uses the following base geometric episode sets. The person library contains 100 humanoid appearance assets: STT/DT use 70 training and 30 non-overlapping validation identities, and AT uses 30 identities with target and distractors sharing appearance within an episode.

Use / split	HM3D sc.	MP3D sc.	Total sc.	HM3D ep.	MP3D ep.	Total ep.
Training: train	660	43	703	6,295	962	7,257
Evaluation: val	87	14	101	684	721	1,405

Evaluation uses the `val` split, disjoint from training scenes. Each task family contains 1,405 evaluation episodes across 101 scenes, for 4,215 episodes in total.

The person library contains 100 humanoid appearance assets. STT/DT use 70 training target identities and 30 non-overlapping validation identities; AT uses 30 identities, with target and distractor sharing appearance within an episode and distinguished by relative order of appearance.

B.2 Collected data volume

Table 6: Collected data volume. Collection statistics and the data actually used by the final checkpoint are recorded separately.

Task	Episode JSONL files	Supervision samples	Referenced RGB frames
STT	8,919	878,074	949,426
DT	6,913	718,347	773,651
AT	2,841	204,102	226,830
Total	18,673	1,800,523	1,949,907

Raw supervision samples split roughly 48.8%/39.9%/11.3% across STT/DT/AT. JSONL counts describe collection output and the base manifest describes task instances; training sampling, success-trajectory filtering, and the final checkpoint's actual data are recorded in the training configuration.

B.3 Velocity-to-waypoint conversion

With the current robot pose as the local origin, planar velocity $(v_{x,k}, v_{y,k})$ and angular velocity ω_k are integrated at $\Delta t = 0.1\,\mathrm{s}$:

$$\begin{aligned} x_{k+1} &= x_k + (v_{x,k}\cos\psi_k - v_{y,k}\sin\psi_k)\Delta t, \\ y_{k+1} &= y_k + (v_{x,k}\sin\psi_k + v_{y,k}\cos\psi_k)\Delta t, \\ \psi_{k+1} &= \psi_k + \omega_k \Delta t. \end{aligned} \tag{9}$$

Eight future waypoints are sampled from the integrated trajectory, with a validity mask handling insufficient labels at the tail. This conversion interval belongs to training supervision; RT-EVT's action-scheduling granularity is set independently.

With normalized predictions and labels $\widehat{\mathbf{w}}'_i$ and $\mathbf{w}'_i$ and valid-waypoint set $\mathcal{V}$, the trajectory loss of Section 4.2 is

$$\mathcal{L}_{\text{traj}} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \|\widehat{\mathbf{w}}'_i - \mathbf{w}'_i\|_2^2 . \tag{10}$$

B.4 Base training stages

Table 7 summarizes the per-stage update scopes of the base training pipeline (Section 4.2).

Table 7: Base training stages: update scope, supervision, and purpose.

Stage	Update scope and supervision	Purpose
Vision-language alignment	Freeze both visual encoders and the language backbone; train only the projector with a language-modeling loss on description tokens over image-description pairs, one epoch	Align visual features to the language hidden space
Action-head initialization	Freeze visual encoders, projector, and language backbone; train only the action head	Establish the initial mapping from multimodal representations to local trajectories
Joint policy fine-tuning	Keep visual encoders frozen; jointly update projector, language backbone, and action head	Adapt representation and trajectory prediction to tracking
Interaction-driven refinement	Collect failure scenarios from simulated and real interaction, add quality-controlled and relabeled data, retrain and re-evaluate	Extend coverage of hard scenarios and long-tail interactions

B.5 Training hyperparameters

The stage-wise learning-rate schedules and shared optimizer settings used for the reported model are given in Table 8.

Table 8: Training hyperparameters. The first block gives the stage-wise schedules used for the reported model. LR denotes the initial learning rate; the cosine schedule decays each stage to the listed minimum LR ratio, with 3% warm-up. The second block lists settings shared across stages.

Stage	Epochs	Global batch	Backbone LR	Projector LR	Action head LR	Warm-up	Min LR ratio
Stage 1: Projector alignment	1	64	0	1×10^{-4}	0	3%	0.1
Stage 2: Action-head initialization	1	64	0	0	1×10^{-4}	3%	0.1
Stage 3: Joint fine-tuning	2	64	2×10^{-5}	1×10^{-4}	1×10^{-4}	3%	0.1
Stage 4: Interaction-driven refinement	1	32	2×10^{-5}	1×10^{-4}	1×10^{-4}	3%	0.1
Stage 5: Latency-aware fine-tuning	1	32	2×10^{-5}	1×10^{-4}	1×10^{-4}	3%	0.1

Shared setting	Value
Optimizer	AdamW
Adam betas	$(0.9,\ 0.95)$
Weight decay	0.01
Learning-rate schedule	Cosine decay
Gradient clipping	1.0
Gradient accumulation	1
Precision	BF16
Gradient checkpointing	Enabled

B.6 Latency-aware training: sample construction and controls

During each interaction, we sample a delay from $\{0, 50, 100, 150, 200\}$ ms and retain the request-time visual-language input. The expert labels the trajectory from the action-effect state reached after the wait. The resulting pairs are mixed with base supervision; the projector, backbone and trajectory head are updated while the visual encoders remain frozen. Two collection rounds produce relabeled data amounting to roughly 20% of the base supervision volume, with delayed rollouts re-collected as the policy improves. The loss weight is $\lambda = 1$ in the reported model.

The training ablation fixes architecture, task splits, sample counts and total update budget. Conditions B and C share the collected delayed rollouts and request-time inputs, differing only in whether the expert labels the request state or the action-effect state. Condition A uses base trajectory supervision. Table 3 reports this comparison.

C Evaluation and Deployment Settings

C.1 Baseline selection and published reference results

Our two primary baselines, OmTrackVLA and LightNav-0, provide tracking checkpoints and inference code for matched RT-EVT evaluation. Our audit of official project releases on 24 September 2026 found no publicly downloadable tracking checkpoints for the following models:

- **TrackVLA**: the public repository releases EVT-Bench and a Uni-NaVid evaluation interface, but does not provide the TrackVLA policy checkpoint (Wang et al., 2025c).
- **TrackVLA++, NavFoM and VLingNav**: the official project pages describe the methods and reported results without providing downloadable tracking checkpoints (Liu et al., 2025; Zhang et al., 2026; Wang et al., 2026b).
- **ABot-N0 and Qwen-RobotNav**: the public releases provide reports or benchmark resources without the corresponding tracking weights (Chu et al., 2026; Team, 2026).

These resource limits prevent matched reruns with repaired target identities and injected action latency. SimpleNav and Uni-NaVid provide open weights, but report lower STT/DT success rates than LightNav-0 (SimpleNav Contributors, 2026; Zhang et al., 2025; Wang et al., 2026a). SimpleNav uses a 4B backbone, matching LightNav-0's scale, whereas Uni-NaVid uses a larger 7B backbone. We therefore selected LightNav-0 for the primary comparison. SimpleNav's higher STT tracking rate does not change this selection based on success rate. Published scores remain in Table 9, and Tables 17 and 18 retain our supplementary Uni-NaVid and SimpleNav evaluations.

Table 9 retains the published EVT-Bench results; these scores provide context across released models, whereas our matched comparisons use the shared RT-EVT protocol (Appendix D.2).

Table 9: Published EVT-Bench results and a repaired RT-EVT reference (SR↑/TR↑/CR↓, %). Backbone size refers to the pretrained language or multimodal backbone; for TrackRT, the 0.9B total includes the visual encoders and action head, whereas the MiniCPM4 backbone is 0.5B. Checkpoints, training data, and evaluation implementations differ across models, so the table serves as a capability reference rather than a strictly fair ranking; "—" marks values not reported officially. Values come from each model's paper or official repository, compiled on 2026-09-11; LightNav-0's README reports STT/DT SR and TR only.* The TrackRT row uses the current repaired RT-EVT evaluation at 0 ms because a separately reported original-EVT-Bench run is unavailable.

Model	Backbone size	STT	DT	AT
Open weights				
Uni-NaVid (Zhang et al., 2025)	7B	53.3/67.2/12.6	31.9/50.1/21.3	15.8/41.5/26.5
OmTrackVLA (Lee et al., 2025)	0.6B	81.41/82.77/5.13	41.54/58.80/11.31	60.04/73.89/7.60
LightNav-0 (Wang et al., 2026a)	4B	91.7/87.7/—	82.6/80.1/—	—
SimpleNav (SimpleNav Contributors, 2026)	4B	82.8/93.5/1.2	45.2/76.7/6.0	39.7/79.2/3.8
TrackRT (this work)*	0.5B	84.98/85.09/3.20	55.87/67.41/12.88	60.50/73.58/8.61
Closed or unverified weights				
TrackVLA (Wang et al., 2025c)	7B	85.1/78.6/1.65	57.6/63.2/5.80	50.2/63.7/17.1
NavFoM (Zhang et al., 2026)	7B	85.0/80.5/—	61.4/68.2/—	—
TrackVLA++ (Liu et al., 2025)	7B	86.0/81.0/2.10	66.5/68.8/4.71	51.2/63.4/15.9
VLingNav (Wang et al., 2026b)	7B	88.4/81.2/2.07	67.6/73.5/5.51	—
ABot-N0 (Chu et al., 2026)	4B	86.9/87.6/8.54	66.7/75.4/11.6	67.3/79.5/7.05
Qwen-RobotNav-4B (Team, 2026)	4B	77.4/90.0/6.40	—	—
Qwen-RobotNav-8B (Team, 2026)	8B	78.6/89.7/5.70	—	—

C.2 Deployment measurement details

Hardware selection. We selected the Jetson Orin NX 16 GB fitted to our Unitree Go2 EDU to represent onboard deployment. For workstation deployment, we used an RTX 4090, following TrackVLA, NavFoM, TrackVLA++, and VLingNav (Wang et al., 2025c; Zhang et al., 2026; Liu et al., 2025; Wang et al., 2026b). This pairing tests latency-aware tracking across the compute budgets available onboard and on a workstation.

Table 10: TrackRT latency on Jetson Orin NX in MAXN_SUPER mode. Online timings cover RAM RGB input through waypoint output after history initialization and 10 warmup requests. Stage means come from an independent replay; stage percentiles are not summed.

Path / scope	Mean (ms)	P50 (ms)	P95 (ms)	Requests
Online single-frame update				
PhyAI, RGB → waypoints	145.00	144.98	145.32	100
Transformers, RGB → waypoints	245.37	245.33	246.09	100
PhyAI stages, independent single-frame replay				
DINOv3 ViT-S/16	22.83	—	—	100
SigLIP So400m	86.63	—	—	100
Policy including action head	32.90	—	—	100

MAXN measurement protocol. The Orin NX runs in mode 0, `MAXN_SUPER`, with `jetson_clocks --fan`: CPU 1,984 MHz, GPU 1,173 MHz, and EMC 3,199 MHz. A real 10 FPS tracking video is pre-decoded to 201 frames of 384×384 RGB. After history initialization, requests advance continuously through the frames; 10 warmup requests precede 100 timed single-frame updates at batch size 1. Timing includes RAM image preprocessing, host-to-device transfer, model execution, and output processing, with CUDA synchronization at the boundaries. Video decoding, disk reads, model loading, networking, camera frame age, and robot control are outside this scope. Stage timings use a separate 100-request replay; PhyAI's main measurement disables internal timing, while its stage replay uses CUDA events. The optional complete-window measurement uses 10 warmups and 50 samples, and measures warmed window recomputation rather than model cold start.

Both MAXN environments use Python 3.10.12, PyTorch 2.10.0, torchvision 0.25.0, CUDA 12.6, and cuDNN 9.6.0. OmTrackVLA and TrackRT's Transformers backend use Transformers 4.57.0; PhyAI and LightNav-0 use Transformers 5.8.0. Weight SHA256 checks match the earlier checkpoints, including DINOv3 ViT-S/16. OmTrackVLA and TrackRT retain 31 history frames plus the current frame. LightNav-0 uses 16 frames at 256×448, post-ViT pooling factor 2, and its native 4 FPS relative-time convention. Its Transformers path re-encodes the window, while vLLM 0.19.1 records seven tubelet-cache hits and one miss per update.

Independent stage measurements for TrackRT give mean latencies of 22.83 ms for DINOv3, 86.63 ms for SigLIP, and 32.90 ms for the policy including its action head. Table 10 reports the end-to-end mean, P50, and P95.

RTX 4090 measurements. The latest PhyAI run uses 100 consecutive requests after 10 warmups, with internal timing enabled (PyTorch 2.11.0, CUDA 12.8, cuDNN 9.19.0). The container uses the observed GPU boost state. Table 1 reports mean, P50 and P95 timings for seven model–configuration combinations; P50 determines the corresponding injected delay. Uni-NaVid uses FP16 with online single-frame rolling updates. SimpleNav uses its official BATS rolling cache. Device, runtime, and timing-instrumentation differences are recorded separately from the MAXN measurements.

C.3 Real-world validation on Unitree Go2

We conducted a controlled real-world validation of TrackRT on a Unitree Go2 in an indoor office corridor. The target wore a black top and white shorts, and the instruction was *"follow the first*

person you see." The target walked at approximately $0.7\,\mathrm{m\,s^{-1}}$; this moderate speed was chosen because locomotion-induced camera shake became more pronounced at higher speeds. We evaluated straight following, a 90° turn and person crossing, with 10 trials per scenario for each backend. For person crossing, a distractor wearing a white top and black trousers crossed the robot's field of view from approximately 45° behind the target. The robot, target appearance, instruction, environment and motion setup were fixed across backends; only the inference backend and resulting action latency changed. The HF and PhyAI backends had approximate latencies of 245 ms and 145 ms, respectively.

Table 11: Controlled real-world validation of TrackRT. Each backend was evaluated in 10 trials for each scenario.

Scenario	HF	PhyAI	HF failure modes	PhyAI failure modes
Straight following	8/10	10/10	2 target collisions from delayed stopping	None
90° turn	4/10	7/10	4 wall collisions; 2 missed turns	2 wall collisions; 1 missed turn
Person crossing	3/10	5/10	7 identity switches to distractor	5 identity switches to distractor
Overall	**15/30 (50.0%)**	**22/30 (73.3%)**	–	–

The uncut onboard deployment video contains two person-crossing events and one 90° turn in a continuous run.

C.4 Architecture-ablation protocol

Architecture variants are evaluated on RT-EVT at zero latency, with one factor varied per block and all other settings fixed. Backbone-scale comparisons fix visual input, data splits and training budget. Efficiency measurements use an A100-SXM4-80GB at batch size 1. Model-path P50 includes the model's full inference path; action-head comparisons instead report head-only latency. Table 19 distinguishes these timing scopes and repeats the reference configuration within each block.

C.5 Timing scopes in prior robot systems

The physical observation-to-command interval is $D_k^{\mathrm{cmd}} = t_k^{\mathrm{accept}} - t_k^{\mathrm{cap}}$, measured from each request's capture and command-acceptance timestamps. SmolVLA separates observation transmission, server inference, and action return (Shukor et al., 2025), whereas RTC expresses inference delay in controller steps (Black et al., 2025a). Table 12 records reported latency components under their original timing scopes, providing context for RT-EVT's separate inference and transport parameters. The hardware measurements in Table 1 determine our device-specific evaluation conditions.

Table 12: Latency components reported by robot systems. Entries retain each source's timing scope and statistic. They provide context for RT-EVT's inference and transport parameters; the measurements used for device-specific comparisons are given in Table 1.

Term	System	Reported quantity	Value
Compute	RTC (Black et al., 2025a)	π_0 KV-cache prefill vs. control period	46 ms vs. 20 ms
Compute	SmolVLA (Shukor et al., 2025)	control cycle of the asynchronous stack	33 ms
Compute	Qwen-RobotNav-4B (Team, 2026)	average end-to-end latency, Jetson Thor, FP8	204 ms
Compute	LightNav-0 (Light Origins Team, 2026)	server-side per-step median (P50), Jetson Thor, BF16	268 ms
Transport	RTC (Black et al., 2025a)	remote inference over a LAN	10–20 ms
Transport	π_0 (Black et al., 2026)	off-board Wi-Fi network latency (laboratory)	13 ms
Transport	Rupam Mahmood et al. (2018)	wireless packet inter-arrival on a robot link	0.2–127 ms
Middleware	Kronauer et al. (2021)	ROS 2 transport overhead over raw DDS	up to 50%

D SUPPLEMENTARY RESULTS

D.1 COMPLETE RT-EVT LATENCY SWEEPS

Tables 13–15 report complete SR/TR/CR results for STT, DT and AT at every 10 ms interval from 0 to 270 ms. Each model was evaluated on 1,405 episodes per task at all 28 delays using the hold policy and deterministic settings. Values retain four decimal places from the source records.

Table 13: TrackRT results across 0–270 ms latency. SR/TR/CR (%) for STT, DT and AT. Each completed condition uses the hold policy over 1,405 episodes.

Delay (ms)	**STT**			**DT**			**AT**		
	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓
0	84.9822	85.0914	3.2028	55.8719	67.4114	12.8826	60.4982	73.5795	8.6121
10	84.0569	84.3792	3.2740	54.9466	66.1155	12.2420	60.4982	73.3212	7.9715
20	83.9146	84.2232	3.2740	55.3025	66.0360	12.5267	61.7082	73.7848	7.9715
30	83.2740	83.6359	3.1317	54.7331	65.9586	12.5979	60.7117	73.5498	8.1139
40	82.1352	82.9491	3.3452	54.8043	66.0693	12.3132	59.9288	73.0795	8.1851
50	82.5623	83.1614	2.8470	54.3060	65.2975	13.0961	59.7153	72.2922	8.6121
60	82.0641	83.1081	3.1317	54.0925	65.0097	12.5267	59.7865	72.7445	8.6121
70	82.1352	83.1285	3.4164	53.5231	64.6566	13.1673	58.7900	72.1376	9.2527
80	81.4947	82.6225	3.2740	52.8826	64.2275	13.2384	59.7153	72.0904	8.7544
90	81.5658	82.5269	3.4875	53.7367	64.5688	13.0961	59.2883	72.0510	9.1103
100	81.3523	82.7010	3.2740	52.8114	64.0449	13.5943	60.1423	72.1518	9.4662
110	72.1708	77.9405	6.9039	45.6228	58.1647	18.4342	53.2384	66.9500	15.3025
120	70.0356	76.4108	7.2598	43.2740	56.3786	19.4306	51.3167	65.7223	15.9431
130	67.8292	75.3956	8.2562	42.9181	56.4602	20.4270	51.6014	65.5542	16.1566
140	66.3345	73.4458	8.7544	40.9964	54.7678	20.7117	49.8221	64.7350	17.8648
150	65.3381	73.7045	9.3950	40.6406	54.7194	22.0641	47.2598	63.0915	18.7189
160	63.7722	73.2036	9.4662	37.9359	52.4485	23.2740	47.4733	63.0634	20.1423
170	62.8470	71.5322	10.2491	38.0071	52.1013	24.1281	46.6192	61.6636	20.3559
180	60.2847	69.9126	11.6726	35.6584	50.4774	24.6975	45.8363	61.3453	21.2811
190	57.5089	68.4033	12.8114	35.2313	49.1799	25.6940	42.5623	59.1009	22.9181
200	55.0178	66.9346	13.3808	34.0214	49.0458	26.4769	41.0676	58.2874	23.7722
210	37.7936	55.1790	20.4270	17.7224	36.2220	33.3808	26.1922	46.4276	30.6762
220	35.7295	53.4294	20.7117	16.3701	35.0312	35.3025	24.2705	44.4508	31.7438
230	31.9573	51.9484	22.2776	14.8043	34.1281	34.1637	22.1352	43.6113	34.0214
240	30.3915	50.3255	22.9893	13.7367	32.0472	36.8683	20.2135	41.8971	33.3096
250	28.3986	48.6601	23.0605	12.6690	31.1432	37.4377	18.7189	40.2526	35.3737
260	25.1957	47.2554	24.8399	10.6050	29.3489	38.8612	16.3701	38.1777	35.3737
270	22.1352	43.7186	26.1922	9.8932	28.3550	38.5053	14.0214	37.2719	37.9359

Table 14: OmTrackVLA results across 0–270 ms latency. SR/TR/CR (%) for STT, DT and AT. Each completed condition uses the hold policy over 1,405 episodes.

Delay (ms)	**STT**			**DT**			**AT**		
	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓
0	64.8399	71.7529	5.0534	33.8078	52.5014	7.9715	39.2171	60.1871	5.0534
10	64.6975	71.5305	5.1957	34.0925	52.5328	7.6157	38.1495	60.1845	4.9110
20	64.3416	71.3046	4.7687	34.1637	52.1787	7.6157	39.0747	60.1657	5.1246
30	64.7687	72.0161	4.7687	34.0925	51.9770	7.2598	39.5730	59.8082	5.0534
40	63.9146	71.0532	4.6263	33.6655	52.1371	7.0463	39.0747	59.6941	4.7687
50	63.8434	70.8367	4.9822	33.5943	52.1069	7.3310	39.3594	60.3939	4.9822
60	62.2064	69.7648	4.6263	34.3060	52.5909	7.7580	39.6441	60.4989	5.3381
70	61.7794	69.0032	5.2669	33.5943	51.4932	8.0427	38.7900	59.9487	5.4093
80	59.7153	67.6800	5.4804	32.7402	51.1866	8.8968	39.0036	59.7550	5.1957
90	56.3701	66.8387	5.3381	32.5979	51.5588	9.1815	36.7972	58.3974	5.8363
100	54.0925	65.9553	6.0498	30.0356	50.2997	10.4626	36.9395	57.9158	7.2598
110	38.7189	52.6367	15.8719	21.2811	39.6840	19.5730	29.1815	49.5916	16.0142
120	36.4413	50.7007	18.1495	20.6406	38.5934	20.4270	26.0498	47.7454	17.3665
130	29.8932	47.3012	20.2847	17.6512	36.7809	22.0641	25.3381	46.1804	19.2883
140	26.7616	44.2018	22.1352	17.2242	36.6880	23.3452	19.7865	42.3786	21.7082
150	21.9929	41.2642	24.4128	13.8078	32.3873	26.4769	18.8612	40.9553	23.8434
160	20.2135	37.6932	26.0498	13.9502	32.2181	28.8968	16.5836	37.9667	27.7580
170	18.5053	34.8518	29.6797	11.1032	28.4547	32.5267	14.0214	34.2302	31.3879
180	15.8719	32.6986	35.1601	9.4662	26.0695	37.0819	11.1744	31.5674	35.7295
190	14.0214	30.4178	39.2171	8.6833	24.2845	40.7829	9.8221	27.7448	40.6406
200	9.0391	27.5619	46.1922	5.5516	20.9747	45.6228	6.6192	26.0742	47.4733
210	4.9110	19.9872	53.0249	3.5587	15.9992	49.0391	3.2740	18.6734	53.3096
220	3.9146	19.3312	52.7402	2.2064	14.9690	51.0320	3.9858	18.9582	51.8861
230	3.2740	19.1516	52.9537	1.8505	15.2867	51.3879	2.7046	18.0906	53.7367
240	3.7011	18.6778	53.1673	2.3488	14.3057	51.3167	2.5623	17.3197	54.1637
250	2.9893	17.6138	55.3737	1.6370	13.8631	53.6655	2.6335	17.1232	54.9466
260	2.1352	17.0870	56.0854	1.4235	13.3638	54.9466	2.2064	16.1793	57.5801
270	1.4235	15.2227	59.8577	1.0676	12.6633	58.0071	1.9929	15.2351	60.0712

Table 15: LightNav-0 results across 0–270 ms latency. SR/TR/CR (%) for STT, DT and AT. Each completed condition uses the hold policy over 1,405 episodes.

Delay (ms)	STT			DT			AT		
	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓
0	89.7509	83.2360	2.7758	78.9324	76.1939	6.8327	65.2669	73.6731	7.7580
10	89.2527	82.5417	3.1317	78.5053	75.6469	6.7616	64.2705	72.9937	8.1851
20	88.7544	82.5105	3.1317	77.6512	75.4000	6.9039	66.0498	73.3086	7.3310
30	88.1139	81.9001	3.4875	77.9359	74.8902	6.8327	65.8363	72.6545	7.5445
40	87.6157	81.3993	3.4164	77.1530	74.8372	7.0463	64.4128	72.5281	8.7544
50	87.0463	80.3356	3.8434	76.8683	74.5204	7.0463	64.9110	71.7973	8.6833
60	86.1922	80.4887	3.7722	76.2278	73.7728	8.0427	64.0569	71.4788	8.2562
70	85.8363	79.6130	4.3416	75.3737	73.1776	7.6868	62.9181	70.2659	9.8221
80	85.7651	78.7560	4.7687	75.0890	72.7198	8.6833	60.5694	68.7200	9.8221
90	84.3416	76.9320	4.6975	72.2420	69.6762	9.5374	58.6477	67.3575	10.3203
100	81.2811	73.5028	4.9822	70.6050	68.6771	9.3950	56.0854	63.4893	11.0320
110	48.0427	53.1144	15.4448	41.9929	48.4481	18.5765	28.2562	43.9584	22.7046
120	49.1815	50.5466	13.4520	37.7936	45.3491	20.0712	25.2669	40.7699	22.4911
130	44.4128	47.3292	13.5943	34.5196	41.8946	19.0747	24.1281	37.0595	24.9822
140	43.2028	44.6367	13.1673	33.2384	40.0137	18.7900	20.0000	35.2100	24.2705
150	40.7117	41.6764	13.5231	27.9004	37.3179	19.2171	19.0036	32.3640	23.3452
160	36.5836	38.1285	13.2384	27.6157	35.8617	18.4342	16.0142	29.4261	24.8399
170	31.5302	34.5218	14.2349	25.8363	33.7228	20.3559	13.6655	28.1868	24.1281
180	30.6762	31.7546	15.0890	21.2100	29.1095	21.2811	13.3808	26.1625	25.1246
190	27.1886	30.0525	15.3025	19.9288	28.9049	19.7153	11.6014	23.8141	25.4804
200	22.9181	26.5995	16.9395	17.1530	24.9931	21.9929	9.7509	20.7233	28.0427
210	12.3132	19.0069	20.6406	8.5409	19.6943	24.0569	6.9751	17.2935	29.1815
220	10.2491	18.7038	21.8505	7.9715	18.1203	24.1281	4.9822	15.8520	29.1103
230	12.3843	18.0536	20.4982	7.0463	16.4827	25.1246	5.3381	15.7737	28.6833
240	9.1103	16.9857	20.0000	6.4057	16.3696	24.5552	5.5516	15.4460	28.6121
250	9.0391	16.1053	21.4947	4.8399	15.0980	25.9075	4.5552	14.8175	28.8968
260	7.5445	15.5564	22.2064	5.7651	15.5352	25.9075	3.5587	13.8819	28.8968
270	7.1174	14.8675	21.9217	5.4093	15.3084	24.8399	3.7722	14.2167	28.6833

D.2 COMMON-DELAY AND HARDWARE-DERIVED COMPARISONS

Table 16 summarises zero-latency, common-delay and hardware-derived results under the repaired RT-EVT protocol. Hardware labels specify the source of the injected delay. The raw LightNav-0 Orin P50 is measured at model output, whereas the 1,210 ms RT-EVT condition includes action decoding.

Table 16: RT-EVT results at zero, common and hardware-derived delays. Each task column gives SR/TR/CR (%) over one deterministic evaluation of 1,405 episodes. Hardware conditions use the P50 measurements in Table 1 under the scheduling rule in Section 3.3. The raw LightNav-0 vLLM P50 ends at model output, whereas the 1,210 ms RT-EVT condition includes action decoding.

Model / backend	**Delay (ms)**	**SR↑/TR↑/CR↓ (%)**		
		STT	**DT**	**AT**
Zero-latency reference				
TrackRT	0	85.0/**85.1**/3.2	55.9/67.4/12.9	60.5/73.6/8.6
OmTrackVLA	0	64.8/71.8/5.1	33.8/52.5/8.0	39.2/60.2/**5.1**
LightNav-0	0	**89.8**/83.2/2.8	**78.9/76.2/6.8**	**65.3/73.7**/7.8
Uni-NaVid	0	45.6/55.0/7.2	23.7/43.7/22.9	20.1/44.0/27.0
SimpleNav	0	74.9/82.3/2.7	24.7/47.4/14.7	26.7/54.5/12.5
Common delay				
TrackRT	200	**55.0/66.9/13.4**	**34.0/49.0**/26.5	**41.1/58.3/23.8**
OmTrackVLA	200	9.0/27.6/46.2	5.6/21.0/45.6	6.6/26.1/47.5
LightNav-0	200	22.9/26.6/16.9	17.2/25.0/**22.0**	9.8/20.7/28.0
Uni-NaVid	200	11.1/23.7/33.5	6.5/19.6/43.6	7.1/22.4/44.0
SimpleNav	200	5.2/19.3/59.6	0.9/10.4/65.3	1.4/12.4/67.8
Orin NX MAXN P50				
TrackRT (PhyAI)	150	**65.3/73.7/9.4**	**40.6/54.7/22.1**	**47.3/63.1/18.7**
OmTrackVLA	270	1.4/15.2/59.9	1.1/12.7/58.0	2.0/15.2/60.1
LightNav-0 (vLLM)	1210	0.5/10.7/26.1	0.4/9.9/26.8	0.4/11.4/25.6
RTX 4090 P50				
TrackRT (PhyAI)	40	**82.1/82.9/3.3**	**54.8/66.1**/12.3	**59.9/73.1**/8.2
OmTrackVLA	60	62.2/69.8/4.6	34.3/52.6/**7.8**	39.6/60.5/**5.3**
LightNav-0 (vLLM)	210	12.3/19.0/20.6	8.5/19.7/24.1	7.0/17.3/29.2

D.3 SUPPLEMENTARY BASELINE EVALUATIONS

Uni-NaVid. Table 17 reports its three-task results at all six delays from 0 to 250 ms in 50 ms increments. At zero latency, STT/DT/AT SR was 45.6/23.7/20.1%, below the other evaluated policies. At 200 ms, it fell to 11.1/6.5/7.1%, remaining below LightNav-0 and TrackRT but exceeding OmTrackVLA on all three tasks.

Table 17: Uni-NaVid results at selected RT-EVT delays. SR/TR/CR (%) for STT, DT and AT. Each completed condition uses the hold policy over 1,405 episodes.

Delay (ms)	**STT**			**DT**			**AT**		
	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓
0	45.5516	54.9963	7.1886	23.7011	43.6899	22.9181	20.1423	43.9566	27.0463
50	36.7972	53.8622	6.4057	18.2206	41.0728	22.6335	16.5125	43.4039	26.4057
100	26.3345	49.1557	9.4662	14.3772	38.2706	27.2598	12.3132	39.5461	31.3167
150	15.8007	32.9261	23.4164	8.6833	24.5994	38.7900	9.3238	27.8104	39.5730
200	11.1032	23.7448	33.5231	6.4769	19.5589	43.6299	7.1174	22.3757	43.9858
250	5.3381	16.7100	36.9395	4.1281	14.7479	43.7011	3.0605	16.2844	43.1317

SimpleNav. Table 18 reports all three tasks at six delays from 0 to 250 ms in 50 ms increments. Success rates declined across this range, reaching 5.2/0.9/1.4% on STT/DT/AT at 200 ms.

Table 18: SimpleNav results at selected RT-EVT delays. SR/TR/CR (%) for STT, DT and AT. Each completed condition uses the hold policy over 1,405 episodes.

Delay (ms)	**STT**			**DT**			**AT**		
	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓	SR↑	TR↑	CR↓
0	74.9466	82.3432	2.7046	24.6975	47.3599	14.6619	26.6904	54.5191	12.5267
50	71.2456	81.3503	3.3452	22.6335	46.5895	14.7331	25.6940	51.8819	11.0320
100	56.7972	70.7263	7.7580	17.0819	38.8431	22.1352	19.5730	44.9651	20.4982
150	20.4982	36.8553	38.5765	4.6263	17.9827	52.4555	6.2633	21.8197	53.6655
200	5.1957	19.3410	59.5730	0.9253	10.4287	65.2669	1.4235	12.3529	67.8292
250	1.8505	11.6082	61.9217	0.1423	7.8010	64.1993	0.4982	9.0318	67.3310

D.4 COMPLETE ARCHITECTURE ABLATIONS

Table 19 reports all architecture variants under the protocol in Appendix C.4.

Table 19: Full architecture ablations. Scores are SR↑/TR↑/CR↓ (%) on RT-EVT at zero latency. Each block varies one factor and repeats the released configuration. P50 denotes A100 model-path latency; † marks head-only timing.

Variant	STT	DT	AT	Vis. tokens	P50 (ms)
Visual information source					
Dual encoders (released)	84.98/85.09/3.20	55.87/67.41/12.88	60.50/73.58/8.61	188	84.098
DINOv3 only	72.7/85.2/2.4	42.8/67.8/7.5	51.7/77.6/4.9	188	33.215
SigLIP only	75.9/86.9/2.6	54.1/72.7/9.0	55.2/78.6/5.8	188	77.910
History window					
0 frames	76.7/86.4/3.3	45.2/68.5/10.2	41.8/74.6/9.0	64	—
3 frames	73.1/84.6/2.7	40.9/65.8/7.8	40.1/72.5/4.5	76	—
7 frames	75.9/86.3/2.5	45.3/67.9/6.0	48.1/76.0/4.1	92	—
15 frames	75.8/86.4/2.0	48.0/69.3/6.0	52.8/77.6/4.0	124	—
31 frames (released)	84.98/85.09/3.20	55.87/67.41/12.88	60.50/73.58/8.61	188	84.098
Temporal token allocation					
All fine	69.9/83.9/2.5	40.6/66.2/8.2	47.9/75.2/4.3	2,048	116.893
All coarse	73.5/85.8/2.3	52.9/72.3/7.6	62.1/80.6/4.6	128	83.530
ACTT (released)	84.98/85.09/3.20	55.87/67.41/12.88	60.50/73.58/8.61	188	84.098
History/current token budget (h/c)					
1/64	78.9/87.6/2.4	54.6/71.8/8.0	58.3/79.3/4.7	95	—
4/64 (released)	84.98/85.09/3.20	55.87/67.41/12.88	60.50/73.58/8.61	188	84.098
16/64	76.0/86.4/2.6	55.9/73.0/8.7	62.8/80.6/4.9	560	—
4/16	75.9/86.9/2.8	48.3/70.2/7.9	54.5/78.4/4.7	140	—
4/144	75.3/86.3/2.7	54.0/72.4/8.5	58.9/80.1/6.1	268	—
Temporal and stream markers					
Without time	75.0/85.9/2.2	53.5/71.8/7.5	59.1/79.6/4.9	188	—
Without stream	77.4/87.2/2.3	53.9/72.3/6.4	58.2/79.6/4.7	188	—
Without time and stream	74.7/86.0/2.7	51.5/70.6/7.2	55.0/78.2/4.5	188	—
Full markers (released)	84.98/85.09/3.20	55.87/67.41/12.88	60.50/73.58/8.61	188	84.098
Action head					
Six-layer funnel MLP (released)	84.98/85.09/3.20	55.87/67.41/12.88	60.50/73.58/8.61	188	0.212†
v-prediction DiT (12 steps)	82.1/89.1/4.4	54.4/72.8/12.1	60.8/80.9/6.7	188	20.464†
x-prediction DiT (12 steps)	80.6/88.7/3.9	55.3/73.6/9.6	63.3/81.4/5.4	188	21.636†

Visual token budgets. Among the tested history/current budgets, 4/64 attains the highest STT SR. The 16/64 budget has comparable DT SR and higher AT SR, with 560 visual tokens compared with 188 for 4/64.

Temporal and stream markers. Removing temporal markers, stream markers or both reduces SR on all three tasks.