AR-WAM: Efficient Streaming World-Action Modeling for Autonomous Driving

A Preprint

Baiting Luo **Shatadal Mishra** **Ahmadreza Moradipari** **Nejib Ammar**
Toyota Motor North America R&D - InfoTech Labs
{baiting.luo, shatadal.mishra, ahmadreza.moradipari, nejib.ammar}@toyota.com

Abstract

Autonomous driving requires anticipating how traffic will evolve while responding quickly to new observations. Video foundation models offer the first ability, but repeated history encoding and iterative video generation make them costly to use for driving. We present AR-WAM, an efficient streaming world-action model that combines autoregressive temporal modeling with single-pass future conditioning. Because its video backbone is causal, AR-WAM updates its visual context incrementally, reusing past computation and conditioning future prediction on observed motion. We further hypothesize that future features computed from recent observations can improve generalization in dynamic traffic without iterative video generation. AR-WAM therefore obtains these features in one forward pass and uses them to condition trajectory generation, and reinforcement learning then improves the policy through direct optimization of driving rewards. With a single front camera, AR-WAM achieves 91.5 PDMS after supervised training and 94.0 after reinforcement learning on NAVSIM-v1, outperforming the compared strong end-to-end world-action models. In a dynamic setting of continuous driving, supervised AR-WAM keeps its visual history across decisions and outperforms strong world-action baselines.

Project page: https://baitingluo.github.io/ar-wam/

1 Introduction

End-to-end driving policies increasingly build on pretrained foundation models. Vision-language-action (VLA) models transfer semantic knowledge from image-text pretraining to trajectory prediction [16, 17]. World-action models (WAMs) instead start from video foundation models, whose pretraining on how scenes evolve provides priors over geometry, motion and near-future dynamics [24, 30, 39, 50]. These priors suit driving, because the scene changes continuously and the right action depends on how other road users are moving: a lead vehicle at the same distance may be braking or pulling away, and a single image often cannot tell the two apart.

A driving policy therefore needs recent observations, which reveal changes in relative position and speed. Current-image policies, including OpenVLA and π_0 in robot manipulation [3, 14] and current-frame driving WAMs [50], omit this visual history, although memory-based policies demonstrate its value on temporally dependent tasks [32]. Driving WAMs such as DriveVA and DriveLaW do encode a window of recent frames [24, 39]. With bidirectional temporal attention, however, historical features depend on the other frames in the window, so each new observation requires re-encoding the window [42]. More history then buys motion information at the cost of computation repeated at each decision.

Future prediction adds a second source of latency. WAMs such as DriveWAM and DriveVA iteratively predict future video before or alongside action generation [24, 30], so each decision pays for video denoising even when, as in DriveWAM, the observed history is cached. To avoid this cost, Fast-WAM and SimWAM remove future video tokens at inference while retaining video supervision during training [44, 50]. Yet Simple-WAM shows, in controlled comparisons on robot manipulation, that retaining future conditioning improves generalization, and that a single forward pass over fully noised future tokens recovers most of the benefit [52]. Motivated by these findings, we seek a driving policy

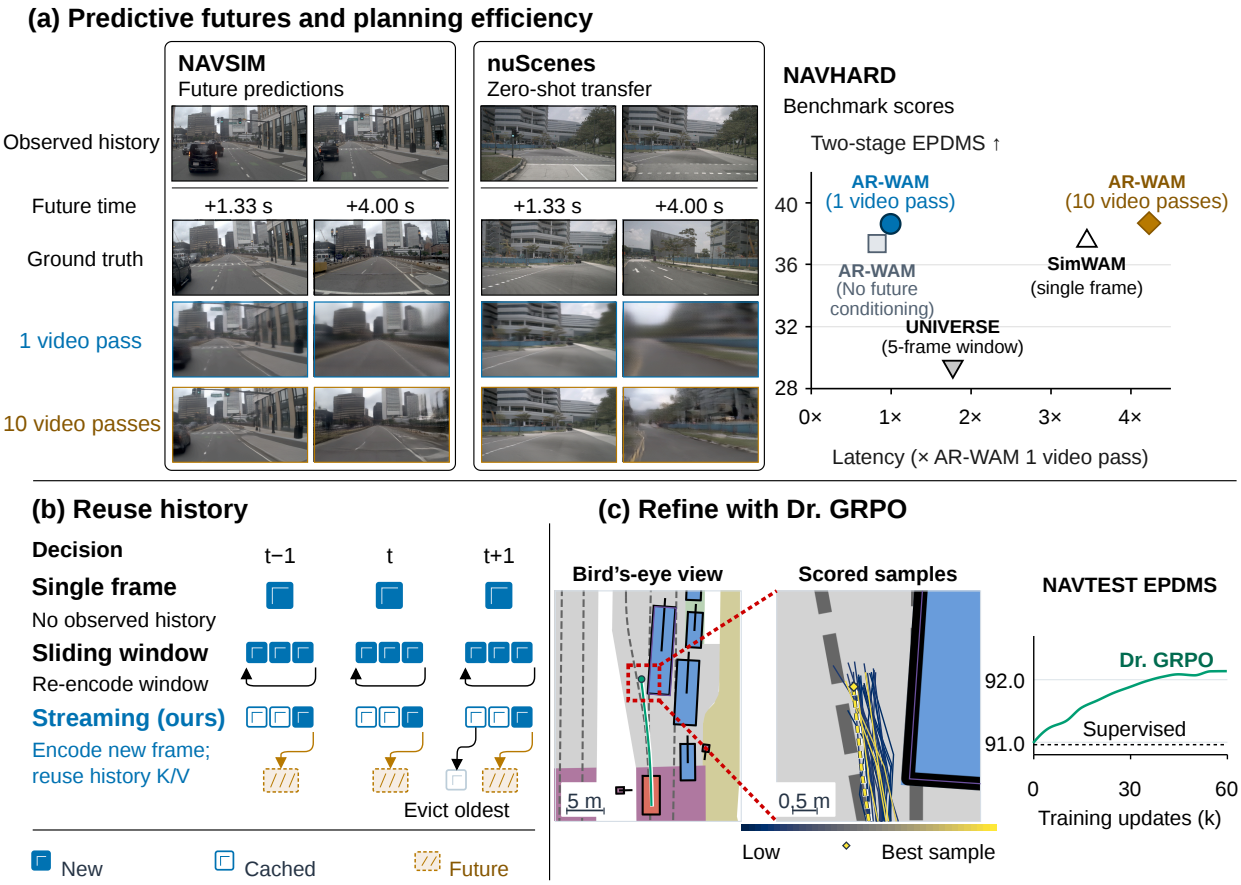


Figure 1: AR-WAM plans from cached visual history and a single predictive pass. (a) Predicted futures and NAVHARD EPDMS against latency. One video pass scores above the variant without future conditioning (38.6 versus 37.3) and matches ten video passes (38.8) at under a quarter of their latency, while SimWAM and UNIVERSE score lower at higher latency. (b) AR-WAM encodes only the new frame and reuses its cached history, whereas a sliding-window policy re-encodes its window at every decision. (c) Reward refinement with Dr. GRPO raises NAVTEST EPDMS to 92.2; the scored samples show 32 plans for one scene, colored by reward.

that uses both temporal history and predictive features without repeating observation encoding or generating future video iteratively.

We present AR-WAM, an efficient streaming world-action model that combines autoregressive temporal modeling with single-pass future conditioning (Fig. 1). Because causal attention makes the keys and values (K/V) of completed observation chunks reusable when new frames arrive, as in streaming video generation [12, 42], AR-WAM updates its visual history incrementally and makes it available to future prediction. Autoregressive pretraining also learns how future states depend on past observations, and recent manipulation results show that it can strengthen the benefit of visual history [11]. We therefore ask how much test-time future prediction contributes once the policy already observes recent motion, and how much of it a driving decision needs.

At each decision, AR-WAM computes only what has changed. It adds the new observation to its cache and looks ahead with one backbone pass over the current image and fully noised future tokens. Its trajectory expert then reads the resulting predictive features and the cached history at matching backbone depths, so no future video is denoised. The ego state and driving command also change at every decision, and conditioning the history on them would require recomputing it each time [11]. AR-WAM therefore applies them only to the predictive pass and the expert, which

keeps the cache valid as driving intent changes. We train video and trajectory prediction jointly, and reinforcement learning then improves the policy through direct optimization of driving rewards [22, 50].

Built on OmniDreams, a pretrained causal video backbone [2], with a single front camera, AR-WAM attains a higher NAVSIM-v1 PDMS than strong WAMs that rely on bidirectional video backbones, such as SimWAM, which plans from a single observation, and UNIVERSE, which re-encodes a window of frames [23]. In a dynamic setting of continuous driving, it also outscores both while running 3.4 and 1.8 times faster, and visual memory accounts for most of its lead over SimWAM. With this history available, one predictive pass matches ten passes of joint future denoising at under a quarter of their latency, and its benefit shows on the perturbed NAVHARD scenes. After refinement with the EPDMS reward, AR-WAM reaches the highest NAVSIM-v2 EPDMS on both NAVTEST and NAVHARD among the compared WAMs that predict a single trajectory without scoring candidates at test time.

Our contributions are:

- We introduce AR-WAM, a causal world-action model that plans from an incrementally updated cache of observed history and a single predictive pass. Two designs keep this cache reusable: the ego state and driving command enter only the per-decision pass, and decision-relative attention lets one trajectory expert read cached and predictive features at any point of a stream.
- We study the role of test-time future prediction once visual history is available, and integrate it into autoregressive world-action modeling as a single backbone pass over noised future tokens.
- We demonstrate strong single-camera planning performance, achieving 91.5 NAVSIM-v1 PDMS after supervised training and 94.0 after reinforcement learning. In the dynamic setting, AR-WAM scores higher than the compared world-action models at lower latency.

2 Related work

Future modeling at test time. World-action models are trained to predict the future, but they differ in how much of it they compute when they act. Generating future video before or alongside the trajectory [24, 30] makes every decision pay for iterative denoising. To avoid this latency, Fast-WAM, SimWAM and UNIVERSE keep video prediction as a training signal and remove the future tokens at inference [23, 44, 50]. The action head then reads no predictive features, however, and Simple-WAM and Faster-WAM show in manipulation that this costs generalization [48, 52]. Other methods keep predictive features while avoiding a full rollout at every decision [1, 8, 39, 46, 47], and Simple-WAM finds that a single pass over pure-noise future tokens recovers most of the loss. Yet it and Adaptive-WAM establish this only for policies that see the current observation, so it remains unclear how such a pass should work beside a cache of history and how much it adds once the policy observes recent motion. Building on the single-pass insight, AR-WAM computes predictive features in one pass that attends to cached observation history, and compares it with a variant without future tokens and with full denoising.

Visual history and memory. The right action depends on motion that a single image does not show. SimWAM and Adaptive-WAM nevertheless plan from the current frame [1, 50], and so cannot observe how other road users are moving. DriveVA, DriveLaW and UNIVERSE observe motion through a window of recent frames [23, 24, 39], but because the window is encoded again whenever a frame arrives, their cost grows with the history they keep. Caching the K/V of observed frames removes this repetition. DriveWAM caches observed video and action chunks [30], yet it generates future video at every decision, and on NAVSIM it makes a single prediction from the current frame, so its memory plays no role there. In manipulation, Long-WAM reuses the K/V of observed frames within one action solve and notes what would invalidate them across decisions, such as an updated robot state or shifted temporal positions [11], whereas MemoryVLA++ retrieves history from a separate memory bank and still denoises future latents with a separate world model [31]. OmniDreams, the causal backbone we build on, reports preliminary results for a policy that runs it once per decision [2], but that policy reads the backbone through a single token and is not described as reusing computation across decisions. Motivated by these gaps, AR-WAM pairs a cache of observed history with the rollout-free pass above, so that neither history encoding nor future denoising is repeated at a decision. Because conditioning visual features on the ego state or driving command, as SimWAM does for the command, would invalidate the cache whenever they change, AR-WAM computes history without these inputs and applies them only in the per-decision pass. Because cached features keep the positions at which they were computed, its trajectory expert measures positions relative to the current decision. It therefore sees the same layout at the start of a stream and after minutes of driving, and no cached feature has to be modified.

Reward refinement. Flow-GRPO makes the sampler of a flow-matching model stochastic, so that sampled outcomes can drive policy-gradient updates [22], and SimWAM applies it to driving with rewards computed in the logged

scene [50]. SimWAM restricts the update to LoRA adapters of its action expert, whereas DSRL keeps the policy frozen and trains an auxiliary policy over its initial noise [35]. Building on these methods, AR-WAM refines the full expert, which is affordable because all sampled plans share one frozen visual pass. Motivated by the analysis of Dr. GRPO [27], it replaces per-group reward standardization with a shared scale when training on mixed scenes.

Together, these gaps motivate the design principle of AR-WAM: the visual computation of a decision never depends on the candidate trajectory, and its history part never depends on the ego state or command. As a result, history is reused across decisions, the predictive pass across action samples, and both across reward-refinement updates, so one model addresses three challenges jointly.

3 Preliminaries

The policy observes a front-camera history $H = (I_{-N_H+1}, \ldots, I_0)$ and a vector q containing ego state, recent ego poses and the driving command. It predicts $A = (a_1, \ldots, a_T)$, where each a_j is a future ego pose (planar position and heading) in the current ego frame. Training also provides the demonstrated trajectory $A^\star$ and logged future video. N_H denotes the number of observed images, L the number of blocks in each of the two networks of Sec. 4, and K the number of action integration steps.

For normalized clean data x and Gaussian noise ϵ, flow matching uses $x^\sigma = (1-\sigma)x + \sigma\epsilon$ and regresses the velocity $\epsilon - x$ [21, 26]. The trajectory expert v_θ, with parameters θ, predicts this velocity and integrates from noise to a clean normalized trajectory:

$$A^{\sigma_{k+1}} = A^{\sigma_k} + (\sigma_{k+1} - \sigma_k) v_\theta(A^{\sigma_k}, \sigma_k; c, q), \tag{1}$$

with $\sigma_1 = 1$ and $\sigma_{K+1} = 0$. Here c is the visual context, the features that the expert reads from the video backbone (Sec. 4.2). The returned physical plan is $A = \mathrm{norm}^{-1}(A^0)$. Action flow time σ and video flow time τ are noise coordinates.

The pure-noise prediction target. Let $Y = y_{1:F}$ be the clean latents of the F future frames and ϵ_v independent Gaussian noise. Video flow matching uses $Y^\tau = (1-\tau)Y + \tau\epsilon_v$ with target velocity $\epsilon_v - Y$. At $\tau = 1$, the population minimizer of the squared-error video objective is

$$\begin{aligned} u^\star &= \mathbb{E}[\epsilon_v - Y \mid \epsilon_v, H, q] \\ &= \epsilon_v - \mathbb{E}[Y \mid H, q], \\ \widehat{Y}^\star &= \epsilon_v - u^\star = \mathbb{E}[Y \mid H, q]. \end{aligned} \tag{2}$$

The implied clean estimate therefore targets a conditional mean in latent space. This result describes the video prediction itself. The trajectory expert reads intermediate features instead, which the action loss also trains, so Sec. 5.2 tests whether these features help planning and whether further denoising adds to them.

4 AR-WAM

AR-WAM pairs a pretrained causal video diffusion transformer (DiT), the backbone with parameters ϕ, with the trajectory expert v_θ of Eq. (1); each has L blocks (Fig. 2). Visual history is computed once and cached as frames arrive (Sec. 4.1), each decision adds one predictive pass (Sec. 4.2), and the expert reads both at every depth (Sec. 4.3).

4.1 Cached visual history

A policy that re-encodes a window of recent frames repeats most of its visual computation at every decision. Causal video generators avoid such repetition: because a frame never depends on later ones under causal attention, the attention keys and values (K/V) of earlier frames can be cached and reused [12, 42]. Building on these designs, AR-WAM keeps its observation history in the causal form of the backbone and bounds it with a sliding window.

Let n index the frames of a stream. When frame n arrives, a causal tokenizer encodes it into one latent, carrying its internal state forward. The backbone processes the history latents y_{h} in fixed-size chunks under chunk-causal attention: a chunk attends to itself and at most W preceding chunks, never to later tokens. The K/V of a completed chunk therefore enter a cache that keeps the newest W chunks, and only the newest, incomplete chunk is recomputed when a frame arrives. Positions are encoded with rotary embeddings [33], under which attention depends on positions only through the offset between query and key; each history latent keeps its stream index as its temporal position. Each decision then costs $C = C_{\mathrm{inputs}} + C_{\mathrm{history}}(W) + C_{\mathrm{world}}(W) + K\,C_{\mathrm{expert}}(W)$, with terms for encoding the new frame, updating at most one history chunk, the predictive pass of Sec. 4.2 and the trajectory expert's K integration steps. For fixed W, memory and per-decision work stay bounded however long the stream runs.

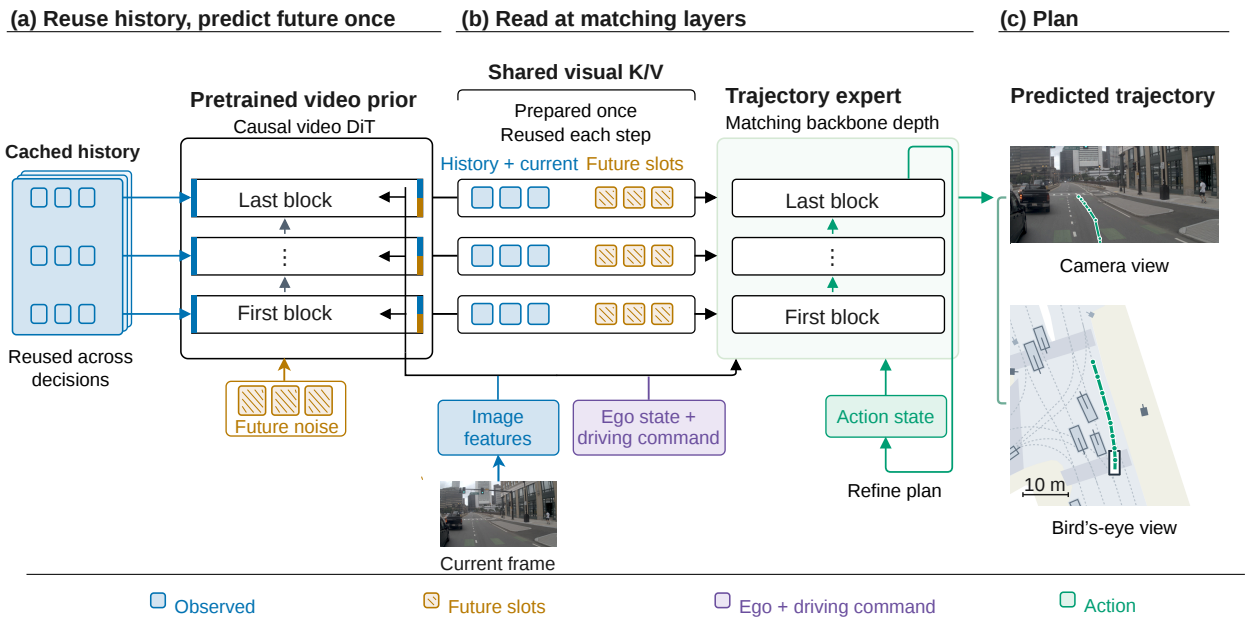


Figure 2: AR-WAM architecture. (a) Each decision runs one pass of the backbone, a causal video DiT, over the current frame and noised future slots, attending to the cached K/V of earlier frames; the ego state and driving command enter this pass but not the cache. (b) At matching backbone depths, the trajectory expert reads the shared visual K/V. (c) The predicted trajectory for a NAVTEST scene.

4.2 One predictive pass per decision

Generating future video by iterative denoising multiplies the backbone cost of each decision. Simple-WAM [52] finds, for manipulation policies conditioned on the current observation, that one forward pass over future tokens left at pure noise retains most of the benefit of test-time future modeling. Without observation history, that study leaves open how such a pass coexists with a cache: what it may attend to, whether it is stored, and how per-decision inputs enter. Building on this insight, AR-WAM introduces a world segment that is rebuilt at each decision.

The segment $X_{\mathrm{w}} = \mathrm{Patch}([y_0; y^{\tau}_{1:F}])$ holds patch tokens of the current latent y_0, encoded separately from the history, and of F future slots: the future latents $y_{1:F}$ noised to video flow time τ, or one fixed Gaussian sample ($\tau = 1$) at deployment. It takes temporal positions $n - F, \ldots, n$. Its tokens attend to all cached history K/V and follow the chunk-causal mask among themselves. The backbone pass Φ_ϕ thus yields the visual context of Eq. (1), $c_\tau = \Phi_\phi(y_{\mathrm{h}}, y_0, y^{\tau}_{1:F}; q)$, the K/V of history and world tokens at every block. At $\tau = 1$, the video flow objective trains the future slots toward the conditional mean of the future given the observations (Eq. (2)), so one pass yields prediction-trained features without generating video.

Because the ego state and driving command q change at every decision whereas observed frames do not, conditioning the history on q would invalidate the cache at each change. AR-WAM therefore injects q only into the world segment and the expert. World tokens receive an embedding of q from a learned ego conditioner through the backbone's conditioning cross-attention, and history tokens always use the unconditional context (Appendix A). No candidate trajectory enters the pass either, so its K/V are shared by all action samples and integration steps.

4.3 Layerwise trajectory expert

Pooling the backbone into a single feature vector would discard the spatial and temporal detail that motion reasoning needs. World-action models that pair a video backbone with a separate action network instead let that network attend to the backbone's features inside each attention layer [44, 50], reading the current frame only. Because AR-WAM must also read a cache whose keys carry ever-growing stream positions, we extend this layerwise attention into a decision-relative form, in which the trajectory expert sees the same relative positions however long the stream has run.

The expert has one block per backbone block, and its block $\ell \in \{1, \ldots, L\}$ attends to backbone block ℓ. From the backbone it receives the keys and values of the history tokens, $\mathbf{K}^{\ell}_{\mathrm{h}}, \mathbf{V}^{\ell}_{\mathrm{h}}$, and of the world tokens, $\mathbf{K}^{\ell}_{\mathrm{w}}, \mathbf{V}^{\ell}_{\mathrm{w}}$, which over

all blocks form c_τ. The expert adds two kinds of tokens of its own. Action tokens embed the noisy trajectory A^{σ_k} and give the queries $\mathbf{Q}_{\mathrm{a}}^{\ell}$ and the keys and values $\mathbf{K}_{\mathrm{a}}^{\ell}, \mathbf{V}_{\mathrm{a}}^{\ell}$. Auxiliary tokens hold higher-resolution features of the current image I_0 from a separate encoder and an embedding of q, and give $\mathbf{K}_{\mathrm{x}}^{\ell}, \mathbf{V}_{\mathrm{x}}^{\ell}$. With $\mathrm{Attn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})$ denoting attention and $[\,\cdot\,;\,\cdot\,]$ concatenation over tokens, each action query attends to all four groups:

$$\begin{aligned}\mathbf{O}^{\ell} = \mathrm{Attn}\big(&\mathcal{R}_{\Delta}\mathbf{Q}_{\mathrm{a}}^{\ell},\, [\mathbf{K}_{\mathrm{h}}^{\ell}; \mathbf{K}_{\mathrm{w}}^{\ell}; \mathcal{R}_{\Delta}\mathbf{K}_{\mathrm{x}}^{\ell}; \mathcal{R}_{\Delta}\mathbf{K}_{\mathrm{a}}^{\ell}],\\ &[\mathbf{V}_{\mathrm{h}}^{\ell}; \mathbf{V}_{\mathrm{w}}^{\ell}; \mathbf{V}_{\mathrm{x}}^{\ell}; \mathbf{V}_{\mathrm{a}}^{\ell}]\big).\end{aligned} \tag{3}$$

Here $\mathcal{R}_{\Delta}$ shifts the temporal position of a query or key by $\Delta = n - F$, the stream position at which the world segment starts. The shift reconciles two ways of counting time: backbone keys carry positions in the stream, which keep growing, whereas the expert places its own tokens as if every world segment started at zero. Shifting the expert's queries and keys by Δ moves them onto the stream's axis. Because rotary attention depends only on the offset between query and key, the expert then sees each backbone key at its distance from the current decision, and attention among its own tokens is unchanged. Only the expert's side is shifted, so no cached key is modified, and the same weights apply at the start of a stream ($\Delta = 0$) and deep into a long one. The expert integrates the K Euler steps of Eq. (1), reusing the decision's visual K/V at every step.

4.4 Joint training

The backbone, its ego conditioner, the current-image encoder and the trajectory expert are trained jointly with two flow-matching losses:

$$\mathcal{L} = \mathcal{L}_{\mathrm{act}} + \lambda_v \mathcal{L}_{\mathrm{video}}, \tag{4}$$

where $\mathcal{L}_{\mathrm{act}}$ regresses the velocity of the normalized demonstrated trajectory $A^{\star}$, $\mathcal{L}_{\mathrm{video}}$ regresses the velocity of the F future latents through a video head used only in training, and λ_v weights the video term. At deployment the future slots are pure noise, so the expert must be trained on such inputs, while the backbone should keep video supervision at every noise level. We therefore adopt the mixed noise schedule of Simple-WAM [52]: all future slots share one video flow time, set to $\tau = 1$ with probability p and drawn from the usual schedule otherwise. Appendix A gives the losses, schedules and weights.

4.5 Reward refinement

Imitation fits the recorded maneuver without comparing the consequences of alternative plans. Policy gradients can make that comparison, but the deterministic integration of a flow policy gives no tractable likelihood for a sampled plan. Flow-GRPO [22] injects noise into the sampling steps of flow-matching generators, which yields exploration and tractable path likelihoods, and SimWAM [50] shows that rewards computed in the logged scene can refine a driving expert in this way. Building on these results, AR-WAM refines its full expert at low cost: because the visual pass does not depend on the trajectory, each scene needs one frozen pass, shared by all sampled plans and policy updates. We inject Gaussian noise into the expert's K integration steps and sample G plans per scene. A rule-based driving score R, such as the PDM score [7], rates each plan in the scene's logged state. Group-relative advantages compare each plan with the others from its scene and drive a clipped policy-gradient update with a velocity anchor to the starting expert. Such advantages are commonly divided by the standard deviation of the rewards in their group. With rule-based driving scores this is a poor fit, because nearly tied groups are the norm. Such scores consist mostly of pass/fail checks, so the plans that a well-trained policy samples in an ordinary scene all pass and differ only slightly in score, and the division would weigh these small differences as heavily as a collision avoided in a difficult scene. Dr. GRPO [27] identifies this bias toward groups whose rewards differ little and removes the division. We do the same and train on a mix of mined hard scenes and ordinary scenes with one shared reward scale. At deployment, the expert integrates deterministically from a zero trajectory. Appendix A.2 gives the exploration kernel, the objective and the scene pools.

5 Experiments

5.1 Evaluation protocol

Training and evaluation. AR-WAM is trained on all 103,288 NAVTRAIN scenes with one front camera. We report NAVSIM-v1 PDMS and its components, NAVSIM-v2 EPDMS and the two-stage NAVHARD score [6, 7] on the 12,146 NAVTEST and 5,912 NAVHARD scenes. Standard NAVTEST queries reset the streaming policy and supply four history frames. The supervised policy plans with ten integration steps and the reward-refined policies with four; Appendix A gives the training details.

Table 1: Future modeling at inference. PDMS: NAVTEST; EPDMS: NAVHARD two-stage; L2: nuScenes zero-shot average in metres (Sec. 5.5); Passes: backbone world passes per decision; Latency: median per decision relative to one pass, which takes 146 ms on an H200. Bold: best result in each column.

Future	PDMS ↑	EPDMS ↑	L2 ↓	Passes	Latency
No future	**91.5**	37.3	**0.87**	1	**0.8**×
One pass (AR-WAM)	**91.5**	38.6	**0.87**	1	1.0×
Clear future	**91.5**	**38.8**	0.88	10	4.2×

Table 2: Comparison on NAVTEST and NAVHARD. Baseline values are publication-reported. Cam.: cameras (L: LiDAR; M: multiview). †: reward post-training. ‡: proposal scoring. Bold: best result in each column, not counting the human driver and ‡ rows.

			NAVTEST, NAVSIM-v1						NAVTEST	NAVHARD
Method	Ref.	Cam.	NC	DAC	TTC	Comf.	EP	PDMS	EPDMS	EPDMS
Reference planners without a world model										
Human driver [7]	–	–	100.0	100.0	100.0	99.9	87.5	94.8	–	–
DiffusionDrive [20]	CVPR'25	3+L	98.2	96.2	94.7	**100.0**	82.2	88.1	–	–
ReCogDrive† [17]	ICLR'26	1	97.9	97.3	94.9	**100.0**	87.3	90.8	–	–
DrivoR‡ [15]	CVPR'26	4	98.9	98.3	96.2	100.0	89.1	93.1	–	48.3
World-action models, direct prediction, supervised										
Epona [47]	ICCV'25	1	97.9	95.1	93.8	99.9	80.4	86.2	–	–
DriveVLA-W0 [16]	ICLR'26	1	–	–	–	–	–	–	86.1	24.4
PWM [49]	NeurIPS'25	1	98.6	95.9	95.4	**100.0**	81.8	88.1	–	–
DriveLaW [39]	CVPR'26	1	99.0	97.1	96.7	**100.0**	81.3	89.1	88.6	30.6
DriveWAM [30]	NeurIPS'26	1	98.3	98.1	95.2	**100.0**	84.3	90.1	–	–
SimWAM, Wan2.2 [50]	arXiv	1	98.6	98.0	95.8	–	84.0	90.3	90.2	37.6
DriveVA [24]	ECCV'26	1	99.0	97.3	98.4	**100.0**	83.2	90.5	–	–
GeoWAM [28]	arXiv	M	–	–	–	–	–	–	90.7	39.6
SUV [45]	arXiv	1	99.1	97.8	96.7	**100.0**	84.6	90.8	91.0	36.9
UNIVERSE [23]	NeurIPS'26	1	99.1	97.6	**98.5**	**100.0**	83.6	91.0	–	–
WA-JEPA [38]	arXiv	4	**99.5**	98.3	97.7	**100.0**	85.0	91.8	91.7	–
World-action models, post-trained or proposal scoring										
SimWAM, Wan2.2†	arXiv	1	98.5	99.1	96.1	99.8	86.5	91.9	–	–
SimWAM, Cosmos2.5†	arXiv	1	98.8	99.2	96.4	–	87.0	92.4	–	–
DA-WAM‡ [51]	arXiv	1	99.1	98.9	96.8	99.8	90.0	93.7	89.2	–
AR-WAM		1	99.4	98.2	98.0	**100.0**	84.5	91.5	90.9	38.6
AR-WAM + RL, PDMS†		1	99.0	**99.3**	96.1	**100.0**	**90.7**	**94.0**	–	–
AR-WAM + RL, EPDMS†		1	99.3	**99.3**	97.1	**100.0**	86.6	92.7	**92.2**	**42.3**

Baselines. Baseline values in Table 2 are published results, each with its own cameras and training data. The dynamic setting of Sec. 5.3 has no published results, so we run four models that handle history and the future differently. SimWAM plans from the current frame with the expert design closest to ours, and UNIVERSE, the strongest peer-reviewed single-camera world-action model in Table 2, re-encodes a window of frames. DriveVA [24] also re-encodes a window but generates future video with the trajectory, and DriveWAM [30] does so from the current frame before planning. Appendix B gives details.

5.2 Does future modeling improve decisions?

Table 1 compares the single predictive pass with two alternatives. In the no-future variant, the trajectory expert reads only the observed history and the current latent, although the backbone is still trained to predict future video, as in current-frame WAMs that drop future tokens at inference [44, 50]. This variant repeats AR-WAM's final training stage with the same initialization, data order, noise draws, losses and schedule, and differs only in what the expert reads. In the clear-future variant, AR-WAM's own weights denoise the future video jointly with the action, following Simple-WAM's joint inference [52], so the last action steps read an almost clean predicted future at the cost of one backbone pass per action step.

Table 3: Dynamic setting: the 47,220 non-waiting frames of 138 NAVTEST logs, run in order at 2 Hz with a persistent history cache. Columns are one-stage NAVSIM-v2 sub-scores. Hist.: camera frames used per decision. Latency: median per decision at batch size one on an H200. Current frame: AR-WAM retrained without visual history. Bold: best result in each column.

Method	Hist.	Latency (ms)	NC	DAC	DDC	TLC	EP	TTC	LK	HC	EC	EPDMS
SimWAM [50]	1	499.1	98.0	99.1	99.4	99.5	88.0	98.2	98.1	**98.7**	87.2	90.4
UNIVERSE [23]	5	256.9	98.5	99.0	**99.5**	**99.8**	87.8	98.7	97.9	98.6	64.5	88.3
DriveVA [24]	5	258.8	98.5	99.0	**99.5**	**99.8**	87.7	98.6	98.2	98.5	57.4	87.4
DriveWAM [30]	1	895.7	98.0	99.1	99.4	99.4	88.3	98.1	98.3	**98.7**	**91.8**	90.9
AR-WAM, current frame	1	**106.9**	98.2	99.1	**99.5**	99.7	88.0	98.3	98.5	**98.7**	84.8	90.7
AR-WAM	5	146.3	**98.7**	99.1	**99.5**	99.6	88.0	98.8	**98.8**	**98.7**	88.0	91.5
AR-WAM + RL, EPDMS	5	132.8	**98.7**	**99.6**	**99.5**	99.5	**88.6**	**98.9**	98.0	**98.7**	89.9	**92.2**

Test-time future prediction improves decisions at a small cost in latency. One predictive pass raises EPDMS on the perturbed two-stage scenes of NAVHARD by 1.3 over the no-future variant, with unchanged NAVTEST and nuScenes accuracy, while latency grows only from $0.8\times$ to $1.0\times$. Denoising a clear future adds a further 0.2 EPDMS but takes more than four times as long, so one pass recovers most of its benefit at under a quarter of the latency.

5.3 Evaluation in a dynamic setting

Each standard NAVTEST query rebuilds the policy's history from four frames, so it cannot measure the effect of history carried across decisions. We therefore also evaluate a dynamic setting in which each policy runs through 138 full NAVTEST source logs in chronological order (69,711 frames at 2 Hz) and keeps its history across frames. Observations come from the logs and do not depend on the predicted actions. Plans are scored with one-stage EPDMS, including extended comfort (EC) between consecutive plans.

Evaluated frames. NAVSIM builds NAVTRAIN and NAVTEST without scenes it considers trivial, such as standing still or driving straight at near-constant speed [7], but full logs keep them. In 32% of the frames in our logs, the logged vehicle moves less than 1 m over the next 4 s, compared with 1.9% of NAVTEST frames. Policies trained on NAVTRAIN therefore rarely see sustained waiting, while UNIVERSE and DriveVA train on full logs. To compare policies on the benchmark's scene distribution, we exclude these waiting frames. The criterion uses only the logged vehicle, so all policies are scored on the same frames.

On the 47,220 non-waiting frames valid for all compared policies (Table 3), AR-WAM outscores all four baselines and runs faster. Its lead over SimWAM, which replans from the current frame, comes mainly from fewer at-fault collisions and time-to-collision violations, and visual memory accounts for most of it: the current-frame variant, which repeats AR-WAM's final training stage without visual history, falls between the two, closer to SimWAM. This memory costs about 40 ms per decision. The window models UNIVERSE and DriveVA have much lower extended comfort, meaning less consistent consecutive plans. DriveVA, which generates future video at every decision, scores below UNIVERSE. DriveWAM has the highest extended comfort but trails AR-WAM by 0.6 EPDMS at six times AR-WAM's latency. Refinement with the EPDMS reward raises both EPDMS and extended comfort further.

5.4 Benchmark comparison

Table 2 compares AR-WAM with world-action models and reference planners, and Table 5 in the appendix lists all compared methods. Direct methods output one trajectory per decision; proposal-scoring methods select one of up to 64 candidates with a scorer trained on PDM sub-scores. Among the compared supervised direct methods, AR-WAM has the highest NAVSIM-v1 PDMS with a single camera, and only the four-camera WA-JEPA scores higher. Its NAVSIM-v2 EPDMS is within 0.1 of the best supervised single-camera result, and on NAVHARD only GeoWAM, pretrained on 832 additional hours of driving video, scores higher among direct methods. The supervised policy is cautious: its no-collision rate is close to the best, and ego progress is its weakest term. Reward refinement shifts this balance toward progress. With the PDMS reward, AR-WAM reaches 94.0 PDMS, above every direct method and above the proposal-scoring DA-WAM and DrivoR. With the EPDMS reward, it has the best NAVSIM-v2 scores among direct methods on both NAVTEST and NAVHARD, although DrivoR, a planner without a world model that scores its own proposals from four cameras, stays ahead on NAVHARD.

Table 4: NAVSIM-to-nuScenes transfer. Lower is better. L2 is in metres; Col. is the horizon-averaged collision percentage. The Adaptive-WAM row is publication-reported; the others are our evaluations on 4,519 samples (Appendix B). Of the 4,437 samples with all horizons, 635 are waiting: the logged vehicle moves less than 1 m within 3 s. Bold: best result in each column.

	All samples					Non-waiting		Waiting	
	L2				Col.	L2	Col.	L2	Col.
Method	1 s	2 s	3 s	Avg.	Avg.	Avg.	Avg.	Avg.	Avg.
Constant velocity	0.33	1.13	2.34	1.27	0.47	1.46	0.54	**0.15**	0.06
SimWAM [50]	0.29	0.83	1.77	0.97	0.05	0.94	0.05	1.12	0.05
UNIVERSE [23]	0.30	0.75	**1.44**	**0.83**	0.07	0.91	0.08	0.37	**0.01**
DriveVA [24]	0.34	0.79	1.48	0.87	0.10	0.97	0.08	0.30	0.20
DriveWAM [30]	0.27	0.77	1.68	0.91	**0.03**	**0.88**	**0.03**	1.08	**0.01**
Adaptive-WAM [1]	0.35	**0.71**	1.58	0.88	0.08	–	–	–	–
AR-WAM	**0.24**	0.73	1.63	0.87	0.05	**0.88**	0.04	0.82	0.14

5.5 Zero-shot transfer

To test transfer beyond NAVSIM, we apply AR-WAM to nuScenes [4] without fine-tuning, following DriveVA's evaluation protocol [24] on 4,519 samples from 150 validation scenes. SimWAM, UNIVERSE, DriveVA and DriveWAM are evaluated on the same samples, together with a constant-velocity reference, since ego state alone can explain much of open-loop performance [18]. Table 4 also splits the samples with the waiting criterion of Sec. 5.3. NAVSIM's training protocol removes most scenes in which the vehicle stands still, so the non-waiting samples correspond to the kind of scenes AR-WAM is trained on. On these samples, AR-WAM and DriveWAM transfer best, with the lowest L2 error and about half the collision rate of UNIVERSE and DriveVA. Over all samples, including the 14% of waiting samples that its training largely excludes, AR-WAM stays close to UNIVERSE (0.87 versus 0.83 m average L2), matches DriveVA and has the lowest one-second error. DriveWAM has the lowest collision rate but higher L2 error than AR-WAM at every horizon. On the waiting samples, UNIVERSE and DriveVA, which train on full logs, stay near the log, while AR-WAM outperforms SimWAM and DriveWAM, the other NAVSIM-trained policies.

5.6 Reward refinement

Reward refinement on a mix of mined hard scenes and ordinary scenes raises NAVTEST PDMS from 91.6 to 94.0 at four integration steps (Table 2). The gain comes from ego progress and drivable-area compliance, while time-to-collision compliance drops. Refining with the NAVSIM-v2 EPDMS reward instead raises EPDMS from 90.9 to 92.2 on NAVTEST and from 38.6 to 42.3 on NAVHARD.

5.7 Runtime and history reuse

UNIVERSE re-encodes its five-frame window at every decision, whereas AR-WAM encodes only the newest frame and reuses its cache. With the same five frames, AR-WAM is 1.8 times faster than UNIVERSE and DriveVA in median latency (Table 3), although it also computes predictive features. It is 3.4 times faster than SimWAM, which plans from the current frame with ten flow steps, and 6.1 times faster than DriveWAM. Four action steps instead of ten lower AR-WAM's latency further at unchanged NAVTEST PDMS. Table 6 in the appendix adds tail latency and peak memory.

6 Conclusion

AR-WAM pairs a cache of observed history with one predictive pass and reads both through decision-relative attention. Because its visual computation depends on neither the candidate trajectory nor, for history, the ego state or command, it is reused across decisions, action samples and reward-refinement updates. With history available, one predictive pass matches iterative future denoising at under a quarter of its latency. AR-WAM reaches 94.0 NAVSIM-v1 PDMS after reward refinement and leads other strong world action baselines in the dynamic setting at lower latency.

References

[1] Sining Ang, Yuguang Yang, and Yan Wang. Adaptive-WAM: Quality-guided early-exit planning from intermediate video-diffusion features. *CoRR*, abs/2608.06008, 2026.

[2] Aarti Basant, Amlan Kar, Despoina Paschalidou, Fangyin Wei, Francesco Ferroni, Guillermo Garcia Cobo, Haithem Turki, Huan Ling, Jaewoo Seo, James Lucas, et al. NVIDIA OmniDreams: Real-time generative world model for closed-loop autonomous vehicle simulation. *arXiv preprint arXiv:2606.03159*, 2026.

[3] 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. *CoRR*, abs/2410.24164, 2024.

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[5] Tianhui Cai, Xinglong Sun, Chao Fang, Zhenxin Li, Rui Song, Jose M. Alvarez, Yunxiang Mao, Jiaqi Ma, and Langechuan Liu. AffordDrive3D: Affordance-aware world-action modeling with spatial understanding. *arXiv preprint arXiv:2610.11060*, 2026.

[6] Wei Cao, Marcel Hallgarten, Tianyu Li, Daniel Dauner, Xunjiang Gu, Caojun Wang, Yakov Miron, Marco Aiello, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. Pseudo-simulation for autonomous driving. In *9th Annual Conference on Robot Learning*, 2025.

[7] Daniel Dauner, Marcel Hallgarten, Tianyu Li, Xinshuo Weng, Zhiyu Huang, Zetong Yang, Hongyang Li, Igor Gilitschenski, Boris Ivanovic, Marco Pavone, Andreas Geiger, and Kashyap Chitta. NAVSIM: Data-driven non-reactive autonomous vehicle simulation and benchmarking. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

[8] Zhaoxin Fan, Tianbao Zhang, Wenjun Wu, Xiaofeng Wang, Yeying Jin, Jian Zhao, Zheng Zhu, and Shuicheng Yan. Drive-HWM: Hierarchical world models for dynamic-latent guided autonomous driving. *arXiv preprint arXiv:2609.03572*, 2026.

[9] Jiacheng Fu, Yibo Yuan, Meng Tian, Yue Li, Jiangtong Zhu, Jianhua Han, Yueyi Zhang, Jianwu Fang, Jianru Xue, Hang Xu, and Zhiwei Xiong. 4D-WAM: 4D consistent world modeling for autonomous driving. *arXiv preprint arXiv:2608.10107*, 2026.

[10] Yingdong Gu, Haoyu Wang, Shaocheng Yan, Zhenjun Zhao, Hao Han, Pengcheng Shi, Xianfei Li, Qinghua Tan, Wenlong Liao, Tao He, Jiayuan Li, and Pai Peng. MORE: Learning motion-oriented registers for driving world models and planning. *arXiv preprint arXiv:2610.14871*, 2026.

[11] Wei Huang, Bohan Zhang, Chenzhi Liu, Isabella Liu, Shuai Yang, Weian Mao, Luozhou Wang, Yicheng Xiao, Weifeng Lin, Qixin Hu, Bryan Chu, Sifei Liu, Linxi Fan, Xiaojuan Qi, Song Han, and Yukang Chen. Long-WAM: Scaling the context of world-action models. *arXiv preprint arXiv:2610.10528*, 2026.

[12] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

[13] Changhui Jing, Haochen Tian, Hongchen Li, Tianyu Li, Caojun Wang, Zihao Zhang, Yanhao Liu, Chengen Xie, Bailin Li, Hao Zhao, and Zhenjie Yang. SimVTA: Simulation-grounded video-trajectory alignment for driving world action models. *arXiv preprint arXiv:2610.13781*, 2026.

[14] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Paul Foster, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning, 6-9 November 2024, Munich, Germany*, pages 2679–2713. PMLR, 2024.

[15] Ellington Kirby, Alexandre Boulch, Yihong Xu, Yuan Yin, Gilles Puy, Éloi Zablocki, Andrei Bursuc, Spyros Gidaris, Renaud Marlet, Florent Bartoccioni, Anh-Quan Cao, Nermin Samet, Tuan-Hung Vu, and Matthieu Cord. Driving on registers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pages 32058–32069. Computer Vision Foundation, 2026.

[16] Yingyan Li, Shuyao Shang, Weisong Liu, Bing Zhan, Haochen Wang, Yuqi Wang, Yuntao Chen, Xiaoman Wang, Yasong An, Chufeng Tang, et al. DriveVLA-W0: World models amplify data scaling law in autonomous driving. In *International Conference on Learning Representations*, 2026.

[17] Yongkang Li, Kaixin Xiong, Xiangyu Guo, Fang Li, Sixu Yan, Gangwei Xu, Lijun Zhou, Long Chen, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. ReCogDrive: A reinforced cognitive framework for end-to-end autonomous driving. In *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026.

[18] Zhiqi Li, Zhiding Yu, Shiyi Lan, Jiahan Li, Jan Kautz, Tong Lu, and Jose M. Alvarez. Is ego status all you need for open-loop end-to-end autonomous driving? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

[19] Ao Liang, Youquan Liu, Mingxiao Li, Wei Chow, Lingdong Kong, Yue Liao, Dongyue Lu, Yu Yang, Wei Yin, and Runnan Chen. VITAL: From visual motion to precise action. *arXiv preprint arXiv:2610.20604*, 2026.

[20] Bencheng Liao, Shaoyu Chen, Haoran Yin, Bo Jiang, Cheng Wang, Sixu Yan, Xinbang Zhang, Xiangyu Li, Ying Zhang, Qian Zhang, and Xinggang Wang. DiffusionDrive: Truncated diffusion model for end-to-end autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 12037–12047. Computer Vision Foundation / IEEE, 2025.

[21] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[22] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

[23] Mengmeng Liu, Diankun Zhang, Jiuming Liu, Jianfeng Cui, Hongwei Xie, Guang Chen, Hangjun Ye, Francesco Nex, Hao Cheng, and Michael Ying Yang. UNIVERSE: Unified video action models for autonomous driving with flexible mask-modulated modality generation. In *Advances in Neural Information Processing Systems*, 2026.

[24] Mengmeng Liu, Diankun Zhang, Jiuming Liu, Jianfeng Cui, Hongwei Xie, Guang Chen, Hangjun Ye, Michael Ying Yang, Francesco Nex, and Hao Cheng. DriveVA: Video action models are zero-shot drivers. In *European Conference on Computer Vision*, 2026.

[25] Shuai Liu, Hechangle Gong, Hao Jiang, Runlin He, Junxiang Zhan, Kai Huang, Sheng Yang, and Shaoqing Ren. MM-Future: Multi-mode joint world-action modeling for autonomous driving. *arXiv preprint arXiv:2609.20377*, 2026.

[26] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023.

[27] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. *arXiv preprint arXiv:2503.20783*, 2025.

[28] Yiren Lu, Xin Ye, Jiaming Liu, Philip Jacobson, Jin Yao, Yi chung Chen, Liam Merino, Dhruva Dixith Kurra, Min Cai, Tom Lampo, Yu Yin, Danhua Guo, and Burhan Yaman. GeoWAM: Visual geometry world action models for autonomous driving. *arXiv preprint arXiv:2608.23486*, 2026.

[29] Dhruv Parikh, Fengcheng Yu, Quankai Gao, Jiawei Yang, Junjie Ye, Maulik Bhatt, Thang Vu, Charles Ochoa, Rowan McAllister, Igor Vasiljevic, Rajgopal Kannan, Viktor Prasanna, Vitor Guizilini, and Yue Wang. PhysWAM: Physically consistent world action model for autonomous driving. *CoRR*, abs/2609.37970, 2026.

[30] Chen Shi, Jinrui Xu, Shaoshuai Shi, Kehua Sheng, Bo Zhang, and Li Jiang. DriveWAM: Video generative priors enable scalable world-action modeling for autonomous driving. *arXiv preprint arXiv:2605.28544*, 2026.

[31] Hao Shi, Weiye Li, Bin Xie, Yulin Wang, Renping Zhou, Tiancai Wang, Xiangyu Zhang, Ping Luo, and Gao Huang. MemoryVLA++: Temporal modeling via memory and imagination in vision-language-action models. *arXiv preprint arXiv:2606.09827*, 2026.

[32] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. MemoryVLA: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *The Fourteenth International Conference on Learning Representations, ICLR 2026, Rio de Janeiro, Brazil, April 23-27, 2026*. proceedings.iclr.cc, 2026.

[33] Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[34] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[35] Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of the 9th Conference on Robot Learning*, pages 258–282, 2025.

[36] Jinyang Wang, Shiwei Li, Junjian Wang, Zhiqiang Deng, Jianbin Gao, Yihang Zhao, Liu Liu, Yongjia Zhao, Jinlong Chen, Huirui Xu, Yifeng Pan, Kangwei Liu, Fan Ren, Ji Tao, and Minghao Yang. SV-WAM: An efficient surround-view world-action model for end-to-end autonomous driving. *arXiv preprint arXiv:2609.03602*, 2026.

[37] Shuzheng Wang, Haoxuan Li, Shengzhen Li, Chenxi Wu, Yajing Sun, Lu Hou, Wenyu Liu, and Xinggang Wang. DriveWLA: Unifying future dynamics and scene semantics through world-language-action modeling for autonomous driving. *arXiv preprint arXiv:2610.17903*, 2026.

[38] Xinlin Wang, Yujiao Xiang, Yuheng Zhou, Jingqi Wang, Minqing Huang, Jiajie Huang, Dongxu Wei, Tingguang Zhou, Xiyang Wang, Gong Chen, Zhi Xu, Feiyang Tan, Hangning Zhou, and Mu Yang. WA-JEPA: Rethinking the video JEPA paradigm for world-action modeling in autonomous driving. *CoRR*, abs/2608.20974, 2026.

[39] Tianze Xia, Yongkang Li, Lijun Zhou, Jingfeng Yao, Kaixin Xiong, Haiyang Sun, Bing Wang, Kun Ma, Guang Chen, Hangjun Ye, Wenyu Liu, and Xinggang Wang. DriveLaW: Unifying planning and video generation in a latent driving world. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2026, Denver, CO, USA, June 3-7, 2026*, pages 39701–39712. Computer Vision Foundation, 2026.

[40] Jingbo Xu, Hanchao Leng, JiaHui Huang, Hanbing Li, Jingwei Zhao, Kuiyuan Yang, Chen Gao, Guang Chen, Hangjun Ye, and Yong Li. DriveUniTok: A unified multimodal tokenizer for driving world-action models. *arXiv preprint arXiv:2610.35044*, 2026.

[41] Jinchang Xu, Hongda Yu, Fengwei Dong, Wenhui Huang, Xi Wei, Yongzhi Liu, Sunan Zhang, Jirao Wang, Chen Lv, Bingbing Li, Guodong Yin, and Weichao Zhuang. PlanWAM: Planning-shaped future representations for end-to-end autonomous driving. *arXiv preprint arXiv:2610.11382*, 2026.

[42] Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Frédo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pages 22963–22974. Computer Vision Foundation / IEEE, 2025.

[43] Fengcheng Yu, Dhruv Parikh, Junjie Ye, Maulik Bhatt, Thang Vu, Igor Vasiljevic, Vitor Guizilini, and Yue Wang. DriveReferee: Geometric safety verdicts need not be learned for driving world-action models. *CoRR*, abs/2609.22762, 2026.

[44] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? *CoRR*, abs/2603.16666, 2026.

[45] Yibo Yuan, Jiacheng Fu, Jiangtong Zhu, Yi Li, Jianhua Han, Meng Tian, Zhuohan Liu, Zhiwei Xiong, Hang Xu, Jianwu Fang, and Jianru Xue. SUV: Future scene understanding as video generation for end-to-end driving. *arXiv preprint arXiv:2608.03084*, 2026.

[46] Chushan Zhang, Jinguang Tong, Xuesong Li, Yikai Wang, and Hongdong Li. Keep the future, drop the rollout: RIFT for world action models. *CoRR*, abs/2608.11521, 2026.

[47] Kaiwen Zhang, Zhenyu Tang, Xiaotao Hu, Xingang Pan, Xiaoyang Guo, Yuan Liu, Jingwei Huang, Li Yuan, Qian Zhang, Xiao-Xiao Long, et al. Epona: Autoregressive diffusion world model for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

[48] Weiheng Zhao, Haoyi Jiang, Xin Shi, Liu Liu, Fan Huang, Zhizhong Su, Wei Sui, and Xinggang Wang. Faster-WAM: Efficient inference-time future conditioning for robust world action models. *arXiv preprint arXiv:2608.04404*, 2026.

[49] Zhida Zhao, Talas Fu, Yifan Wang, Lijun Wang, and Huchuan Lu. From forecasting to planning: Policy world model for collaborative state-action prediction. In *Advances in Neural Information Processing Systems*, 2025.

[50] Zongchuang Zhao, Xin Zhou, Tianyang Xu, Zhengyang Sun, Kaixuan Zhou, Yu Wu, Honglin Li, Dingkang Liang, and Xiang Bai. SimWAM: A simple world action model for end-to-end autonomous driving. *CoRR*, abs/2608.07468, 2026.

[51] Ruiguo Zhong, Benshan Ma, Xiaolong Chen, Lang Zhang, Mingyue Feng, Yaonong Wang, Pei Liu, and Jun Ma. DA-WAM: Decision-aligned future latents for driving world models. *CoRR*, abs/2608.19085, 2026.

[52] Renping Zhou, Zanlin Ni, Zihao Fan, Guohao Fu, Zeyu Liu, Hao Shi, Jie Zhang, Chi Bene Chen, Yang Yue, Xueyang Fu, and Gao Huang. What makes world action models generalize? An empirical study of test-time future modeling. *arXiv preprint arXiv:2609.34981*, 2026.

[53] Yueting Zhu, Shaoyu Chen, Yuehao Song, Hui Sun, Qian Zhang, Wenyu Liu, and Xinggang Wang. ReDrive: Shaping representations with world modeling for end-to-end driving. *arXiv preprint arXiv:2609.33854*, 2026.

A Implementation details

Architecture and inputs. Front-camera images are resized to 352×640. The frozen causal Wan tokenizer [34] emits one latent of shape $16 \times 44 \times 80$ per observed frame, using its retained state. The current image and logged continuation are encoded separately for the world segment, which contains one current and three future latents ($F = 3$). Spatial 2×2 patching gives 880 tokens per history latent and 3,520 tokens in the world segment. The backbone's native chunks hold two latents. History positions follow the stream index, and world positions are relative to the newest observation. The current-blind mask blocks access from the current latent to its first future chunk-mate.

The video backbone has 28 blocks of width 2048, with 16 attention heads of dimension 128. It is initialized from the public map-conditioned OmniDreams weights [2]. The trajectory expert has 28 blocks of width 1536 and feed-forward width 6144, using the same attention head space. Compatible attention and feed-forward weights are initialized by resizing backbone weights. Residual gates and the output layer start at zero. The image branch encodes an independently tokenized 384×672 current image into 252 tokens of width 256.

Each backbone block cross-attends to a text context. Let e_0 be the backbone's projection of a null-text row and P_ϕ the ego conditioner. World tokens receive the two-row context

$$E_q = [e_0;\ e_0 + P_\phi(q)], \tag{5}$$

and history tokens the null context $[e_0; e_0]$. P_ϕ projects the 24-dimensional q through a 1024-wide SiLU MLP to a 1024-dimensional offset, with a zero-initialized final layer. It belongs to the backbone optimizer group and is included in the exponential moving average (EMA). Using two cross-attention rows, rather than appending the ego row to the backbone's hundreds of identical null-text rows, keeps the ego input from being diluted; with equal initial rows, the pretrained null-only response is preserved exactly. As in SimWAM [50], the narrower expert projects its queries into the backbone's per-head attention space and projects the result back to its own width; timestep-modulated residual blocks predict the action velocity.

Supervised initialization. An initial supervised stage uses four-frame clips, a causal hold adapter with 13 RGB slots, local four-latent rotary grids for both segments, and ego context throughout the backbone. Its final EMA initializes streaming training. This stage uses AdamW with $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$ and weight decay 0.01. Both parameter groups peak at 7×10^{-5} after 2,000 warm-up updates and decay by a cosine schedule to 10^{-6}. Global batch size is 32 on four NVIDIA H200 GPUs with micro-batch eight. Training runs for 25 NAVTRAIN passes, or 80,694 updates. Gradients are clipped at norm 1.0 and activation checkpointing is enabled. EMA uses decay 0.9999 with an initial warm-up. Both flow schedules use shift $s = 5$; the video weight is $\lambda_v = 3$, the endpoint probability $p = \frac{1}{2}$ and the endpoint video weight one. Inference uses $K = 10$ flow steps with fixed action and future-noise seeds. Plans contain $T = 8$ poses over four seconds. Coordinates are normalized using lower bounds $(-1.57, -19.68, -1.67)$ and ranges $(66.74, 42, 3.53)$ for (x, y, ψ); yaw is wrapped after denormalization.

Losses and scheduler weights. For $x = \mathrm{norm}(A^\star)$, independent action noise ϵ_a and video noise ϵ_v, the two objectives in Eq. (4) are

$$\begin{aligned} \mathcal{L}_{\mathrm{act}} &= \mathbb{E}\big[w(\sigma)\,\mathrm{MSE}(v_\theta(x^\sigma, \sigma; c_\tau, q), \epsilon_a - x)\big], \\ \mathcal{L}_{\mathrm{video}} &= \mathbb{E}\Big[\tfrac{\omega(\tau)}{F} \textstyle\sum_{f=1}^{F} \mathrm{MSE}(u_{\phi,f}, \epsilon_{v,f} - y_f)\Big], \end{aligned} \tag{6}$$

where $u_{\phi,f}$ is the video head's velocity for future latent f. Action flow time follows the shifted schedule $\mathcal{S}_s(u) = su/(1 + (s-1)u)$ with $u \sim \mathcal{U}(0,1)$; video flow time is 1 with probability p and $\mathcal{S}_s(u_v)$ otherwise, shared by all future latents while observed latents stay clean. Both noise regimes contribute to both losses. Video loss updates the backbone and its ego conditioner, while action loss also updates the current-image encoder and the expert; the tokenizer stays frozen. In the final training stage, older history is computed without gradients but stays in the attention context; gradients pass through the newest history chunks and the world segment.

Let $r(t) = \exp[-2(t - \frac{1}{2})^2] - \exp[-\frac{1}{2}]$ and $N = 1000$. The continuous scheduler normalizes this weight over its shifted reference grid:

$$Z = \frac{1}{N}\sum_{i=1}^{N} r(\mathcal{S}_5(i/N)), \qquad w(t) = \frac{r(t)}{Z + 10^{-10}}. \tag{7}$$

For sampled video flow times, $\omega(\tau) = w(\tau)$. Pure-noise mixture samples instead use $\omega(1) = 1$. This is the training weight in the released SimWAM continuous-flow scheduler, with Simple-WAM's endpoint override [50, 52].

A.1 Streaming training

Streaming training starts from the initial supervised EMA at update 80,694. The backbone, ego conditioner, current-image branch and trajectory expert are fine-tuned jointly for 10,000 updates, or 320,000 scene presentations. The tokenizer remains frozen. AdamW uses the same betas, epsilon and weight decay as the initialization stage, with a peak learning rate of 2×10^{-5} after 500 warm-up updates and cosine decay to 10^{-6}. Global batch size is 32 on four GPUs. EMA decay is 0.9995 after its initial warm-up. Gradient clipping, activation checkpointing and both flow-matching objectives are unchanged. Results use the final EMA, not a selected intermediate checkpoint.

The history cache covers all 103,288 training scenes and stores 246,165 available camera frames across 9,339 runs in 1,192 logs. A run ends at a missing image or a timestamp gap greater than two seconds. Within a run, the first image is encoded alone. Each later latent encodes three copies of the preceding image followed by the new image, carrying the tokenizer's causal state between calls. Training uses local positions within each sample, while deployment retains stream positions. The action rotation in Eq. (3) makes the action interface invariant to this common temporal offset.

A.2 Reward-based policy refinement

Data and visual context. Refinement draws from 85,190 NAVTRAIN scenes, and 12,660 scenes from disjoint recordings are held out for development. The training pool combines 29,254 hard scenes, mined with an earlier supervised model, and 14,627 ordinary scenes drawn once at random from the remaining training scenes, so hard scenes make up two thirds of the pool. Rewards are official non-reactive NAVSIM-v1 PDM scores, computed for each candidate against the fixed PDM-Closed reference [7]. A second policy uses the NAVSIM-v2 EPDMS reward, which adds lane keeping, traffic-light compliance and comfort between consecutive plans. Both policies start from the supervised policy.

Each decision resets the policy and prefills four observed frames with $W = 2$. A fixed future-noise realization supplies the predictive features. The tokenizer, video backbone, backbone ego conditioner and current-image branch remain frozen. Each scene supplies 32 stochastic paths and one deterministic monitoring plan; the latter is excluded from advantages and policy losses. The full trajectory expert is trainable.

Short sampler and SDE exploration. The trajectory expert uses the reference scheduler's four-step grid, $1 = \sigma_1 > \cdots > \sigma_{K+1} = 0$, with $x_1 = 0$ and $\Delta_k = \sigma_{k+1} - \sigma_k < 0$. Let $v_k = v_\theta(x_k, \sigma_k; c, q)$. During training, the transition is

$$\begin{aligned} g_k &= \eta\sqrt{\sigma_k(1-\sigma_k) + \varepsilon_g}, \\ b_k &= v_k + \frac{g_k^2}{2\sigma_k}\big(x_k + (1-\sigma_k)v_k\big), \\ m_{\theta,k} &= x_k + \Delta_k b_k, \qquad a_k = g_k\sqrt{-\Delta_k}, \\ x_{k+1} &= m_{\theta,k} + a_k\xi_k, \quad \xi_k \sim \mathcal{N}(0, I_{24}). \end{aligned} \tag{8}$$

The drift follows the score-corrected flow-SDE construction used by Flow-GRPO [22]. We use $\eta = 0.018$ and $\varepsilon_g = 0.25$, giving positive variance at every transition, including $\sigma_1 = 1$. The conditional density is $p_{\theta,k} = \mathcal{N}(m_{\theta,k}, a_k^2 I_{24})$. Noise amplitudes remain fixed during rollout reuse. Training uses this discrete stochastic policy; deployment sets $\eta = 0$, recovering $x_{k+1} = x_k + \Delta_k v_k$.

Scene-centered rewards. For the $G = 32$ noisy rewards, define

$$\widehat{\mathcal{A}}_i = \frac{R_i - \bar{R}}{d_R}, \qquad \bar{R} = G^{-1}\sum_{j=1}^{G} R_j. \tag{9}$$

We use the fixed scale $d_R = s_R = 0.1$, following Dr. GRPO's removal of group-specific standardization [27]. A common denominator preserves the size of reward differences across scenes, so nearly tied plans receive smaller advantages than plans separated by large reward differences. Groups with constant rewards have zero advantages.

Clipped updates and reference anchor. For a stored path $\zeta_i = (x_{i,2}, \ldots, x_{i,K+1})$ with $x_{i,1} = 0$, the conditional log density is

$$\ell_i(\theta) = \sum_{k=1}^{K} \log p_{\theta,k}(x_{i,k+1} \mid x_{i,k}, c, q), \tag{10}$$

which sums $3T = 24$ coordinates at each of the $K = 4$ transitions. We temper its change before clipping:

$$\rho_i(\theta) = \exp\left[\frac{\ell_i(\theta) - \ell_i^{\text{old}}}{24K}\right], \tag{11}$$

where ℓ_i^{old} is the behavior log density recorded at sampling. This is the geometric-mean density ratio over 96 transition coordinates. The loss is

$$\begin{aligned}\mathcal{L}_{\text{RL}}(\theta) = -\mathbb{E}_i\big[&\min\{\rho_i \widehat{\mathcal{A}}_i, \\ &\text{clip}(\rho_i, 1-\delta, 1+\delta)\widehat{\mathcal{A}}_i\}\big] \\ &+ \lambda_{\text{ref}}\mathcal{L}_{\text{ref}}(\theta).\end{aligned} \tag{12}$$

The reference penalty anchors velocities on the stored states:

$$\mathcal{L}_{\text{ref}}(\theta) = \mathbb{E}_{i,k}\,\text{MSE}(v_\theta(x_{i,k}, \sigma_k; c, q), v_{\theta_0}(x_{i,k}, \sigma_k; c, q))\,. \tag{13}$$

The teacher θ_0 is the expert from which refinement starts. It, the sampled states, old log densities, rewards and frozen visual features are detached. Gradients reach θ through transition densities and the reference penalty. The expert's auxiliary projections are recomputed during each optimizer pass.

Optimizer and budgets. We use AdamW with learning rate 10^{-6}, betas $(0.9, 0.95)$, zero weight decay and 100 warm-up updates. The clipping threshold is $\delta = 0.02$, the anchor coefficient is $\lambda_{\text{ref}} = 100$, and the gradient norm is clipped at 1.0. Four workers, one per GPU, collect two scenes each. Each rollout batch supports four optimizer updates, with the same states, old likelihoods and rewards reused. We report the PDMS-reward policy after 40,000 updates and the EPDMS-reward policy after 45,000.

B Evaluation details

Baselines and reported policies. Table 5 lists every method we compare against. Table 2 includes the peer-reviewed world-action models and, from earlier preprints, the methods closest to AR-WAM in design or with the best aggregate scores. Concurrent work (○) appears only in Table 5. All baseline values are published results, and the reference planners' values come from their own papers. DriveLaW's NAVSIM-v2 values and DriveVLA-W0's NAVHARD value are taken from SimWAM [50], whose supervised Wan2.2 and RL Cosmos2.5 rows come from its backbone ablation, which does not report comfort. Each baseline keeps its published training setup: SimWAM uses 85,109 training samples, MM-Future and SV-WAM also train on NAVVAL, and the DriveVA row is its NAVSIM-only model (DriveVA reaches 90.9 PDMS when CARLA data are added) [24, 25, 36]. In the Cam. column, M marks multiview input without a stated camera count. AR-WAM's two RL rows are separate policies, refined with the PDMS or the EPDMS reward, and both plan with four integration steps. Our NAVSIM-v2 scores apply the official human-reference filter of the NAVSIM devkit, and DrivoR's scores are those of its NAVTRAIN-only model, with NAVHARD under the same filter [15]. In the dynamic setting and on nuScenes, SimWAM, UNIVERSE, DriveVA and DriveWAM run their released supervised checkpoints with their native inputs. The single released DriveVA checkpoint is labeled 90.9 PDMS and scores 90.1 on NAVTEST with our scorer, and the released DriveWAM checkpoint scores 90.1, matching its reported value. UNIVERSE does not report NAVHARD; its score in Fig. 1 is our evaluation of the released checkpoint.

nuScenes. Our evaluation under DriveVA's protocol uses the validation keyframes that have four past and at least six future keyframes. Ground-truth poses use the rear-axle ego frame at front-camera timestamps. The command follows the final future heading with an $8°$ threshold, as in the reference evaluation. L2 error is interpolated at 1, 2 and 3 s. An oriented ego box is rasterized against annotations on a 0.1 m grid, with its centre 1.49 m ahead of the rear axle and heading from displacement (zero below 1 m). Logged-colliding steps are excluded from predicted-collision counts according to that protocol, and collision rates average keyframes up to each horizon. Velocity and acceleration come from CAN signals. Scoring uses DriveVA's released code. Its final sample filter is not released, so our 4,519 samples may differ from the published sample sets. On them, the released SimWAM and UNIVERSE checkpoints reproduce their reported average L2 of 0.96 and 0.83 m within 0.01 m, the released DriveVA checkpoint gives 0.87 m against a reported 0.84 m, and the released DriveWAM checkpoint gives 0.91 m against the 0.96 m that SimWAM reports for it [50]. SimWAM's average collision rate is within 0.01 points of its reported value, whereas UNIVERSE's is 0.07% against a reported 0.03%.

Dynamic setting. Each decision uses the logged history at its time. Other vehicles may react during scoring, but predicted sensor observations are not fed into the next policy input. Anchor-frame scores can differ from standalone

Table 5: All compared methods on NAVTEST and NAVHARD, with the columns and notes of Table 2. ○: concurrent work. †: reward or preference post-training; the DriveReferee row is its distilled single-pass policy. ‡: proposal scoring. Bold: best result in each column, not counting the human driver and proposal-scoring rows.

			NAVTEST, NAVSIM-v1						NAVTEST	NAVHARD
Method	Ref.	Cam.	NC	DAC	TTC	Comf.	EP	PDMS	EPDMS	EPDMS
Reference planners without a world model										
Human driver [7]	–	–	100.0	100.0	100.0	99.9	87.5	94.8	–	–
DiffusionDrive [20]	CVPR'25	3+L	98.2	96.2	94.7	**100.0**	82.2	88.1	–	–
ReCogDrive† [17]	ICLR'26	1	97.9	97.3	94.9	**100.0**	87.3	90.8	–	–
DrivoR‡ [15]	CVPR'26	4	98.9	98.3	96.2	100.0	89.1	93.1	–	48.3
World-action models, direct prediction, supervised										
Epona [47]	ICCV'25	1	97.9	95.1	93.8	99.9	80.4	86.2	–	–
DriveVLA-W0 [16]	ICLR'26	1	–	–	–	–	–	–	86.1	24.4
PWM [49]	NeurIPS'25	1	98.6	95.9	95.4	**100.0**	81.8	88.1	–	–
DriveLaW [39]	CVPR'26	1	99.0	97.1	96.7	**100.0**	81.3	89.1	88.6	30.6
DriveWAM [30]	NeurIPS'26	1	98.3	98.1	95.2	**100.0**	84.3	90.1	–	–
SV-WAM [36]	arXiv	6	98.6	98.6	95.1	**100.0**	83.9	90.2	91.0	36.1
SimWAM, Wan2.2 [50]	arXiv	1	98.6	98.0	95.8	–	84.0	90.3	90.2	37.6
DriveVA [24]	ECCV'26	1	99.0	97.3	98.4	**100.0**	83.2	90.5	–	–
GeoWAM [28]	arXiv	M	–	–	–	–	–	–	90.7	39.6
SUV [45]	arXiv	1	99.1	97.8	96.7	**100.0**	84.6	90.8	91.0	36.9
4D-WAM [9]	arXiv	1	99.1	97.9	96.6	**100.0**	84.6	90.9	90.6	35.9
UNIVERSE [23]	NeurIPS'26	1	99.1	97.6	**98.5**	**100.0**	83.6	91.0	–	–
ReDrive○ [53]	arXiv	1	99.1	97.9	97.2	**100.0**	84.3	91.0	90.8	34.4
PhysWAM○ [29]	arXiv	3	99.0	97.5	**98.5**	99.8	88.7	91.4	90.3	38.1
WA-JEPA [38]	arXiv	4	**99.5**	98.3	97.7	**100.0**	85.0	91.8	91.7	–
World-action models, direct prediction, post-trained										
VITAL†○ [19]	arXiv	1	98.7	98.5	95.8	**100.0**	84.8	90.8	91.1	38.6
SimVTA†○ [13]	arXiv	1	98.0	97.2	97.0	**100.0**	86.7	91.3	88.2	–
AffordDrive3D†○ [5]	arXiv	3	98.6	99.0	96.3	**100.0**	84.7	91.3	89.9	–
DriveUniWAM†○ [40]	arXiv	1+L	98.7	98.0	95.6	99.8	87.4	91.5	–	–
SimWAM, Wan2.2†	arXiv	1	98.5	99.1	96.1	99.8	86.5	91.9	–	–
DriveReferee†○ [43]	arXiv	1	–	–	–	–	–	92.0	91.6	–
DriveWLA†○ [37]	arXiv	1	99.1	98.8	97.0	**100.0**	86.4	92.3	90.1	–
SimWAM, Cosmos2.5†	arXiv	1	98.8	99.2	96.4	–	87.0	92.4	–	–
World-action models, proposal scoring										
Adaptive-WAM† [1]	arXiv	1	98.6	97.9	95.6	100.0	85.1	90.8	89.9	–
MORE○ [10]	arXiv	3	98.8	98.4	95.6	100.0	90.7	93.2	92.5	–
DA-WAM [51]	arXiv	1	99.1	98.9	96.8	99.8	90.0	93.7	89.2	–
PlanWAM○ [41]	arXiv	4	98.9	98.8	96.3	100.0	91.0	93.8	90.9	–
MM-Future○ [25]	arXiv	4	98.7	99.0	95.8	100.0	91.6	94.0	91.5	–
AR-WAM		1	99.4	98.2	98.0	**100.0**	84.5	91.5	90.9	38.6
AR-WAM + RL, PDMS†		1	99.0	**99.3**	96.1	**100.0**	**90.7**	**94.0**	–	–
AR-WAM + RL, EPDMS†		1	99.3	**99.3**	97.1	**100.0**	86.6	92.7	**92.2**	**42.3**

evaluation because extended comfort pairs consecutive plans of the stream. After waiting frames are excluded, each remaining frame keeps the score computed in the continuous stream, including extended comfort relative to the preceding plan.

C Runtime validation

The accelerated implementation preserves the current-blind mask, ego-conditioned world cross-attention and width-1536 trajectory expert. A static per-layer ring holds $2W$ history latents. Separate regions hold world, auxiliary, action and open-latent K/V. FlashAttention reads the valid ranges without concatenating the cache at every action step. Split attention outputs are merged with their log-sum-exp normalizers. Timestep modulations are precomputed, and CUDA graphs capture completed-chunk, world and action updates. Inductor compiles the block computations.

Table 6: Per-decision latency, peak GPU memory and NAVTEST PDMS at batch size one on an H200. Hist.: camera frames used per decision, which AR-WAM reads from its cache. Future: future-video computation at inference. Baseline PDMS values are publication-reported. Bold: best result in each column. †Three cameras and LiDAR. ‡Publication-reported mean on an A100.

	Hist.		Latency (ms)		Mem.	
Method	(fr.)	Future	Med.	P90	(GB)	PDMS
DiffusionDrive†	1	none	**40.7**	**52.5**	**0.3**	88.1
SimWAM	1	dropped	499.1	933.9	13.8	90.3
Adaptive-WAM‡	1	early exit	170	–	–	90.8
UNIVERSE	5	dropped	256.9	327.2	24.4	91.0
UNIVERSE	5	3 steps	341.8	350.9	24.4	–
DriveVA	5	3 steps	258.8	316.3	24.4	90.9
DriveWAM	1	rollout	895.7	905.4	26.1	90.1
AR-WAM, $K{=}10$	5	1 pass	146.3	154.8	22.5	91.5
AR-WAM, $K{=}4$	5	1 pass	132.8	137.3	22.5	**91.6**

History cross-attention to identical null rows has a constant output, while the ego-conditioned world cross-attention remains query-dependent. The last backbone block supplies only the K/V consumed by the expert. Cached decisions agree with recomputation of the same causal prefix, including the context that formed retained states.

Both runtimes use BF16 computation; the accelerated path also stores BF16 weight copies. On a 1,024-scene NAVTEST subset with cold four-frame inputs and batch size four, the accelerated runtime scores 91.704 PDMS against 91.705 for the reference runtime with the same checkpoint. Mean per-scene trajectory separation is 1.28 cm, with a 90th percentile of 2.47 cm and a maximum per-scene mean of 5.16 cm.

For Table 6, each method runs the first 40 frames of three log segments after model loading and compilation, giving 115 to 117 timed decisions per configuration after warm-up, on an otherwise idle GPU. Memory is the peak PyTorch allocation, including weights. Adaptive-WAM has no released code, so its reported A100 mean is quoted. Timing starts from decoded RGB frames in host memory and ends with the trajectory on the CPU, so it includes resizing, encoding, the ego vector and action sampling. AR-WAM issues the newest frame's three encodes concurrently on separate CUDA streams; its plans agree bitwise with sequential encoding on all 120 frames for every cached history and step count. Baselines use their released NAVSIM settings. UNIVERSE takes five history frames at 480×832 and three flow steps, either trajectory-only or jointly with future video; the joint row excludes decoding the predicted frames to pixels, which its released code adds and which raises the median to 732 ms. DriveVA takes the same five frames and denoises future video latents together with the trajectory in three flow steps without decoding them to pixels. DriveWAM takes the current front frame at 256×448, rolls out one future video latent in three steps and then the trajectory in ten. SimWAM takes the current front frame and ten flow steps. DiffusionDrive takes the current left, front and right cameras and two denoising steps. Our NAVTEST sensor data contain no LiDAR, so DiffusionDrive receives a synthetic 100,000-point sweep, which enters only through a fixed-size BEV histogram.

AR-WAM + RL uses the same network as AR-WAM and plans with four steps, so its latency is that of the $K{=}4$ row in Table 6.